

WEB APPLICATION FOR THE RESEARCH ETHICS COMMITTEE

William H. Muter

A Dissertation submitted to
the School of Computing Sciences of The University of East Anglia
in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE.
OCTOBER, 2018

SUPERVISOR(S), MARKERS/CHECKER AND ORGANISER

The undersigned hereby certify that the markers have independently marked the dissertation entitled “**Web Application for the Research Ethics Committee**” by **William H. Muter**, and the external examiner has checked the marking, in accordance with the marking criteria and the requirements for the degree of **Master of Science**.

Supervisor:

Dr. Dan Smith

Markers:

Marker 1: Dr. Dan Smith

Marker 2: Dr. Pierre Chardaire

External Examiner:

Checker/Moderator

Moderator:

Dr. Wenjia Wang

DISSERTATION INFORMATION AND STATEMENT

Dissertation Submission Date: **October, 2018**

Student: **William H. Muter**
Title: **Web Application for the Research Ethics Committee**
School: **Computing Sciences**
Course: **Knowledge Discovery and Datamining**
Degree: **M.Sc.**
Duration: **2017-2018**
Organiser: **Dr. Wenjia Wang**

STATEMENT:

Unless otherwise noted or referenced in the text, the work described in this dissertation is, to the best of my knowledge and belief, my own work. It has not been submitted, either in whole or in part for any degree at this or any other academic or professional institution.

Permission is herewith granted to The University of East Anglia to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Student

Abstract

Within the University of East Anglia, for some projects or modules that involve interactions with external parties for research purposes or deal with potentially sensitive topics, ethics clearance may be required to protect the reputation of the university. Presently, this paper-based system is laborious and time-consuming, and it can often take a considerable amount of time for documents to find the relevant parties for processing. This dissertation project seeks to produce a web-application in order to partially automate this system, thus increasing the speed of the ethics clearance application processing whilst simultaneously reducing the amount of paper and resource wastage. Through the investigation of current development standards, with particular attention to web accessibility and usability, a website will be constructed in order to achieve this. To conclude, a system evaluation will be conducted in order to establish the feasibility for implementation within the university, and recommendations will be made as to how the system can be further improved and developed in the future.

Acknowledgements

This dissertation project marks the conclusion of my studies for the degree of Master of Science in Knowledge Discovery and Datamining. As such, I wish to express my gratitude to the following individuals for their constant support:

- Dr Dan Smith
- Dr Beatriz de la Iglesia
- Dr Wenjia Wang
- Dr Michal Mackiewicz

I am also very grateful to my friends and family for their patience and love.

William H. Muter

Table of Contents

Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Dissertation Structure	1
1.2 Background and Motivation	1
1.2.1 Description of Problem	1
1.2.2 Project Motivation	2
1.3 Aim and Objectives	2
1.3.1 Overall Aim	2
1.3.2 Project Objectives	2
2 Literature Review	3
2.1 Current System	3
2.1.1 Issues	3
2.1.2 Considerations	4
2.2 User Experience	4
2.2.1 Elements of User Experience	6
2.2.2 UX Deliverables	7
2.3 Accessibility	9
2.3.1 Standards and Regulations	9
2.3.2 Tools for Evaluation	10
2.3.3 Visual Impairments	10
2.3.4 Accessibility and User Experience	11
2.4 Languages and Tools	11
2.4.1 JavaScript	11
2.4.2 Node.js	11
2.5 System Security	12
2.5.1 Vulnerabilities	12
2.6 System Configuration	14
2.6.1 System Architecture	14
2.6.2 Database Design	14

3	Methodology	16
3.1	Feasibility	16
3.1.1	Integration and Management	16
3.1.2	Scheduling	16
3.2	Requirements Elicitation	19
3.2.1	Technical Requirements	19
3.2.2	Functional Requirements	20
3.3	Use Case Descriptions	21
3.3.1	Use Case 1	21
3.3.2	Use Case 2	22
3.3.3	Use Case 3	23
3.3.4	Use Case 4	24
3.4	Risk Analysis	25
3.4.1	Risks Factors	25
3.4.2	Probability and Impact Matrix	26
3.5	Legal, Social and Ethical Considerations	27
3.5.1	Accessibility	27
3.5.2	Data Protection	27
3.6	Technical Considerations	28
3.6.1	External Libraries	28
3.6.2	System Architecture	29
4	Development of System	30
4.1	Database Design	30
4.1.1	Languages and Tools	30
4.1.2	Database Schemas	30
4.1.3	Data Types	31
4.1.4	Attributes	32
4.1.5	Input Validation	32
4.2	Implementation of Node.js	33
4.2.1	Package Management	33
4.2.2	Frameworks	34
4.2.3	Routing	34
4.2.4	Template Engines	35
4.3	Styling and Design	36
4.3.1	Tools for Consistency	36
4.3.2	Design Features	38
4.3.3	Web Interoperability	41
4.4	Accessibility and Usability	43
4.4.1	Measures Taken	43
4.4.2	User Support	45
4.5	Security Features	46
4.5.1	Password Encryption	46
4.5.2	Vulnerability Databases	46
4.5.3	Cross Site Scripting (XSS)	47

5	Evaluation and Discussion	49
5.1	Website Evaluation	49
5.1.1	User Assessment	49
5.1.2	Future Evaluation	51
5.1.3	Accessibility Testing	52
5.2	Implementation Feasibility	54
5.2.1	Evaluation of Deliverable	54
5.2.2	System Adoption	55
5.2.3	Necessary Amendments	55
5.3	Testing	56
5.3.1	Test Cases	56
5.3.2	Integration Testing	56
6	Conclusions	57
6.1	Project Summary	57
6.1.1	Overall Outcome	57
6.1.2	Final Thoughts	58
6.2	Further Work	58
	Bibliography	59
A	Example Screenshots of Web Application	64
A.1	Index Page	64
A.2	Administrator Perspective	65
A.2.1	Dashboard	65
A.2.2	Form Management	66
A.3	User Perspective	68
A.3.1	New Application	68
A.3.2	Dashboard	68

List of Tables

3.1	Project Tasks	17
3.2	Project Milestones	17
3.3	MoSCoW Analysis - Technical Requirements	19
3.4	MoSCoW Analysis - Functional Requirements	20
3.5	Use Case Description 1 - A user makes an ethics application.	21
3.6	Use Case Description 2 - An administrator creates a new form.	22
3.7	Use Case Description 3 - Administrator edits an existing form.	23
3.8	Use Case Description 4 - An administrator leaves a comment.	24
3.9	Major Risk Factors	25
3.10	Probability / Impact Table	26
4.1	Database Schemas and Their Attributes	32
4.2	Packages Utilised	33
4.3	WCAG Colour Contrast Compliance for Colour Blind Users	39
4.4	Actively Tested jQuery Browser Support	41
4.5	Bootstrap Responsive Breakpoints	42
4.6	Font derivation in Bootstrap	44
5.1	Breakdown of Accessibility Evaluation for WCAG 2.0 AAA Standards .	53

List of Figures

2.1	Nielsen’s 10 Heuristics for Interface Design (Nielsen, 2000)	5
2.2	Navigation Bars of Five Popular Websites	5
2.3	The Elements of User Experience (Garrett, 2003)	6
2.4	OWASP Top 10 owasp.org (2017)	13
2.5	Root Causes of Session Hijacking (Huluka and Popov, 2012)	14
2.6	Model-View-Controller Architecture.	15
3.1	Gantt Chart of Tasks	18
3.2	Probability and Impact Matrix	26
3.3	Model-View-Controller	29
4.1	Bootstrap Buttons - Example	36
4.2	Bootstrap Modal - Implementation of modal to confirm logout.	36
4.3	A simple diagram detailing how EJS partials work - Adding the file path after “include” allows for the template to be imported.	37
4.4	Traffic Light Colours Implemented for Context	38
4.5	Accordion in Usage on an Existing UEA System (evision) - Closed (top) and Open (bottom).	40
4.6	Tabs Allow Users to Toggle Content - All Results (top) and Search Bar (bottom).	40
4.7	Responsive Web Design - The display of the menu is changed based on screen dimension - Desktop (top) and Mobile (bottom).	43
4.8	Feedback after entering a non-existent username (top) and an incorrect password (bottom).	45
5.1	Types of Feedback Sammaneh (2018).	51
5.2	AChecker analysis input.	52
5.3	AChecker analysis output.	53
A.1	Index page (application home page)	64
A.2	Administrator dashboard - all results shown	65
A.3	Administrator dashboard - search bar	65

A.4	Form creation - component selector page	66
A.5	Form can be created by administrator	66
A.6	Form created and ready for population by applicant	67
A.7	Form can then be disabled by administrator	67

Chapter 1

Introduction

1.1 Dissertation Structure

This report will begin by first outlining the background and motivation behind the project, establishing the core aim and objectives (both functional and technical). This will then be followed by a comprehensive review of relevant literature related to both project-specific topics as well as the development of web applications in general. After this literature review, the methodologies used will be documented paying particular attention to system engineering. The development process will then be described; highlighting any technical decisions made with justifications. The penultimate section of this dissertation piece will provide an evaluation of the system, highlighting where the objectives have been reached as well as the shortcomings, followed by a summary of the feasibility of implementing this system at the university. Finally, this report will conclude with an overall summary, documenting the final deliverable and potential areas for future work.

1.2 Background and Motivation

1.2.1 Description of Problem

Within the University of East Anglia, some research projects / modules require ethics consideration in order to safeguard staff, students, those undertaking research, potential participants, and the reputation of the university.

Currently, the process of applying for ethics clearance can be both laborious and time-consuming for all parties involved. It is a paper-based system, and in instances where multiple individuals are required to evaluate an application, it can take a substantial amount of time for documents to circulate and feedback to be provided. This is why the need for a centralised application feedback system has been identified; to allow relevant parties to evaluate and either deny or approve ethics applications.

1.2.2 Project Motivation

The motivation for this project stems from both personal and professional reasons. During my time at the University of East Anglia I am familiar from first-hand accounts of how laborious and time consuming this process can be, and so there is the potential to design a solution that will serve a purpose solving a real-world problem. From a professional perspective, it will provide me with an opportunity to improve my web-based programming skills and enhance my system development abilities.

1.3 Aim and Objectives

1.3.1 Overall Aim

To design, develop, and evaluate a web-based system for the management of applications made to the University of East Anglia Research Ethics Committee.

1.3.2 Project Objectives

These are the objectives which will need to be completed in order to achieve the aim of this project:

- Understand the issues with the current system
- Perform background research and study any relevant literature
- Use my findings to design and construct a new automated system
- Perform a comprehensive system evaluation

Chapter 2

Literature Review

2.1 Current System

The system currently in place within the University of East Anglia for ethics approval is a paper-based one, whereby applicants are either provided with or find the appropriate application forms, fill them in with the correct details / attachments, and then return them to their school's ethics committee representative (where there may be conflicts of interest, some applications are referred to another person for review).

The application is then considered and any issues or necessary changes are highlighted, and is either rejected or approved. If an application is rejected, applicants then reassess the issues highlighted, adjust their application, then resubmit; reiterating through this process again.

2.1.1 Issues

The main issue for this system currently is the length of time it can take for feedback to be provided. As this is a paper-based system it can take a while for documents to find their way into the hands of the appropriate individuals, adding to the estimated application times. An additional issue is that currently, if applications require multiple individuals to review them, ensuring all parties have sufficient access can also pose problems.

2.1.2 Considerations

I have highlighted three key areas of particular concern which should be given additional attention during the design process for the automation of this system; with these areas being security, usability, and accessibility. Security is important as there could potentially be sensitive information being relayed within ethics applications and therefore confidentiality should be a constant area for consideration, with data encryption being investigated. This system will be utilised by non-technically minded individuals so therefore making the system as user-friendly and understandable as possible is a necessity. Additionally, this system will be used by individuals from a large and diverse population; so therefore ensuring that it can easily be used by as many individuals as possible is another necessity.

2.2 User Experience

User Experience (UX) refers to the attitudes of people regarding a particular product, service, or system, and “encompasses all aspects of the end-user’s interaction with the company, its services, and its products”. Although it has always been a salient area of website design, it has experienced a particular growth in research subsequent to improvements in displays and haptics. A prominent component of human-computer interaction, UX helps to explore and understand how the design of particular services impacts their feelings, with the goal being to make the experience as positive as possible in order to encourage effective use and repeat visits.

Krug (2014) discusses the importance of reducing frustration within users. He highlights straight-forwardness and simplicity as important, but emphasises that this shouldn’t compromise providing users with sufficient feedback. He particularly reinforces the notion of “don’t make me think”, whereby interface designers should always attempt to reduce the amount of cognitive effort required by users. Functionality should be obvious and understandable, whilst minimising the amount of clutter on a webpage, particularly if being accessed on smaller displays.

Jacob Nielsen suggested ten heuristics for user interface design, which provide guidelines to consider when designing an interface. These are:

- | | |
|--|---|
| 1. Visibility of system status | 6. Recognition rather than recall |
| 2. Match between system and the real world | 7. Flexibility and efficiency of use |
| 3. User control and freedom | 8. Aesthetic and minimalistic design |
| 4. Consistency and standards | 9. Help users recognize, diagnose, and recover errors |
| 5. Error prevention | 10. Help and documentation |

Figure 2.1: Nielsen’s 10 Heuristics for Interface Design (Nielsen, 2000)

An good example of “recognition rather than recall” can be seen in Figure 2.2, which shows the login page navigation bars of some of the most popular websites.

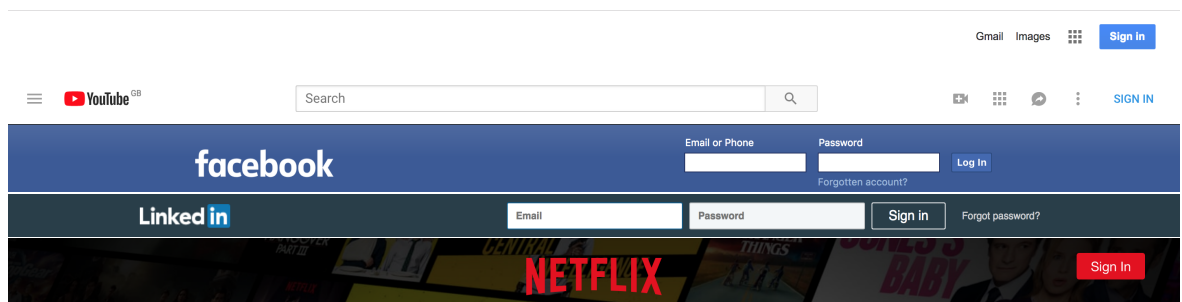


Figure 2.2: Navigation Bars of Five Popular Websites

Top to Bottom with Ranking (Alexa, 2018): Google (1st), YouTube (2nd), Facebook (3rd), Netflix (26th), and LinkedIn (28th)

As visible most of these websites have their logo accompanied with either inputs or a link to sign into an account. Many smaller companies copy this layout as it is what users have come to expect and are familiar with.

Nielsen (2000) also specifically discusses user interface design in web applications, and although this was published almost twenty years ago; some of the key points still hold true, particularly the notions of scannability and legibility. Using clear readable fonts and segmenting large chunks of text into more easily digestible paragraphs ensure users can properly absorb key information without losing their attention.

The U.S. Government also provide information regarding how businesses can better reach their online audiences. They suggest that content should satisfy eight criteria, these are useful (be original and fulfil a need), usable (easy to use), desirable (design elements are used to evoke emotion), findable (navigable and locatable onsite and off-site), accessible (accessible to people with disabilities), credible (users must trust and believe what you tell them) (usability.gov, 2018).

2.2.1 Elements of User Experience

According to Garrett (2003), User Experience (in the context of web design) can be broken into five levels of abstraction; visual design, navigation & information design, information architecture, content requirements, and user needs & site objectives.

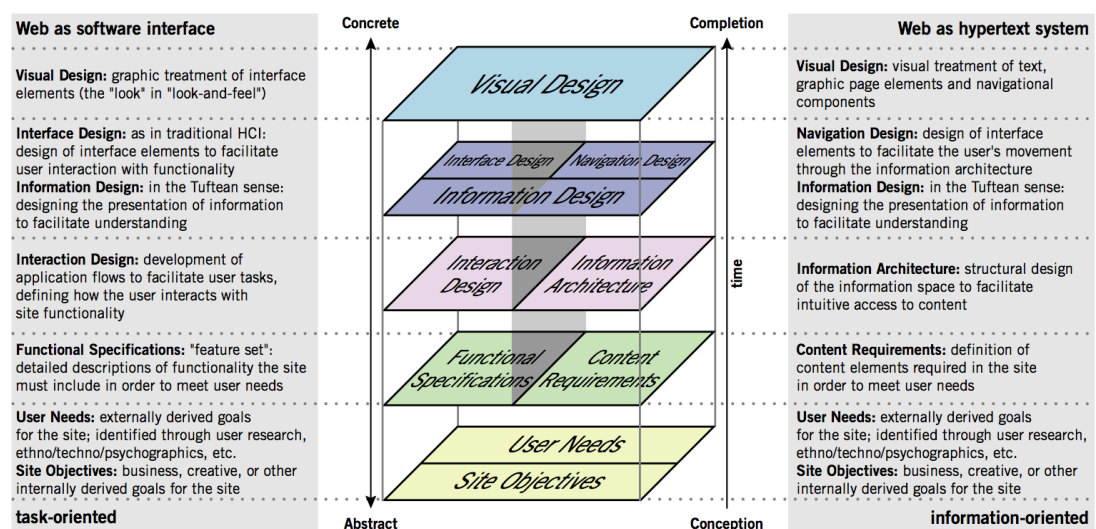


Figure 2.3: The Elements of User Experience (Garrett, 2003)

The two most abstract layers (user needs & site objectives and content requirements) should be defined during requirement elicitation; whereas information architecture is dependent on how users interact with the system. The two more concrete areas (navigation & information design and visual design) must be decided based on the technologies available and the accessibility constraints.

2.2.2 UX Deliverables

“User Experience (UX) Deliverables” is the collective name given to various visualisations or documents that can be compiled in order to express the usability of a system. There are many UX deliverables and therefore it is important to decide which methods are the most suitable for this particular scenario. For this project I have short-listed five different deliverables that should be investigated. These are personae, storyboards, use cases, user journeys, prototypes & user surveys.

User Personae

User personae are a collection of fictional characters, usually representative of the demographics most like to interact with a system, that are devised in order to predict likely needs and behaviours (Lidwell et al., 2010). Creating a persona generally starts by defining an objective, stating what the purpose of the website is. Then a character is defined by being given a brief description (comprised of their personal, professional and technical backgrounds) and a motivation (telling us what they are looking for from their experience). This character can then be used to predict how they are most likely to interact with the interface. The use of personae has been praised for preventing common shortfalls by encouraging designers to take a customer-centric perspective of the system, however there have been criticisms due to the stereotyping of users and the potential design oversights caused by the omission of key demographics.

Storyboards

Differing from developing a persona, a storyboard centres around the occurrence of a potential scenario rather than the general usage of a particular demographic. A grid of images representing the various issues that may be encountered is produced, displayed in the order they will appear, giving the perception that the viewer is witnessing a user interact with the system. Storyboards excel at giving faces to analytical data and stimulate empathy from developers toward users (Little, 2013). As they are task and goal oriented, they are also great at demonstrating how various oversights can lead to adverse outcomes.

Use Cases

Similarly to Storyboards, Use Cases are scenario-centric. A desired outcome is defined (usually a fairly commonly executed one), and the actions required to fulfil this objective are recorded. In addition to this ideal outcome, other alternatives, where unexpected actions (either user or system driven) occur, are also recorded. This allows for developers to consider the full scope of the scenario and mitigate against potential issues that users may encounter.

User Journeys

A user journey is the path taken by a user through a piece of software, and can be used to establish the ease of which a particular task can be completed, the complexity of navigation, and the number of ways any given task can be undertaken. Most commonly measured by the number of interactions required and pages visited, combined with the duration of the visit. In general the aim is to reduce both by as much as possible.

Prototypes & User Evaluation

One of the most effective methods of appraising the design of a system is to build a prototype and actually get users to interact with it. Although this is one of the more time-consuming methods for proof of concept, witnessing first-hand how easily users navigate their way around an interface allows for immediate understanding and correction of issues. Getting users to then partake in evaluations or surveys to understand their frustrations can then document and quantify this data, for further analysis. This can then be considered during the development of subsequent prototypes.

One important factor to consider when performing feedback evaluation is the Halo effect, which considers the idea that assessments of technology can be influenced by factors other than the actual user experience. The results reported herein suggest that the overall affect of these parties may unintentionally contaminate their assessment of a website's design, thereby leading to erroneous conclusions. Therefore efforts must be made to ensure that feedback is gathered in an unbiased way (Soper and Piepkorn, 2018).

2.3 Accessibility

Another important issue to consider is the accessibility of this web application; in January 2015 a document compiled at the University of East Anglia reported that 10.9% of the student population had declared some form of disability; and it is vital that all users have an equal ability to access this service (University of East Anglia, 2015). In addition to this, the legal requirements enforcing the equal access of content and guidelines in place to ensure this must be considered.

2.3.1 Standards and Regulations

The WCAG 2.0 standards (Caldwell et al., 2008), and the more recent WCAG 2.1 (Kirkpatrick et al., 2018) are the best known web accessibility guidelines available and can be used to gauge the overall accessibility of a web application. Schmutz et al. (2018) considered the how meeting of WCAG 2 standards effect the user experience of non-disabled users. They reported no conflicts of interest and found that AA rated pages reduced task completion time compared to pages without any accessibility measure when participants used a tablet but not when they used a laptop. Therefore it can be said that adhering to these guidelines is not only beneficial to the accessibility of the website for disabled users, but also to the user experience of non-disabled users.

The British Standards Institution (BSI) Group is the national standards body of the U.K. that produces technical standards for a wide range of products and services. In 2010, they published the British Standard 8878 (BS 8878), which defines processes for businesses that assist with the implementation of web accessibility strategy. It is aimed at management level, so is therefore written in non-technical language, and developing for the web it is a useful tool to ensure accessibility has been considered (The British Standards Institution, 2008).

2.3.2 Tools for Evaluation

There are various different tools that can be used to assess the accessibility of a webpage, as discussed by Acosta-Vargas et al. (2018), who suggest the use of AccessMonitor, AChecker, eXaminator, TAW, Tenon, WAVE and Web Accessibility Checker as suitable for ensuring that websites are compliant with current standards. In their study they emphasise the differences between warnings and failures, stating that although warnings are much less severe than failures, using these tools to reducing the number of warnings results in enhanced web accessibility.

2.3.3 Visual Impairments

Accounting for the loss or impairment of vision is one of the most significant barriers to ensuring accessibility for all users, impacting many people on varying levels, ranging from mild colour vision deficiency to complete blindness. Graham et al. (2007) state that superfluous information or detail should be suppressed, and that colour contrast can help a large range of individuals. They also found that sound is the best stimulus for input and output as it is inclusive to those who do not understand Braille, which suggests that efforts should be made to accommodate screen reading technology.

Giraud et al. (2018) performed three experiments that showed a substantial benefit of information filtering for cognitive load and abandon rate. Their study takes into account the needs of users with blindness and highlights the potential of a tool based on filtering redundant and irrelevant information, which allows reducing the cognitive load of users with blindness and improving interface usability.

Sirikitsathian et al. (2017) proposed a model for better understanding of factors influencing accessibility to a website and its acceptance by university students with visual impairments. They suggest mapping web accessibility and vision impairment levels against performance expectancy, effort expectancy, social influence and facilitating conditions.

2.3.4 Accessibility and User Experience

In a study undertaken by Aizpurua et al. (2016), results revealed that most user experience attributes are significantly correlated with perceived web accessibility indicating that perceived accessibility is related to hedonic and pragmatic qualities.

2.4 Languages and Tools

For this project I will be using Node.js to develop this system. The reason for this is that it Node is a rapidly growing professional platform, and so it would be useful to familiarise myself with its functionality.

2.4.1 JavaScript

When discussing Node.js, it is important to first briefly discuss JavaScript. JavaScript is a high-level programming language created by Brendan Eich, that was first released in 1995. JavaScript is utilised by almost 95% of websites (W3Techs - Web Technology Surveys, 2018) and is supported by all major web browsers. Its popularity is primarily due to its support of event-driven and functional programming styles, as well as its ability to allow for the dynamic manipulation of the Document Object Model (DOM), such as the ad-hoc alteration of HTML code.

2.4.2 Node.js

Originally, people associated JavaScript with front-end interface manipulation language. However, inspired by observing an upload progress bar, and noting that a query to the web-server was required to calculate how much of a file had been uploaded; developer Ryan Dahl sought to find a way to implement JavaScript server-side.

Initially released in 2009, Node.js is a runtime environment that allows for the server-side execution of JavaScript. According to Tilkov and Vinoski (2010), the core appeal of Node.js is its event-driven architecture, and ability to perform asynchronous I/O

processing (which is the most significant difference between Node and PHP). This difference is further demonstrated in an appraisal compiled by Chaniotis et al. (2014) that states, computationally, “Node.js clearly outperforms PHP”, and particularly excels when executed on Nginx web server. In addition to this, Chaniotis et al. (2014) also praises the notion of implementing an end-to-end JavaScript web application, describing it as “highly advisable”.

Aside from very favourable performance, Node.js also benefits from a large library of packages which can significantly increase development speeds, and with the “npm” package manager, these can be very easily implemented; which in turn has resulted in a growth in popularity.

2.5 System Security

As discussed in the introduction of this report, a key consideration of this system should be any security threats posed. There are many potential areas for security exploitation across a system, so for this report emphasis will be placed on the Open Web Application Security Project (OWASP) Top 10. Acharya et al. (2015) suggest the importance technical staff and developers considering the effects of these OWASP threats, and provided guidelines based on the OWASP Top 10, for the protection of applications accessed on mobile devices.

2.5.1 Vulnerabilities

OWASP is a community of web security specialists, that produce articles, methodologies, documentation, tools, and technologies for web application security, and every few years they release an updated list of the ten most critical security risks prevalent on the web.

Both Cross-Site Scripting and Injection can be overcome with effective implementation of input and web-page validation (Sarmah et al., 2018). Security Misconfiguration and Insufficient Logging & Monitoring are web server configuration issues and are

- | | |
|--------------------------------|--|
| 1. Injection | 6. Security Misconfiguration |
| 2. Broken Authentication | 7. Cross-Site Scripting (XSS) |
| 3. Sensitive Data Exposure | 8. Insecure Deserialization |
| 4. XML External Entities (XXE) | 9. Using Components with Known Vulnerabilities |
| 5. Broken Access Control | 10. Insufficient Logging & Monitoring |

Figure 2.4: OWASP Top 10 owasp.org (2017)

more of a concern to platform engineers, rather than to system developers. Broken Access Control and Insecure Deserialization are directly impacted by the codebase, and therefore it is important to ensure these are implemented correctly.

Query Injection

One of the most common vulnerabilities exploited by hackers is Query Injection, which is the insertion of queries into input fields with the objective of deceiving the server to execute adverse requests to the database.

Cross Site Scripting (XSS)

Cross Site Scripting (XSS) is a variation of script injection that can compromise online security and are one of the most common and most serious security flaws facing web applications. XSS allows malicious scripts to be injected into trusted applications, and occurs as a result of ineffective user input validation (Hydara et al., 2015).

Session Management and Broken Authentication

Session management describes the activities of users during their sessions of interaction with a web application; with its most common implementation being for login functionality (Vlsaggio and Blasio, 2010). Broken authentication is when users attempt to enter a system with privileges they should not have, either by altering their access permissions, or attempting to log in with an account that is not their by using brute-force attacks.

-
1. Usage of guessable session ID
 2. Absence of detection mechanism for repeated guessing trial either with brute-force or systematic methods
 3. Unable to detect repeated guessing trials while there is a mechanism in place
 4. Weak cryptography: a weakness in the cryptography algorithm or a weakness in the way a strong cryptographic algorithm is used
 5. Limitation of HTTP: the statelessness of the protocol or lack of any inherent or integrated state management mechanism
 6. Insecure session handling methods
 7. Solution misuse: the misconfiguration or improper use of basically strong solutions
 8. Weakness in the Inactive session management technique

Figure 2.5: Root Causes of Session Hijacking (Huluka and Popov, 2012)

Figure 2.5 shows the main causes of session high-jacking, when combating this form of attack, these areas should be given particular attention.

2.6 System Configuration

The following section of this report will discuss the optimal configuration of this proposed system; particularly focussing on the architecture and database design.

2.6.1 System Architecture

For this project I will be implementing a Model-View-Controller architecture. From previous experience, I have found that this approach is best practice as a segmented system allows for easier developments and alterations in the future. Fortunately Node.js is supported with an abundance of light-weight web application frameworks that simplify the implementation of an MVC architecture; with examples of these including Sails.js, Meteor.js, Express.js, and Koa.js. These will be discussed further during the development section of this report.

2.6.2 Database Design

As Node.js is a JavaScript runtime environment, although it would be possible to import drivers in order to utilise SQL database languages, it seems more sensible to implement a NoSQL database. This is because it manipulates and stores objects using

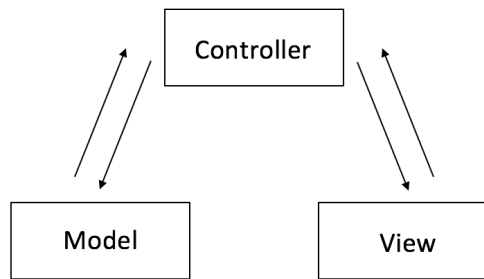


Figure 2.6: Model-View-Controller Architecture.

a JavaScript Object Notation (JSON)-like format (BSON), and would therefore simplify the process retrieving, manipulating and storing data (MongoDB, 2018).

The most commonly used NoSQL database program currently in use is a piece of open source software called MongoDB. MongoDB stores data in flexible JSON-like documents, which allows for dynamic structure change. It also facilitates ad hoc queries, indexing, and real-time aggregation, which provides quick and powerful methods to analyse and distribute data (Gu et al., 2015). Liang et al. (2017) underline the practicality of utilising a MongoDB database with a Node.js application. They highlight the flexibility of using a NoSQL database and the benefit of the ability to run asynchronous functions in the run-time environment.

Gessert et al. (2017) also explored the implementation of NoSQL database systems, and state that if complex queries have to be optimised, MongoDB is an appropriate implementation, because it facilitates expressive ad-hoc queries. In addition to this, Pokorny (2013) discusses the scalability of NoSQL databases, and their efficiency due to their semi-structured data and exclusion of nullified values.

Chapter 3

Methodology

3.1 Feasibility

Understanding whether the objective of a project is possible plays an important element of expectation management. When approaching a possible project, to prevent wasted efforts and resources, time should be dedicated to understanding its feasibility; expectations or means will then need to be adjusted accordingly in order to reach a compromise between both time & resources, and system requirements.

3.1.1 Integration and Management

Two key considerations that concern the adoption of this proposed web application are the integration with existing systems, and establishing how these systems will be maintained post production. For the purposes of simplicity, development will be completed under the assumptions that this system will exist as a stand-alone website, segmented from other university processes (with integration being considered post-development); and that ITCS (the university's IT team) will be responsible for maintenance.

3.1.2 Scheduling

Before commencing any type of project it is important to properly plan for the effective use of time, not only to improve efficiency and ensure goal delivery, but also to guarantee project feasibility. This project has been broken down into tasks (table 3.1) and milestones (table 3.2). These have then been allocated deadline and have been plotted on a Gantt chart visible in figure 3.1.

Necessary Tasks

One of the most fundamental concepts when managing a project is the use of a “divide and conquer” approach; which allows for better time management and affords a greater clarity of progress. One of the simplest ways of breaking down an overarching problem is to segment it into core independent tasks, which has been done in table 3.1.

No.	Objective of Task	Duration (weeks)
1	Hold stakeholder meeting to understand functional requirements.	1
2	Research functional system requirements.	1
3	Research and install necessary software and packages.	1
4	Design and build website UI.	4
5	Database design and implementation.	4
6	Investigate and improve system security.	1
7	Perform integration testing.	1
8	Accessibility checks.	1
9	Amendments to accessibility precautions.	2
10	User feedback session.	1
11	Complete project write up.	6
12	Proof read, adjust, and bind document.	1
13	Prepare presentation material.	1

Table 3.1: Project Tasks

Project Milestones

Milestones are benchmarks that can be used to establish progress and usually demonstrate the end of a key stage of the project. Three milestones have been chosen for this project and can be seen in table 3.2.

No.	Milestone Benchmark	Target (week)
1	Comprehensively document all requirements.	2
2	Finish testing, having produced an initial functioning deliverable.	8
3	Completed the web application evaluation process and implemented necessary changes.	14
4	Conclusion of the report writing and ready for the presentation of work.	11

Table 3.2: Project Milestones

Gantt Chart

Another important principle of project management is understanding task dependencies and their impact on project completion. A core concept of this is the “critical path”, which is the most time-effective combination of activities for scheduling. A simple method of observing the critical path is the implementation of a Gantt chart, which can be seen in figure 3.1.

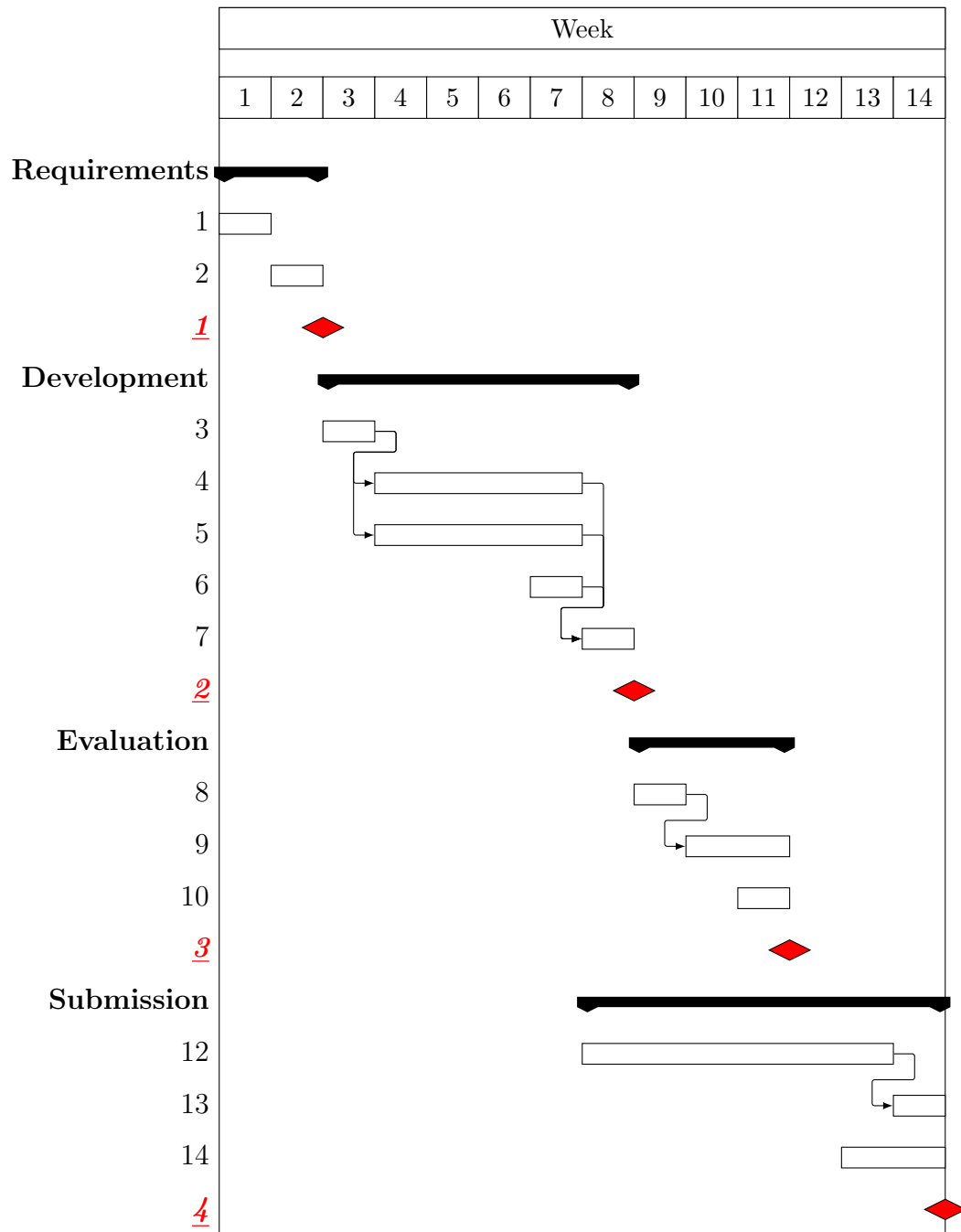


Figure 3.1: Gantt Chart of Tasks

3.2 Requirements Elicitation

Requirement elicitation is the process of understanding and documenting the needs of the stakeholders of a system (Kjeøy and Stalheim, 2007). It is carried out in order to understand needs and expectations, and it is one of the most important stages in the development cycle, determining how successful the system will be. For this web application, requirements were gathered using two key methods. The first of these focussed on establishing the functional requirements, by speaking directly to a member of the University of East Anglia Research Ethics Committee in a one-to-one meeting. The second method was researching online materials, such as viewing the different ethics application processes across the University and any accessibility requirements. The MoSCoW analysis is an effective method of prioritising requirements, categorising them into “must”, “should”, “could”, and “won’t”. The main requirements have been included using a MoSCoW template and can be seen in tables 3.4 and 3.3.

3.2.1 Technical Requirements

Technical requirements are any necessary benchmarks that must be fulfilled from an engineering and development perspective.

MoSCoW Table

Importance	Requirement
Must	<ul style="list-style-type: none"> • Satisfy WCAG 2 guidelines • Utilise data encryption
Should	<ul style="list-style-type: none"> • Follow an Model-View-Controller Architecture • Implement reusable code
Could	<ul style="list-style-type: none"> • Import external packages • Make use of a NoSQL database
Won't	<ul style="list-style-type: none"> • Connect to any UEA servers or use any of their APIs

Table 3.3: MoSCoW Analysis - Technical Requirements

3.2.2 Functional Requirements

Functional requirements are those that are necessary for the system to successfully fulfil the needs of the end-users.

MoSCoW Table

Importance	Requirement
Must	<ul style="list-style-type: none"> • Have an intuitive user interface. • Be accessible for all students. • Keep user data safe. • Allow for administrators to produce and update application forms. • Enable applicants to submit and amend applications. • Provide the ability for administrators to review and then either approve or deny applications, with the opportunity for adding feedback to applicants. • Generate a dashboard to give an overview of current applications.
Should	<ul style="list-style-type: none"> • Be optimised for mobile devices. • Log the status changes of applications. • Determine the priority of applications and give feedback to administrators. • Incorporate partial data exploration.
Could	<ul style="list-style-type: none"> • Be optimised for mobile devices. • Automate feedback and notifications through email. • Provide in depth data exploration.
Won't	<ul style="list-style-type: none"> • Integrate with any existing UEA systems.

Table 3.4: MoSCoW Analysis - Functional Requirements

For the “won’t” section I have included that this system will not integrate with any existing systems within the University of East Anglia. Although this may incorporated in later iterations of this system; the security and configuration complications combined with the time limitations make this an impractical endeavour at this stage of the development process.

3.3 Use Case Descriptions

Use case descriptions are a helpful tool for understanding the required actions of a user in order for them to achieve an outcome, and how deviations from this can be mediated (Liang, 2003). Four of these descriptions have been included in tables 3.5, 3.6, 3.7, 3.8; using a framework adapted from a paper by Kungurtsev et al. (2017).

3.3.1 Use Case 1

Use Case Name	User Makes an Ethics Application
Scenario	A user has made their way on to the webpage and are now wanting to submit an application.
Trigger Event	A user wishes to apply for ethics approval.
Brief Description	An individual will be partaking in an activity that may require ethics approval, and so they wish to apply for permission from their REC Chair.
Actors	Applicant.
Precondition	They have accessed the system successfully.
Postcondition	An application is submitted to a Research Ethics Committee representative.
Flow of Activities	1.0. User clicks on the ‘New Ethics Form’ 2.0. User is given the choice of individual or module forms 3.0. User selects a form 4.0. User fills in the form appropriately 5.0. User adds relevant files 6.0. User is redirected to dashboard
Expectation Condition	4.1. User misses compulsory field 5.1. User is prompted to correct mistakes

Table 3.5: Use Case Description 1 - A user makes an ethics application.

3.3.2 Use Case 2

Use Case Name	Administrator Creates a New Form
Scenario	An administrator has logged into the application, and is wanting to create a new ethics form.
Trigger Event	A specific form is necessary for users, so an administrator wishes to create a new form.
Brief Description	A new type of ethics application is necessary for particular circumstances, an administrator is required to generate a new one.
Actors	System Administrator.
Precondition	The administrator is logged into the web application and on the dashboard.
Postcondition	A form is created by a system administrator and made available to users.
Flow of Activities	<ol style="list-style-type: none"> 1.0. User selects ‘Admin Panel’ on menu bar 2.0. User clicks on ‘Create a Form’ 3.0. User gives the form a title and subtitle 4.0. User adds inputs to the form 5.0. User submits form to the database 6.0. User is redirected to the form page to show it has been implemented
Expectation Condition	<ol style="list-style-type: none"> 5.1. User adds inputs in the wrong order 6.1. User turns on edit controls 7.1. User can use these controls to either move or delete the affected elements.

Table 3.6: Use Case Description 2 - An administrator creates a new form.

3.3.3 Use Case 3

Use Case Name	Administrator Edits an Existing Form
Scenario	Either the requirements or details have changed regarding an existing application.
Trigger Event	A form requires either new or different fields.
Brief Description	The circumstances are such that a form requires altering. An administrator must then login and alter an existing form.
Actors	System Administrator.
Precondition	The admin is logged into the system, the user is on their dashboard, and a form exists that requires altering.
Postcondition	A new version of an existing form is produced, and the previous version is replaced on the display page.
Flow of Activities	<ol style="list-style-type: none"> 1.0. User selects 'Admin Panel' on menu bar 2.0. User clicks on 'Update a Form' 3.0. The user picks from a list which of the forms they wish to update 4.0. Users can then use the controls to move, delete or add components. 5.0. Once complete, the user then submits the form. 6.0. A new version of the form is created, and replaces the form available to applicants.
Expectation Condition	<ol style="list-style-type: none"> 4.1. A user accidentally deletes a necessary field. 5.1. The user can recreate the field and move it to its required position.

Table 3.7: Use Case Description 3 - Administrator edits an existing form.

3.3.4 Use Case 4

Use Case Name	An Administrator Leaves a Comment
Scenario	An administrator wishes to leave feedback regarding an application.
Trigger Event	An administrator wishes to provide feedback regarding why ethics have not been approved.
Brief Description	An administrator is reviewing an ethics application, finds an issue, and they now wish to inform the applicant what must be done to correct it.
Actors	System Administrator.
Precondition	An application has been made, and the administrator is logged in.
Postcondition	A comment is made under an application.
Flow of Activities	<ol style="list-style-type: none"> 1.0. A user navigates to their dashboard. 2.0. They use the search function to find the form they wish to comment on. 3.0. They select the ‘view’ functionality. 4.0. The user can then type their feedback into the comment box and submit it. 5.0. A time stamp is provided detailing when the comment was made to confirm it was submitted properly.
Expectation Condition	<ol style="list-style-type: none"> 5.1. The feedback they give is incorrect. 6.1. The user rewrites their comment. 7.1. The comment is resubmitted.

Table 3.8: Use Case Description 4 - An administrator leaves a comment.

3.4 Risk Analysis

3.4.1 Risks Factors

Before analysing the risks pertinent to the development and implementation of this web application, it was necessary to consider the what these risks are. Table 3.2 details a few of those considered in this report.

No.	Risk	Causation	Mitigation Strategies
1	Vital functional requirements are not appropriately satisfied.	Deviation from the plan or omission of essential functionality.	Requirement implementation should be evaluated regularly during the development process.
2	The system does not meet the 2.1 WCAG guidelines.	Oversight of implementation of accessibility features.	Perform a comprehensive appraisal of all webpages during system evaluation.
3	The projected system delivery date is delayed beyond the estimated deadline.	Scope creep or technical issues, leading to the postponement of feature implementation and the deployment of the application.	Strict adherence to essential requirements and regular progress evaluation.
4	Some functionality is missing from the final deliverable.	Errors in the requirement elicitation phase, by either missing or misinterpreting user needs.	Thorough user evaluation and the quick implementation of missing features.
5	Issues hosting the web application.	Incorrect or insecure server configuration.	Ensure proper research is undertaken to ensure the safe hosting of the web application.
6	The website is targeted by malicious persons.	The website has a vulnerability that may lead to data exposure.	Ensure maximum care is taken to protect users and encrypt high-risk data.
7	Users encounter system errors post-release.	Processing issues during application usage, such as difficulty handling inputs.	Effective testing and input validation to prevent run-time errors.
8	System configuration problems / failures lead to servers crashing and data loss.	Power surges, power cuts, attacks, or runtime errors.	Regularly back up database information, store configuration preferences, log errors and issues, and use a repository for development.

Table 3.9: Major Risk Factors

3.4.2 Probability and Impact Matrix

The probability / impact matrix is a framework which can be used to analyse the risk of an event. It works by assigning values to both its likelihood and the potential adverse effects it would have on an outcome, then these variables are plotted against each other on a grid in order to establish the associated level of risk. For the examples provided in the previous section, I assigned values ranging from one to ten, for both probability and impact factor and were subsequently plotted on a matrix. Both the table (table 3.10), and the plot (figure 3.2) can be seen below.

No.	1	2	3	4	5	6	7	8
Probability	1	4	7	7	4	2	7	3
Impact	10	8	7	6	10	7	4	9

Table 3.10: Probability / Impact Table

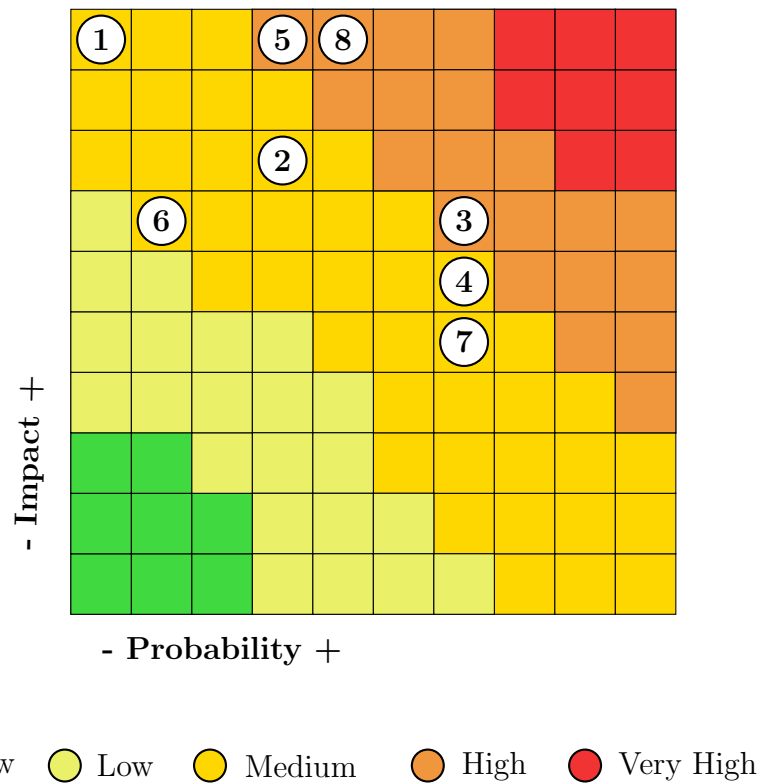


Figure 3.2: Probability and Impact Matrix

From the matrix in figure 3.2, you can see that the most problematic areas are risks 3 (delay), 5 (hosting) and 8 (system configuration); therefore it is important to adhere to the planned schedule and make contingencies for run-time faults and security.

3.5 Legal, Social and Ethical Considerations

3.5.1 Accessibility

Accessibility is a very significant factor influencing the development of this system. A substantial number of students within the university suffer from some form of disability and ensuring that this service can be used by as many individuals as possible, regardless of their ability is vital.

There are three documents in particular that will be considered when approaching the development of this system. The first of these is the Equality Act 2010 (gov.uk, 2010), which will be used to ensure that this website adheres to what is legally required. The second document that will be considered is the UK governments guidelines on accessibility (Government Digital Service, 2018), in order to reduce any potential oversights regarding disabled users. The final of these sources is the WCAG 2.1 guidelines (Kirkpatrick et al., 2018), which will be used as a framework to ensure that the service is compliant, to check the technical implementation of accessibility features, enhancing the user experience of individuals with disabilities.

3.5.2 Data Protection

Ensuring that this system complies with any legal requirements is of the utmost importance, so consequently, there are three significant pieces of legislation that are pertinent to the development of this project. While the U.K. remains in the European Union, demands of the EU directives 95/46/EC (Directive 95/46/EC, 1995) and 2002/58/EC (Directive 2002/58/EC, 2002), must both be adhered to, as well as the General Data Privacy Regulation (Regulation (EU) 2016/679, 2016). In addition to EU directives and regulations, the domestic Data Protection Act (Data Protection Act 1998, 1998) must also be respected.

As most of these laws centre around the storage and monitoring of data, as well as ensuring privacy and security on a technical level, it is also important to ensure that anyone managing this system is also aware of their responsibility to protect data.

3.6 Technical Considerations

3.6.1 External Libraries

External libraries can be fantastic development tools, significantly decreasing build times and simplifying very complex functions, however, they should be used with an element of caution. When implementing external features they can introduce uncertainty and jeopardise security.

Although this cloud surrounding the implementation of external features does exist, it is possible to mitigate against it. One way in which this can be done is to regularly keep track of online vulnerability databases, which flag up issues with software, document the versions effected, and suggest ways to prevent adverse effects. Another preventative measure that can also be taken is the regular patching of external packages and libraries when updated versions are released onto the market.

In addition to preventing the occurrence of potential vulnerabilities and threats, it is important to ensure that libraries are maintainable and well documented. Due to the fast-paced nature of development, companies which produce external packages and libraries occasionally cease to exist and consequently support can follow suit. That's why it is important to ensure that any external code used is either understandable to those managing it, or alternatively, well documented for maintainability and future development later in the product life cycle.

A final consideration when implementing external libraries is the issue of licensing and fair use. There are many types of licences, ranging from “trade secret” (all rights retained) through to “public domain” (all rights relinquished) available; so it is important to understand what can and cannot be used, and for what purposes. As this is a non-commercial piece of software, both free and permissive licences will be used; these will exclusively be either GNU (General Public License) or MIT (Massachusetts Institute of Technology).

3.6.2 System Architecture

As discussed in the literature review, an MVC architecture will be implemented for the web application, as this will allow for flexibility in system scalability and also will simplify an additional features implemented in the future. The decision logic and database connections will occur within the model, data will be presented by the view layer of the system and information will be parsed between the two via the controller.

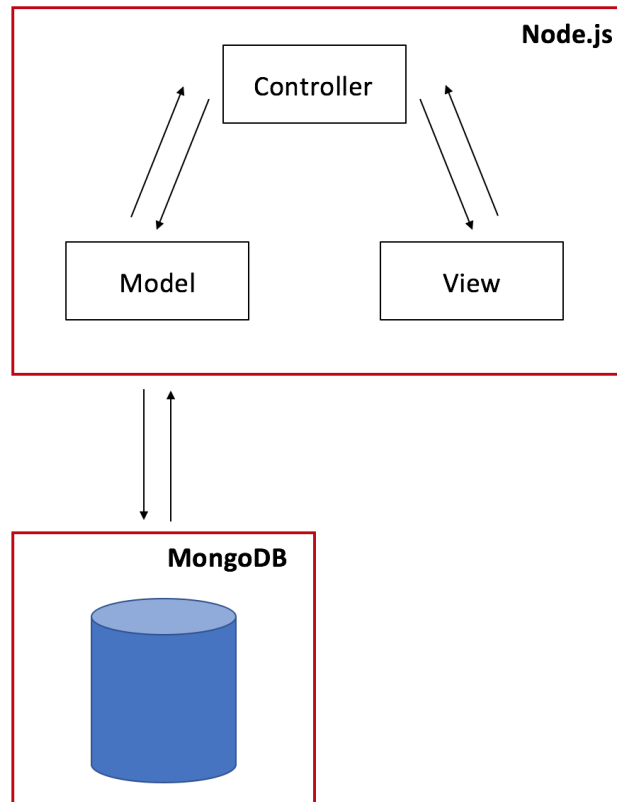


Figure 3.3: Model-View-Controller

In addition to the web application MVC architecture, the database will be a standalone element of the architecture and send / receive data via the model. Ideally complete segmentation of the database from the Node.js server will be achieved, however that will be dependent on schedule.

Chapter 4

Development of System

4.1 Database Design

4.1.1 Languages and Tools

For this system, due to the majority of development being completed using JavaScript and JSON, it made practical sense to implement a JavaScript-based database, and therefore MongoDB was chosen. When developing in Node.js with a MongoDB database, the standard ODM (object data modelling) is Mongoose; so with a substantial amount of documentation; that was chosen as the tool for schema definition. Mongoose also supports built-in validation features and schema definition that allows for greater flexibility and additional security procedures to simplify and reduce development time.

4.1.2 Database Schemas

In database design, schemas provide the blueprints for how data can be stored within the database and are a vitally important area for consideration during the implementation stage. Three database schema models have been utilised by this web application; one is dedicated to the storage user data (both personal information and login details) with any particularly sensitive data being fully encrypted using an asymmetrical encryption technique. The second is designed to store application data, recording both their answers and the feedback given by staff. The final schema stores the layout and version numbers of the various forms. An example of the schema used to generate a user object is displayed in listing 4.1.

```

1  const UserSchema = mongoose.Schema({
2    username: {
3      type: String,
4      unique: true,
5      validate: {
6        validator: function (v) {
7          return /^[a-z]{3}[0-9]{2}[[a-z]{3}?$/ .test(v);
8        },
9        message: props => `${props.value} is not a valid format`
10     },
11     required: [true, 'Username Required']
12   },
13   password: String,
14   email: String,
15   supervisor: String,
16   admin: Boolean,
17   fname: String,
18   lname: String,
19   known_as: String,
20   school: String,
21   type_of_user: String,
22   memorable_word: String,
23   memorable_word_answer: String,
24   visited: Boolean
25 });
26 const User = module.exports = mongoose.model('User', UserSchema);

```

Listing 4.1: Database Schema

4.1.3 Data Types

Data types are implemented to provide structure in the schema. For this project, some data types have been used in less conventional ways. Examples of these include form attachments being stored as arrays of objects, with a predefined structure, and some dates being stored as Number types in order to store their unix timestamp, facilitating the use of the “moment” natural language display JavaScript package.

4.1.4 Attributes

To ensure the correct management of records, the choice of attributes and keys was an important factor to consider. The attributes used for this application are shown in table 4.1, with the primary keys shown in bold and the foreign keys with an asterisk.

Attributes		
User	Form	Form Layout
username*	first_name	form_layout_no*
password	last_name	form_version_no*
email	username*	form_type
supervisor	project_title	questions
admin	type_of_applicant	name
fname	time	template
lname	last_update	raw_html
known_as	status	creator*
school	code	created
type_of_user	seq	
memorable_word	school	
memorable_word_answer	form_layout_no*	
visited	form_version_no*	
	feedback	
	feedback_time	
	attachments	

Table 4.1: Database Schemas and Their Attributes

4.1.5 Input Validation

The majority of input validation is handled in the front-end of the web application, however, contingencies had to be made for the possibility of bypassing the processes put in place. Therefore validation is also performed by Mongoose during database insertion. An example of this can be seen in listing 4.1. In this instance, the validation is carried out using the model, and is checking the format of usernames by using a ‘regex’ function that ensures they follow the XXXAAXXX format; where X represents a lower case letter and A is a numeric value. Inputs that parse Form values are also serialised when being handled, in order to combat cross site scripting. Due to the construction of queries in BSON, validation for the prevention of SQL injection is unnecessary.

4.2 Implementation of Node.js

The following section of this report explains how Node.js was used in the development of this system. Four categories have been identified for particular focus, these are; packet management, application frameworks, routing, and template engines.

4.2.1 Package Management

One of the huge benefits of developing with Node.js is the wide variety of libraries available. As a result of this it is important to effectively manage the installation of packages and be cautious when handling dependencies. Fortunately, Node.js utilises the JavaScript package management tool “npm”, which allows for simple and effective library management. All installations were managed using this; and all libraries (with version numbers) are listed below.

Package	Version	Package	Version
bcryptjs	2.4.3	express-validator	5.3.0
body-parser	1.18.3	fs	0.0.1-security
bootstrap	3.3.7	lodash	4.17.10
connect-busboy	0.0.2	moment	2.22.2
connect-flash	0.1.1	mongodb	3.1.1
cookie-parser	1.4.3	mongoose	5.2.5
cookie-session	2.0.0-beta.3	multer	1.3.1
debug	2.6.9	nodemailer	4.6.8
ejs	2.6.1	nodemon	1.18.4
express	4.16.3	passport	0.4.0
express-fileupload	0.4.0	path	0.12.7
express-session	1.15.6	shelljs	0.8.2

Table 4.2: Packages Utilised

All packages are available for use under MIT licensing.

4.2.2 Frameworks

A framework is a piece of software that provides generic functionality, which can then be customised, in order to provide application-specific functionality. The benefit of implementing a framework is that as a significant amount of functionality has already been pre-coded, reducing development time. There are many frameworks available for Node.js development, so it was important to choose the most appropriate one. I focussed my attention on the most popular (and consequently the most supported) options; these are: Sails.js, Meteor.js, Express.js, and Koa.js.

Of these frameworks; Express is by far the most popular due to its simplicity and suitability for the development of commercial web applications, and has been referred to as the “de facto standard Node.js web application framework” (Serby, 2012). This popularity has stemmed primarily from its routing, middleware, and template engine features that provides a great deal of useful functionality without compromising efficiency. For this project, development time took precedence over features present in the other frameworks (such as data synchronisation and database object-relation mapping); and therefore Express was chosen as the preferred framework.

4.2.3 Routing

Once Express had been established as the choice of web-framework for the application my attention was turned to configuring the routing of the website. Within the domain of Node.js web development, routing is essentially how an application handles and responds to client requests, and there are several approaches that can be taken.

I began by defining route paths (the URI path and its associated JavaScript controller) within the main application file. This directs the the user from their desired web address to the relevant controller. Once the appropriate controller is found, based on the request (and occasionally the access permissions of the user), either the correct actions are performed or the suitable content is displayed, depending on the type of request (the two possible requests are either POST or GET methods).

```
var login = require('./routes/login');  
app.use('/login', login);
```

Within these requests, in addition to defining custom logic and JavaScript functions to be executed, there are various predefined actions (responses) that can then be utilised. An example of this is:

```
res.render('name_of_view_required');
```

which is a predefined function which renders the specified template to the user. Through this routing process it is also possible to parse data into the template in order to manipulate content or permission dependent features.

4.2.4 Template Engines

When developing in Node.js for the web, there are several options for how HTML templates can be delivered to the user, with these various methods are known as “template engines” (Ivanovs, 2018), with examples including Mustache, Jade, doT.js, EJS, Handlebars, underscore.js, and Pug. For this particular project, EJS (Embedded JavaScript) templating was implemented. The reason for this was as a result of its fast and efficient code execution, the ability to perform quick and effective debugging, as well as the abundance of documentation and resources that are available, which allows for quicker development time, and a greater level of support to future developers.

EJS is also particularly useful for content delivery, as it allows for use of JavaScript to manipulate the HTML based on the information provided during the routing stage; with compilation occurring before the delivery of the template to users. Another benefit of EJS is the ability to take advantage of the its “partials” functionality. This allows for effective code reuse, with reduces both development time and required memory. Partials will be discussed further in the styling and design section of this chapter.

4.3 Styling and Design

This section of the report discusses the styling decisions that were made throughout the development of this website; documenting the considerations made toward design consistency, content implementation, colour schemes and web interoperability.

4.3.1 Tools for Consistency

Bootstrap

When designing the interface for a web application, in order to prevent confusion during the user experience, it is important to apply consistent styling throughout, so for this reason Bootstrap v4 was implemented. In previous versions, Bootstrap had just provided Cascading Style Sheet (CSS) files, with predefined styling in order to facilitate faster development; however in more recent versions JavaScript files have also been included within the framework, allowing for content manipulation. In addition to their grid layout system, Bootstrap also provides an extensive library of components and features that can be previewed on their website; and implemented effectively with help from their online documentation.

For this project, eleven Bootstrap components were implemented across the system; these are alerts, badges, buttons, cards, collapsible divs, forms, jumbotrons, modals, navbars, popovers, and tooltips. An example of the styling options available for the buttons are shown in figure 4.1.



Figure 4.1: Bootstrap Buttons - Example

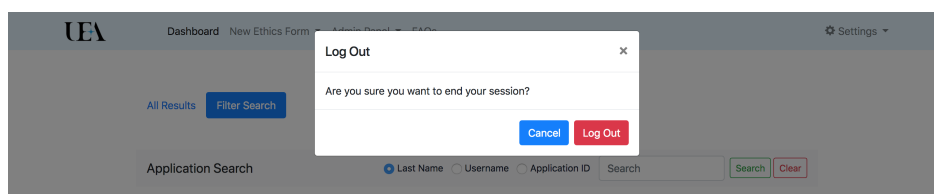


Figure 4.2: Bootstrap Modal - Implementation of modal to confirm logout.

EJS Partials

Partials are Embedded JavaScript (EJS) files, that can be imported into a larger EJS templates to allow for code-base repeatability and re-usability. In order to ensure that the content which appears on every web-page is consistent I implemented “partials” for some elements. Not only does this improve the consistency of design; it also saves time rewriting exiting code and allows for changes to be propagated easily throughout.

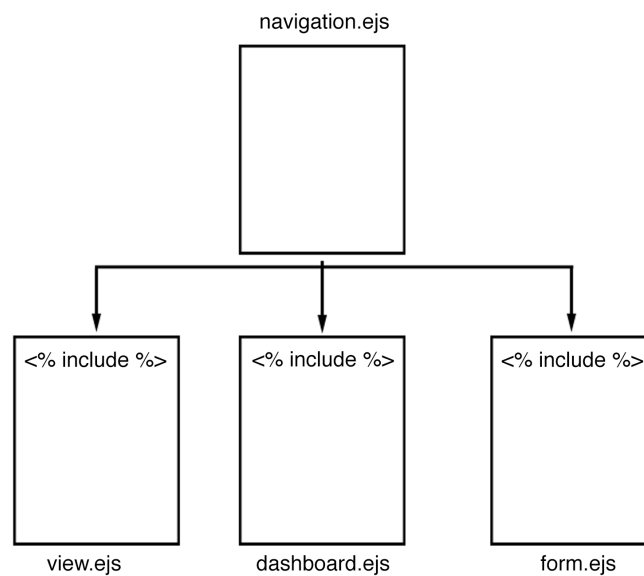


Figure 4.3: A simple diagram detailing how EJS partials work - Adding the file path after “include” allows for the template to be imported.

Three partials are used in the majority of pages for this website: the head tag (containing all the external files and the page title), the navigation bar, and the logout modal; as changes to any of these files would require permeation throughout the system. In addition to these, some administrative functionality was also included as partials.

Customisation

For some of the system specific content and features, the styling provided by the Bootstrap framework was not always sufficient, so therefore a custom cascading style sheet (CSS) file was also included within the web application. This was placed after the Bootstrap import in the code to allow for either the overriding or alteration of existing styling to suit the needs of the system.

4.3.2 Design Features

Colour Schemes

Colour is a great way to effectively convey information to a user, in order to remove the amount of cognition required to recognise and interpret what is being explained to them. This is why a traffic-light based system was implemented for determining the status of their applications (figure 4.4). Nielsen suggested the importance of recognition rather than recall, as many users are familiar with the traffic-light system (green: good, yellow: waiting, and red: bad), the justification for this additional level of feedback is that it would be an extra feature that could help to reduce the user’s cognitive load.

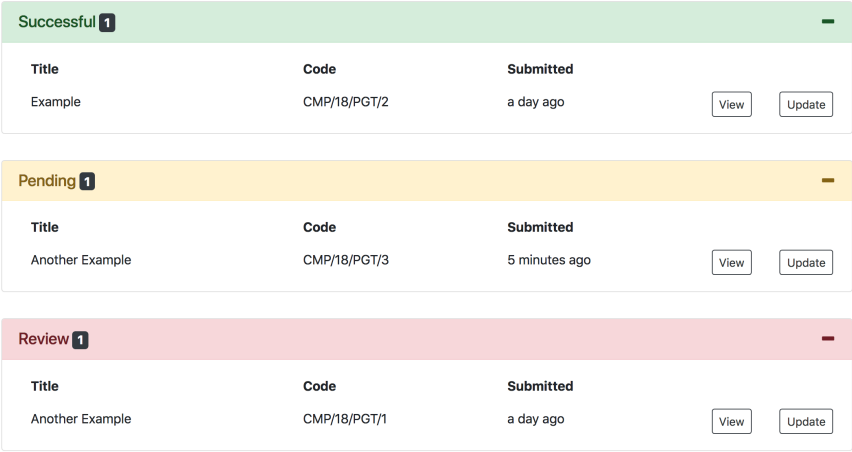


Figure 4.4: Traffic Light Colours Implemented for Context

Relating to accessibility, when considering colour schemes, another important factor to consider is users who may have various forms of colour vision deficiency (colour blindness). One of the most effective ways to improve the experience for individuals that suffer from colour blindness is to ensure that the contrast between foreground background objects is at a significant enough level. There is an effective tool, provided by web designer Jonathan Snook, that is available online which provides information about the contrast of colours and whether they comply with WCAG standards. The results for colours used in this project can be seen in table 4.3.

Background Colour Hex	Foreground Colour Hex	Compliance
E3F2FD	000000	WCAG 2 AA Compliant WCAG 2 AA Compliant (18pt+) WCAG 2 AAA Compliant WCAG 2 AAA Compliant (18pt+)
E9ECEF	212529	WCAG 2 AA Compliant WCAG 2 AA Compliant (18pt+) WCAG 2 AAA Compliant WCAG 2 AAA Compliant (18pt+)
C3E6CB	155724	WCAG 2 AA Compliant WCAG 2 AA Compliant (18pt+) WCAG 2 AAA Compliant (18pt+)
FFF3CD	856404	WCAG 2 AA Compliant WCAG 2 AA Compliant (18pt+) WCAG 2 AAA Compliant (18pt+)
F8D7DA	721C24	WCAG 2 AA Compliant WCAG 2 AA Compliant (18pt+) WCAG 2 AAA Compliant WCAG 2 AAA Compliant (18pt+)

Table 4.3: WCAG Colour Contrast Compliance for Colour Blind Users

Content Display

Following on from the discussion of Krug (2014), deciding how space should be utilised is very important, especially when a system may be accessed from mobile devices such as smartphones and tablets. There were two main features, made available through the implementation of Bootstrap, that were utilised to allow for the management of information being displayed to the user, these were collapsible content “accordions” and navigation bars.

In support of this, Borkin et al. (2015) note the importance of visualizations, stating that items which are memorable “at-a-glance” are those that can be quickly retrieved from memory; therefore choosing which data to display is very important.

The first of these was the accordion, which can be seen on the user dashboard page. Accordions can be used to collapse (hide) information that may not be immediately required by a user, thus giving a less cluttered appearance while allowing for data to

be accessed more readily when required. They are particularly useful when dealing with categorised information that can be easily partitioned into distinct sections, and where some of this data may not always be needed. This allows for information to be viewed without having to navigate between pages and also maximises the use of space. Accordions are also already in use within other web systems at UEA too (figure 4.5).

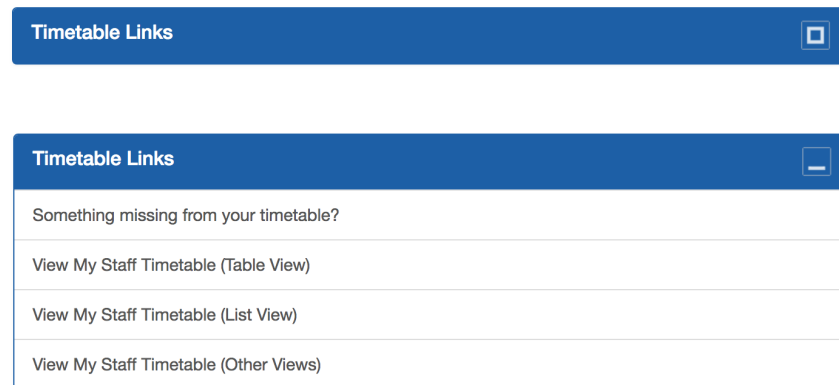


Figure 4.5: Accordion in Usage on an Existing UEA System (evision) - Closed (top) and Open (bottom).

The other feature used in this web application to improve the use of space is the implementation of tabs. Tabs are another method that can be used to hide content that may be useful to the user, but not necessarily immediately required. For this system, it has been used on the administrator dashboard in order to allow administrators to make use of the application search functionality without navigating away from the main dashboard, and to easily toggle between views; an example of this can be seen in in figure 4.6.

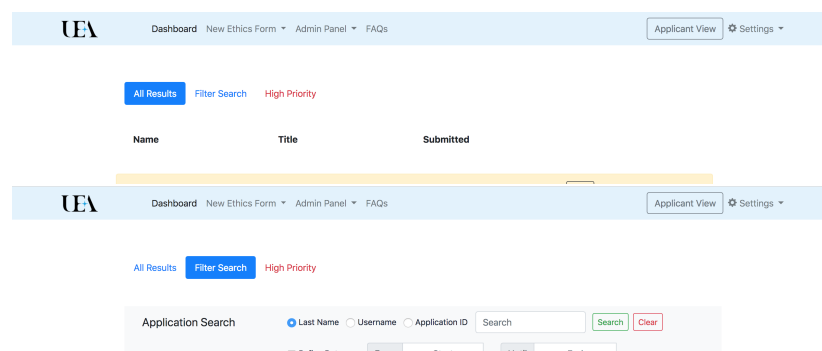


Figure 4.6: Tabs Allow Users to Toggle Content - All Results (top) and Search Bar (bottom).

4.3.3 Web Interoperability

The term “web interoperability” was first used in the now obsolete Web Interoperability Pledge (WIP) and refers to the production of web pages that can be rendered and viewed correctly across multiple web browsers, operating systems, and devices. Although greater interoperability generally results in a more accessible system, it is important to consider user experience vs accessibility; and not jeopardise usability in order to satisfy obsolete or impractical restrictions. Over the past decade, the consideration of how content is displayed on mobile devices has become a vital element of web development, with over half (51.2%) of content being accessed on mobile devices in April 2018 (statista, 2018). Therefore, during the development of this web application, much thought was given to ensuring that the application worked on mobile devices as well as desktop and laptops.

Content Delivery

Many of the mainstream mobile web browsers use WebKit as the underlying browser engine, and as a result JavaScript is available to the majority of devices, and due to this, a large amount of JavaScript (particularly jQuery) has been utilised for front end content delivery. The browsers supporting jQuery can be seen in figure 4.4.

Desktop	
Chrome	v69 and v68 (current and previous)
Edge	v44 and v43 (current and previous)
Firefox	v62, v61 and ESR (current, previous and Extended Support Release)
Internet Explorer	v9+
Safari	v6 and v5 (current and previous)
Opera	v55 (current)
Mobile	
Android 4.0+	Stock browser
Safari	iOS 7+

Table 4.4: Actively Tested jQuery Browser Support

Responsive Layout

In the early days of smartphones websites often had a separate mobile version with an altered design tailored toward rendering content on limited displays; however, with the onset of improved means of manipulating the HTML DOM and more sophisticated styling techniques responsive web design has become the more dominant practice.

As discussed earlier in this report, the use of Bootstrap has allowed for the easy implementation of a responsive layout, with its integrated “grid system” which utilises a series of containers, rows, and columns to layout and align content. Within the Cascading Style Sheet (CSS) there are affordances made for alterations in screen dimensions, that alter the way content is delivered based on how much space is available on the page.

In order to determine when an object should adjust layout for responsive design, Bootstrap uses “breakpoints”. Breakpoints are pre-defined in the CSS files, media queries are then used to determine screen dimensions. When these dimensions satisfy the conditions of the breakpoint, the styling is amended accordingly and the layout of the content is adjusted. These breakpoints can be manually altered, however the default settings can be seen in table 4.5.

Breakpoints		
Size Name	Dimensions (width)	Typical Device
Extra Small	< 576px	Portrait phones (default)
Small	576px ≤	Landscape phones
Medium	768px ≤	Tablets
Large	992px ≤	Desktop computers
Extra Large	1200px <	Large Desktop computers

Table 4.5: Bootstrap Responsive Breakpoints

Each of the sizes overrides the previously defined breakpoint, allowing for the alteration of CSS styling. It is possible to specify in the HTML class tags when specific objects should be adjusted.

One of the most apparent implementations of this responsive design layout is the navigation bar. When the screen dimensions fall below the large breakpoint (992px in width), the menu changes from a conventional bar menu, to a mobile navigation menu with the distinctive “hamburger” menu icon.

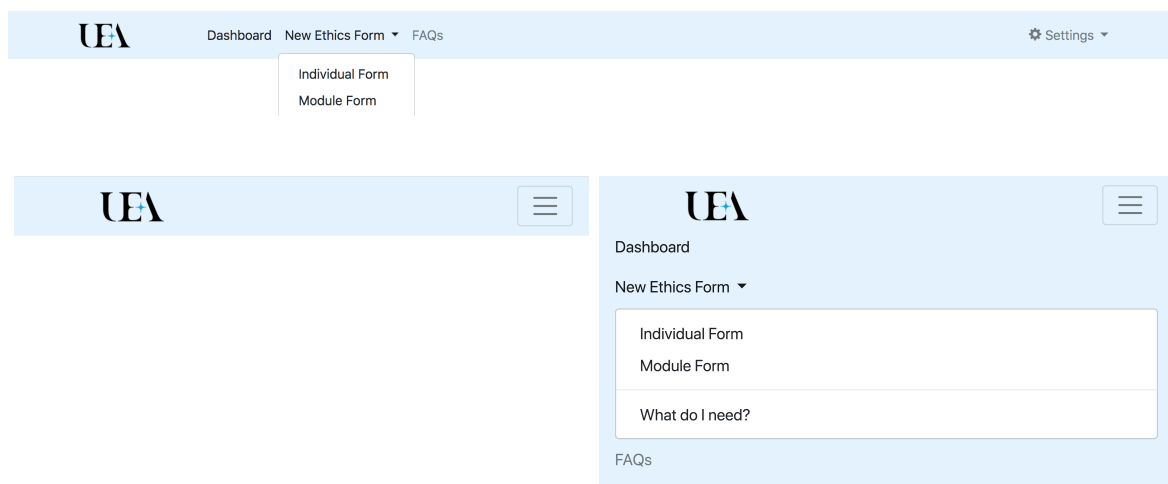


Figure 4.7: Responsive Web Design - The display of the menu is changed based on screen dimension - Desktop (top) and Mobile (bottom).

4.4 Accessibility and Usability

4.4.1 Measures Taken

For this system, precautions have been taken in order to achieve high levels of accessibility and usability. The first of these is the effective implementation of form labelling. For users with impaired vision (particularly those using screen readers), correct labelling of forms is a necessity for them to be able to use the functionality available. Therefore extra care has been taken to ensure all fields have been labelled correctly and have been given adequately descriptive names.

Another precaution taken to assist with the accessibility of the system is the implementation of Alt tags in HTML mark-up for when images are inserted. Although very few images are used in this website, ensuring that users are able to understand what is in front of them is vital and so therefore a great deal of care has been taken to insert these tags where necessary.

A third area that was considered during the development of this system is the use of space and the avoidance of clutter. For individuals who have difficulty processing large quantities of information, having too much going on when viewing a webpage can lead to confusion; therefore efforts were made to reduce how much information users are presented with, whilst avoiding too much unnecessary complexity. This was achieved through the use of toggled content and the implementation of navigation bars, these were however given consideration before implementation as it was important that they don't impact usability Sutcliffe and Hart (2017).

The final measure taken to ensure suitable levels of accessibility is the choice of font throughout the system. In this latest edition of Bootstrap (version 4), the conventional web default fonts of Helvetica Neue, Helvetica, and Arial have all be dropped in favour of a “native font stack” for optimum text rendering across multiple devices and operating systems.

Fonts	
Web Browser	Font
Safari for OS X and iOS (San Francisco)	-apple-system
Chrome < 56 for OS X (San Francisco)	BlinkMacSystemFont
Windows	“Segoe UI”
Android	“Roboto”
Basic Web Fall-back	“Helvetica Neue”, Arial, sans-serif,
Emoji Fonts	”Apple Color Emoji”, ”Segoe UI Emoji”, ”Segoe UI Symbol”

Table 4.6: Font derivation in Bootstrap

As discussed in the earlier section of this report concerning colour choices, the contrasts of fonts against backgrounds have been checked to ensure that they are WCAG compliant for visually impaired users. In any cases where AAA ratings are achieved with font sizes at 18pt or above, it has been ensured that the content is presented at this size.

4.4.2 User Support

Ensuring that users are able to understand and navigate their way around a system is a vital part of the UX development process. The following section will discuss how this was considered during this project.

Interaction Feedback

Providing feedback to the user is a vital part of the user experience and therefore efforts have been made to ensure that there is sufficient user support, and opportunity for interaction. In cases where users are required to perform an action, and the user behaves in an unexpected way, prompts are provided in an attempt to correct the issue that they have encountered, an example of this can be seen in figure 4.8.

The figure displays two screenshots of a login interface for 'UEA'. Each screenshot shows a light blue header with the 'UEA' logo on the left, a 'username' input field, a 'password' input field, and a 'Log In' button on the right. Below the input fields is a link for 'Forgotten Password?'. In the top screenshot, a red error box appears below the login form, containing the text 'Error' and 'The username you entered does not exist.' In the bottom screenshot, a similar red error box appears, containing the text 'Error' and 'The password you entered is incorrect.'

Figure 4.8: Feedback after entering a non-existent username (top) and an incorrect password (bottom).

Documentation for Users

Online support and documentation is a good way of providing solutions to issues encountered by users. Accommodations have been made to allow for the implementation of a frequently asked questions (FAQ) page. This allows users to seek answers to their questions without having to reach out for technical support. Unfortunately, due to time restraints caused by delays during development, this FAQ page was only partially implemented; however these example cases provide an overview of how this will be implemented, and the idea will be discussed further in the evaluation of this report.

4.5 Security Features

The final section of this chapter of the report will discuss the considerations made toward information protection and data encryption, as well as contingencies made for malicious behaviour. Accounting for security vulnerabilities was a key element of this project, in addition to encrypting user data, ensuring it is kept away from malicious persons is also a key responsibility of any system developer. Vulnerabilities were investigated by examining five key areas, which were: ensuring the appropriate processing of user passwords, checking vulnerability databases for the known issues relevant to this application, considering the preventative measures available to combat cross-site scripting, evaluating the impact of query injection attacks and how they can be mitigated, and finally by discussing session management and broken authentication.

4.5.1 Password Encryption

For the protection of data, the JavaScript package “bcrypt.js” was implemented to allow for asymmetrical encryption. This process is performed on all passwords and also on the answers of user’s security questions. Asymmetrical encryption is notoriously complex to reverse and thus provides great protection to sensitive data. As an example, both of the following hashes are encrypted versions of the same password:

```
$2a$10$05RaTehC3VKPbzL9DzDm0.SZRSLG1v45uGGfJwV92kPsvHY3.29ea
```

```
$2a$10$HkLQG1J4xhL/ZK5a/bzXB0iYmbGcaP0ytxJ/Y7mqtXBGglE1o8bWq
```

By encrypting data, it prevents the disclosure of personal information in the event of a potential data breach. Passwords are particularly essential to protect as many people reuse them for logging into different platforms, so the leaking of unencrypted passwords can leave an individual in a compromised position.

4.5.2 Vulnerability Databases

Vulnerability databases are large public domain repositories that contain a comprehensive collection of known software security issues. They detail the nature of the issue, the versions impacted, and information regarding how they can be patched or resolved.

The largest of such databases is the Common Vulnerabilities and Exposures (CVE) database operated by The Mitre Corporation (2018) which pools information from various sources including the National Vulnerability Database (NVD), compiled by the National Institute of Standards and Technology (2018).

From this database, the core technologies were investigated to ensure that vulnerabilities could be established and mitigated. For this project Node.js, Express.js, Bcrypt.js, and EJS were all investigated. No errors were reported with Bcrypt, and although EJS did return known vulnerabilities, they only impacted versions older than 2.5.3, so as this application is running on 5.6.0, this is not relevant. There were several issues flagged with Express.js, however, they were associated with modules not used by this website. The search against Node.js returned the most problematic results, and were concerned predominantly with Denial of Service attacks and system configuration; which should be considered carefully when configuring the host server.

4.5.3 Cross Site Scripting (XSS)

The EJS template delivery framework utilises ‘scriptlet’ code that allows for the embedding of HTML for the mediation of cross-site scripting. Generally, when variables are printed from the template engine they are done so in a style similar in format to the following code: `<%= embedded_item %>`. This specifies escaped HTML, and is primarily used to print client side variables in addition to user content. One notable exception to this is the embedding of HTML objects (such as forms), for this code was formatted in the following way: `<%- embedded_item %>`. Peterson (2015) states that this is only effective when dealing with string data, so that other data types (such as numbers) should be treated with additional caution.

Query Injection Prevention

Query Injection is a scenario where NoSQL databases hold a significant advantage over their more traditional counterparts. Regarding this system, MongoDB in particular assembles queries into an object (not a string) known as a BSON object, and consequently these BSON-formatted queries eliminate the issue of query injection (MongoDB, 2018).

Session Management and Broken Authentication

The final aspect of security that was considered when developing this project, was the management of user sessions and broken authentication. The user session is handled by the `express-session`, which is a widely implemented npm package, with a vast amount of support. The default status of the session ID cookie is “not secure”, however, when run on an HTTPS server, there is an option to enable this security functionality, which is highly recommended for this application, mitigating issues caused by guessable a session ID. Effective encryption of session data, only storing essential data in the session, and the effective deletion of session information post usage were other methods used to secure browsing sessions.

Continuing from the previous point, the use of secure Hypertext Transfer Protocol (HTTPS) is vitally important to the secure exchange of data, and for the prevention of malicious behaviour, therefore this website should be hosted on an HTTPS certified server. In addition to this, automated systems that detect session management vulnerability should also be explored. (Takamatsu et al., 2012)

Chapter 5

Evaluation and Discussion

5.1 Website Evaluation

5.1.1 User Assessment

User Information

In order to gain valuable and useful feedback regarding the deliverable, a user assessment session was undertaken. Contact was made with a member of the administrative team within the University of East Anglia School of Computer Science who deals with ethics requests, and a one-on-one meeting was then arranged.

The representative I spoke to from the University has been working with ethics approval requests for over seven years, and so was therefore very familiar with how the system worked, which was beneficial for the purposes of appraisal.

Assessment Process

The evaluation began by a brief introduction and an explanation of the objective of the evaluation. I then asked a few preliminary questions to establish the users background and what interactions they had with the current system, and what they thought of how tasks were currently performed. After the introduction and preliminary questions, the web application was explained and the user was asked for their opinion of the idea and whether such a system would be helpful.

Once their initial thoughts were recorded, a demonstration was given, and a set of scenarios were carried out, with the user being given opportunities to ask questions throughout this process. During the demonstration, two sets of tasks were presented, one set were applicant-oriented, and the other set were administrator-oriented. Those targeted at the applicant were the completing and updating of an application. The tasks aimed at administrators were searching for, updating, and changing the status of an application; adding a comment; and creating a new form.

After the demonstration had come to an end, the final stage of the user assessment was to ask questions regarding what had just been shown to them and a few follow-up questions regarding their overall opinion of the deliverable and the potential barriers to adoption.

Comments and Feedback

Generally feedback was very positive, with the colour scheme, clear and uncluttered interface, and intuitive design all being praised by the user. The functionality was also deemed as being sufficient to satisfy the requirements of the current system, with the user commenting that applications completed on the website mirror those submitted to the current system in place.

Although most of the feedback was positive, one point was made regarding the system support. The user stated that it would be helpful to produce documentation or tutorials that can be accessed by administrators, in order to explain some of the more complex functionality. It was also mentioned that the training of staff to use the system may also be worth considering.

In addition to this issue with user support, an oversight was also brought to my attention. Typically at the end of each year a spreadsheet is produced detailing a breakdown of how many requests have been submitted to the system and the eventual status of these requests, a feature that was missing from this system.

User Recommendations

As a consequence of this user assessment there were three recommendations made in order to improve the adoption of this system. The first of these was to either create some documentation or performing a training programme in order to assist new users with learning how to utilise all the functionality available on the website. The second suggestion was to discuss with senior faculty which department will be responsible for the maintenance of this website; and if it were to be implemented across the entire university, create an agreement regarding how will it be managed. The final recommendation was to implement functionality that would allow administrators to produce and excel spreadsheet summarising the total number of applications and the success rate of these applications.

5.1.2 Future Evaluation

Ensuring the continued improvement of the web application post-production is an important element of the software life-cycle, so therefore this has been considered in the evaluation of this project.

Currently in the system there is a Frequently Asked Questions section of the application to provide support for common issues to be addressed without the need for the intervention of system administrator.

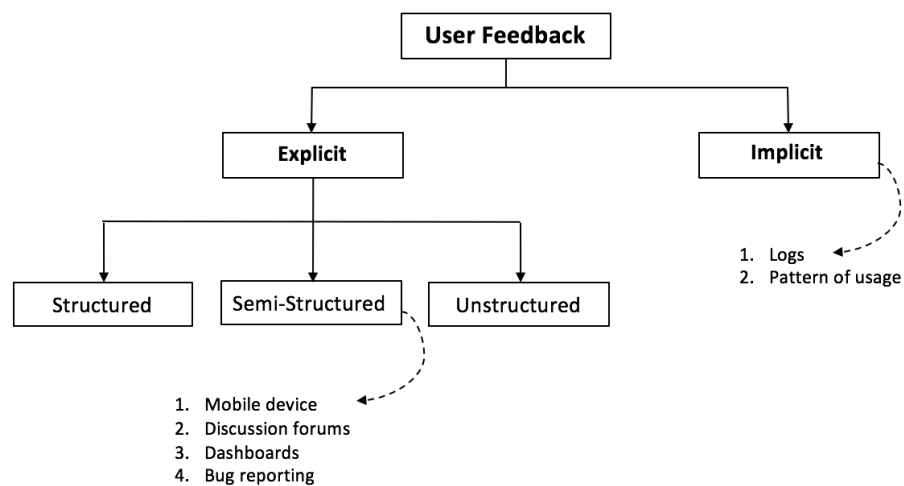
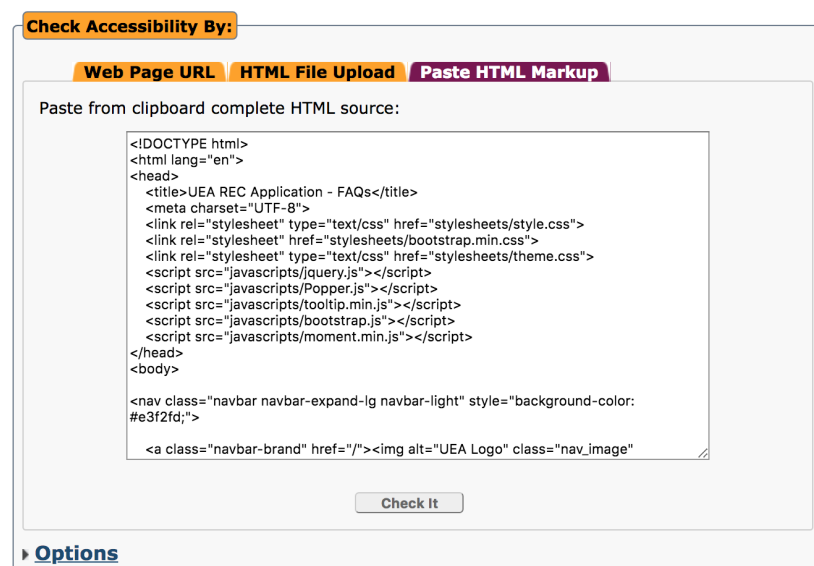


Figure 5.1: Types of Feedback Sammaneh (2018).

As shown in figure 5.1, there are several potential avenues for gaining user feedback to facilitate system improvements. The two most straightforward to implement are the addition of a bug-reporting section for administrators and a discussion forum for users, which are both recommended as future developments.

5.1.3 Accessibility Testing

For this web application, accessibility was assessed using the online tool “AChecker”, which evaluates the compliance of websites with the WCAG 2.0 guidelines (Caldwell et al., 2008) through an inspection of the HTML source files (Wille et al., 2016), and was chosen from a list of tools approved by W3C (2016). Ideally the interface design would have been tested to WCAG 2.1 standards, however, as these were released recently there are not many openly available options to pursue.



The screenshot shows the AChecker web interface. At the top, there's a header "Check Accessibility By:" with three tabs: "Web Page URL", "HTML File Upload", and "Paste HTML Markup". The "Paste HTML Markup" tab is selected. Below the tabs, there's a text area labeled "Paste from clipboard complete HTML source:". Inside this text area, the following HTML code is pasted:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>UEA REC Application - FAQs</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="stylesheets/style.css">
  <link rel="stylesheet" href="stylesheets/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="stylesheets/theme.css">
  <script src="javascripts/jquery.js"></script>
  <script src="javascripts/Popper.js"></script>
  <script src="javascripts/tooltip.min.js"></script>
  <script src="javascripts/bootstrap.js"></script>
  <script src="javascripts/moment.min.js"></script>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light" style="background-color:
#e3f2fd;">

  <a class="navbar-brand" href="/"><img alt="UEA Logo" class="nav_image"
```

Below the text area, there is a "Check It" button. At the bottom left, there is a link labeled "Options".

Figure 5.2: AChecker analysis input.

For this process, the HTML source code was copied and entered into the evaluation text area, and a series of tests were executed (figure 5.2). Each of the interface pages were systematically run through the accessibility checker and any compliance issues were resolved immediately. The three most common errors flagged-up were issues with nested headings, missing alt tags from images, and input fields without labels. These issues were then resolved and each page was then retested, until a positive result was given to known errors and compliance could be signed off (figure 5.3).

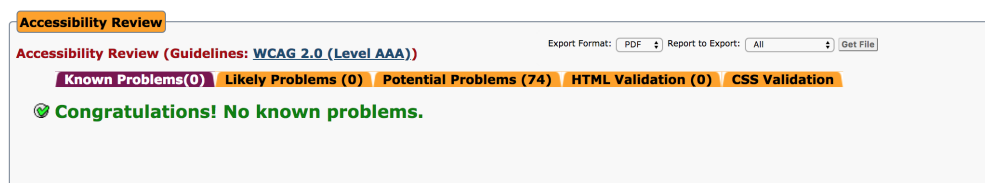


Figure 5.3: AChecker analysis output.

Breakdown of Accessibility Evaluation (WCAG 2.0 AAA)			
	Problems		
Page	Known	Likely	Potential
Index	0	0	102
Admin Dashboard	0	0	194
User Dashboard	0	0	165
View Module Forms	0	0	66
View Individual Forms	0	0	70
New Module Application	0	0	74
New Individual Application	0	0	94
Update Application	0	0	111
Upload File	0	0	85
Create Form	0	0	158
Choose Form to Update	0	0	178
Update Form	0	0	82
Profile	0	0	135
View Application	0	0	166
FAQs	0	0	160
High Priority	0	0	93

Table 5.1: Breakdown of Accessibility Evaluation for WCAG 2.0 AAA Standards

Table 5.1 shows the results from the AChecker accessibility assessment application. From this table it is possible to see that all pages have passed the inspection for known and likely problems. These results are very promising as it demonstrates that this application will be usable for all, including those with disabilities that may normally limit their user experience. Potential errors are quite high in some areas, and should be monitored considered, however they do not compromise the user experience.

5.2 Implementation Feasibility

5.2.1 Evaluation of Deliverable

The final deliverable has achieved all of the ‘must’ functional and technical requirements as outlined in tables 3.3 and 3.4. In addition to these, a significant number of the ‘should’ requirements have also been executed. This demonstrates that this system has the core functionality that is necessary for the effective implementation of such a system. Areas of functionality include the ability to: create and update, and disable forms, complete and amend applications, upload files and associated them with an application, change the status of an application, leave comments and feedback on applications, search for an application, reset passwords, and change their personal information.

In addition to the functionality available to users and administrators, there are also some automated processes, including the setting of estimated evaluation dates (an estimate of when an application will be evaluated) and a high priority page, which displays overdue applications, and ones which require urgent attention (due in three days or less). The evaluation dates are derived by looking at the type of application (module or individual), the status of the applicant (postgraduate, undergraduate, staff etc.), and the school associated with the application. These can be adjusted and set in the website configurations in order to customise for the particular instance.

In terms of the user experience and accessibility for this website, additional efforts have been made to ensure this application passes accessibility guideline assessments, so this system can be reached and used by as many people as possible. Therefore, in this respect, the application is ready for implementation immediately. It is important to highlight that any additional functionality or pages added to this website should adhere to the standards of the current functionality in order to continue complete accessibility and instil style consistency to aid usability and structure.

5.2.2 System Adoption

The adoption of any system can present issues and there are numerous barriers to overcome, for this system one of the key issues will be training. Despite efforts being made to make this system intuitive and clear, some aspects of this system are quite complex (especially for system administrators), and as it will be unfamiliar to the majority of users, support is necessary.

Steps have already been taken to achieve this, such as the partial implementation of a frequently asked question segment onto the system, additional efforts will be needed to complete this library in order to be as effective as possible. Further to this it would be advisable to publish tutorial videos to aid with learning and also run workshops for administrators to run through the less obvious features of the system.

For non-administrative users, it may be useful to set up a question forum that can be used for peer-assisted learning and act as a platform for users to recommend additions or changes to the system, however this should only really be explored if this system were to experience a significant increase of user base, otherwise it will not be a helpful tool.

5.2.3 Necessary Amendments

The effective adoption of this web application will require the adjustment of some elements of the configuration and the implementation of some prerequisite conditions. One of these is the setting up of secure servers (for both the Node.js application and the MongoDB), and should be configured in conjunction with any university data security and privacy policies. There may also be internal policies regarding what data can and cannot be stored, and additional data protection procedures governing, these must also be considered and amendments to what data is recorded should be made accordingly. There may also be a need to implement the facility to record the length of time certain data is held on any databases in order to comply fully with regulations.

5.3 Testing

5.3.1 Test Cases

Test cases were developed in order to populate the database with information to allow for a realistic user experience from the perspectives of both the applicants and the administrators. Multiple variations of Users, Forms and Form layouts were generated in order to gauge how these objects interact with one-another. Users were made from each of the user types (UG, PGT, PGR, S(RA), Faculty and Other). Forms layouts were then created for each of the document types (individual and module), which contained the different variations of inputs (text, number, textarea, boolean select field, and custom select field); in addition to these static text and headings were also added to the forms. Once the users and forms had been created, some example applications were uploaded and files were added.

5.3.2 Integration Testing

Once all of the test data was inserted into the database, integration testing could be completed. From administrator accounts, forms were updated and disabled, application statuses were changed and comments were left, and applications were searched for using the search bar on the dashboard page. From user accounts, new application were created and amended.

After this stage of the integration testing, it was clearer to see how each of the components interacted with one another. These interactions demonstrated that the system was working as expected, however, there was one factor that could not be observed unless manual adjustments were made. This example was the “High Priority” section of the application. Submission and evaluation times were amended in order for this functionality to be demonstrated.

Although this is a fairly simplistic approach to software testing, it is an effective one nonetheless, and used alongside various debugging tests proves to be a very powerful tool when evaluation the system as whole.

Chapter 6

Conclusions

6.1 Project Summary

6.1.1 Overall Outcome

This dissertation project documents the system development life-cycle of a web application for the University of East Anglia Research Ethics Committee.; from the conception of the idea and the elicitation of requirements through to the testing and signing off of a deliverable.

From the evaluation it can be seen that all of the essential requirements proposed in the original MoSCoW analysis have been achieved. The web application has an intuitive design that passes all WCAG 2.0 standards, administrators are capable of creating and updating forms, as well as reviewing applications with functionality to provide feedback; and applicants are able to fill in applications and amend them as necessary.

A fully functioning and tested website has been produced on behalf of the University of East Anglia Research Ethics Committee, which provides the ability for the to fully automate their current system. Topics such as requirement elicitation, systems development, website accessibility, user experience design, system architecture, data protection and security, legal compliance, database design, and software testing have been addressed over the course of this project; and their effect on the development of a web application have been observed.

6.1.2 Final Thoughts

Overall, this project has provided a fantastic opportunity gain insight into many different facets of web development technologies and processes, allowing me to acquire a deeper understanding of user experience design, information visualisation, accessibility requirements and data manipulation. In addition to this, I have also gained valuable experience of project management principles, such as time and resource planning, working to deadlines, addressing scope creep, and requirement elicitation. Following every stage of the software development life-cycle and learning about what is required at each step has given me a greater appreciation of the nuances of application development, and allowed me to improve both my functional and technical perspectives of a software development project.

6.2 Further Work

During this project, although many of the requirements have been implemented, there are areas that require additional consideration and effort. Due to the time and resource restraints of this project, although the security factors impacting the hosting of this application were discussed, the specifics were not finalised, and so therefore this system is currently only locally hosted on a private repository. Therefore one of the core areas that would need to be subject to further investigation is how this website should be hosted and managed. As discussed earlier, due to the nature of the data, an HTTPS certified server would need to be utilised in order for secure session IDs to be implemented. Additionally, Node.js is found to be more effective when running on an Nginx server, so consideration as to the viability could be investigated. Additional research into networking could also be undertaken to establish if any further optimisation could be facilitated. Especially if this system is to be integrated within other University of East Anglia IT services.

Bibliography

- Acharya, S., Ehrenreich, B., and Marciniak, J. (2015). Owasp inspired mobile security. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 782–784.
- Acosta-Vargas, P., Acosta, T., and Lujan-Mora, S. (2018). Challenges to assess accessibility in higher education websites: A comparative study of latin america universities. *IEEE ACCESS*, 6:36500 – 36508.
- Aizpurua, A., Harper, S., and Vigo, M. (2016). Exploring the relationship between web accessibility and user experience. *International Journal of Human - Computer Studies*, 91:13 – 23.
- Alexa (2018). Alexa top 500 global sites. Retrieved from <https://www.alexa.com/topsites>.
- Borkin, M., Bylinskii, Z., Kim, N., Bainbridge, C., Yeh, C., Borkin, D., Pfister, H., and Oliva, A. (2015). Beyond memorability: Visualization recognition and recall. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1.
- Caldwell, B., Cooper, M., Guarino, L., and Vanderheiden, G. (2008). Web content accessibility guidelines (wcag) 2.0. Retrieved from <https://www.w3.org/TR/WCAG20/>.
- Chaniotis, I. K., Kyriakou, K.-I. D., and Tselikas, N. D. (2014). Is node.js a viable option for building modern web applications? a performance evaluation study. *Computing*, 97:1023 – 1044.
- Data Protection Act 1998 (1998). Legislation.gov.uk. Available at: <http://www.legislation.gov.uk/>.
- Directive 2002/58/EC (2002). The european parliament and the council of the european union. Retrieved from <https://eur-lex.europa.eu/eli/dir/2002/58/oj>.

- Directive 95/46/EC (1995). The european parliament and the council of the european union. Retrieved from <https://eur-lex.europa.eu/eli/dir/1995/46/oj>.
- Garrett, J. (2003). *The Elements of User Experience*. [Indianapolis] : New Riders, [2003].
- Gessert, F., Wingerath, W., Friedrich, S., and Ritter, N. (2017). Nosql database systems: a survey and decision guidance. *Computer Science - R&D*, 32(3-4):353–365.
- Giraud, S., Thérouanne, P., and Steiner, D. D. (2018). Web accessibility: Filtering redundant and irrelevant information improves website usability for blind users. *International Journal of Human-Computer Studies*, 111:23 – 35.
- Government Digital Service (2018). Make your public sector website or app accessible. Retrieved from <https://www.gov.uk/guidance/accessibility-requirements-for-public-sector-websites-and-apps>.
- gov.uk (2010). Equality act 2010. Retrieved from <https://www.legislation.gov.uk/ukpga/2010/15/contents>.
- Graham, D., Benest, I., and Nicholl, P. (2007). Cognitive issues in information visualisation design for interaction for the visually impaired. In *2007 11th International Conference Information Visualization (IV '07)*, pages 917–920.
- Gu, Y., Wang, X., Shen, S., Wang, J., and Kim, J. (2015). Analysis of data storage mechanism in nosql database mongodb. In *2015 IEEE International Conference on Consumer Electronics - Taiwan*, pages 70–71.
- Huluka, D. and Popov, O. (2012). Root cause analysis of session management and broken authentication vulnerabilities. In *World Congress on Internet Security (WorldCIS-2012)*, pages 82–86.
- Hydara, I., Sultan, A. B. M., Zulzalil, H., and Admodisastro, N. (2015). Current state of research on cross-site scripting (xss) - a systematic literature review. *Information and Software Technology*, 58:170 – 186.
- Ivanovs, A. (2018). Top 10 templating engines for javascript to improve and simplify your workflow 2018. Retrieved from <https://colorlib.com/wp/top-templating-engines-for-javascript/>.

- Kirkpatrick, A., O'Connor, J., Campbell, A., and Cooper, M. (2018). Web content accessibility guidelines (wcag) 2.1. Retrieved from <https://www.w3.org/TR/WCAG21/>.
- Kjeøy, M. A. and Stalheim, G. M. (2007). Use cases in practice: A study in the norwegian software industry.
- Krug, S. (2014). *Don't make me think, revisited : a common sense approach to web usability*. [Berkeley] : New Riders, [2014].
- Kungurtsev, A., Nguyen, T., and Novikova, N. (2017). Technology for testing of software modules based on use cases. *Trudy Odesskogo Politehnieskogo Universiteta, Vol 3, Iss 53, Pp 79-87 (2017)*, (53):79.
- Liang, L., Zhu, L., Shang, W., Feng, D., and Xiao, Z. (2017). Express supervision system based on nodejs and mongodb. In *16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, Wuhan, China, May 24-26, 2017*, pages 607–612.
- Liang, Y. (2003). From use cases to classes: a way of building object model with uml. *Information & Software Technology*, 45(2):83.
- Lidwell, W., Holden, K., Butler, J., and Elam, K. (2010). *Universal principles of design : 115 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design*. [Beverly, Mass.] : Rockport Publishers, [2010].
- Little, A. (2013). Storyboarding in the software design process. Retrieved from <http://uxmag.com/articles/storyboarding-in-the-software-design-process>.
- MongoDB (2018). FAQ: MongoDB Fundamentals – MongoDB Manual. Retrieved from <https://docs.mongodb.com/manual/faq/fundamentals/#how-does-mongodb-address-sql-or-query-injection>.
- National Institute of Standards and Technology (2018). NVD - Home. Retrieved from <https://nvd.nist.gov/>.
- Nielsen, J. (2000). *Designing Web usability*. [Indianapolis] : New Riders, [2000].
- owasp.org (2017). Owasp top ten project. Retrieved from https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

- Peterson, T. (2015). Will ejs escape save me from xss? sorta. Retrieved from <http://www.managerjs.com/blog/2015/05/will-ejs-escape-save-me-from-xss-sorta/>.
- Pokorny, J. (2013). Nosql databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, (1):69.
- Regulation (EU) 2016/679 (2016). The european parliament and the council of the european union. Retrieved from <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- Sammaneh, H. (2018). Requirements elicitation with the existence of similar applications: A conceptual framework. In *2018 International Conference on Computer and Applications (ICCA)*, pages 444–9.
- Sarmah, U., Bhattacharyya, D., and Kalita, J. (2018). A survey of detection methods for xss attacks. *Journal of Network and Computer Applications*, 118:113 – 143.
- Schmutz, S., Sonderegger, A., and Sauer, J. (2018). Effects of accessible website design on nondisabled users: age and device as moderating factors. *Ergonomics*, 61(5):697 – 709.
- Serby, P. (2012). Case study: How & why to build a consumer app with node.js. Retrieved from <https://venturebeat.com/2012/01/07/building-consumer-apps-with-node/>.
- Sirikitsathian, P., Chaveesuk, S., and Sathitwiriawong, C. (2017). A conceptual framework for better understanding of factors influencing accessibility to a website and its acceptance by university students with visual impairments. In *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–5.
- Soper, D. S. and Piepkorn, F. (2018). Halo effect contamination in assessments of web interface design. *Open Journal of Information Systems (OJIS)*, 5(1):1–23.
- statista (2018). Mobile internet - statistics & facts. Retrieved from <https://www.statista.com/topics/779/mobile-internet/>.
- Sutcliffe, A. and Hart, J. (2017). Analyzing the role of interactivity in user experience. *International Journal of Human-Computer Interaction*, 33(3):229 – 240.
- Takamatsu, Y., Kosuga, Y., and Kono, K. (2012). Automated detection of session management vulnerabilities in web applications. In *2012 Tenth Annual International Conference on Privacy, Security and Trust*, pages 112–119.

- The British Standards Institution (2008). Bsi british standards invites comments on new draft standard on accessible websites. Retrieved from <https://www.bsigroup.com/en-GB/about-bsi/media-centre/press-releases/2008/12/BSI-British-Standards-invites-comments-on-new-draft-standard-on-accessible-websites/>.
- The Mitre Corporation (2018). Common Vulnerabilities and Exposures (CVE). Retrieved from <http://cve.mitre.org/>.
- Tilkov, S. and Vinoski, S. (2010). Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 97:80 – 83.
- University of East Anglia (2015). Equality data annual report. Retrieved from <https://portal.uea.ac.uk/documents/6207125/6639021/Equality+Data+Report+2015.pdf>.
- usability.gov (2018). User experience basics. Retrieved from <https://www.usability.gov/what-and-why/user-experience.html>.
- Vlsaggio, C. A. and Blasio, L. C. (2010). Session management vulnerabilities in today’s web. *IEEE Security Privacy*, 8(5):48–56.
- W3C (2016). Web accessibility evaluation tools list. Retrieved from <https://www.w3.org/WAI/ER/tools/?q=wcag-20-w3c-web-content-accessibility-guidelines-20>.
- W3Techs - Web Technology Surveys (2018). Usage description of javascript. Retrieved from <https://w3techs.com/technologies/details/cp-javascript/all/all>.
- Wille, K., Dumke, R., and Wille, C. (2016). Measuring the accessibility based on web content accessibility guidelines. In *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pages 164–169.

Appendix A

Example Screenshots of Web Application

A.1 Index Page

The screenshot shows the index page of the University of East Anglia (UEA) REC Approval System. At the top, there is a light blue header bar containing the UEA logo on the left, and two input fields labeled 'username' and 'password' on the right, followed by a 'Log In' button. Below the header, the text 'University of East Anglia' and 'REC Approval System' is displayed. A link for 'Forgotten Password?' is located to the right of the header. The main content area is a light blue box with the heading 'First time here? Create an Account.' Below this heading, there are four input fields arranged in two rows. The first row contains 'First Name*' and 'Last Name*'. The second row contains 'Known As' and 'UEA Email*'. The 'UEA Email*' field has a dropdown menu showing '@uea.ac.uk'. Below these fields, there are two more fields: 'School*' and 'Type of User*'. The 'School*' field has a dropdown menu showing 'School'. The 'Type of User*' field has a dropdown menu showing 'Type of User'.

UEA

username password Log In

Forgotten Password?

University of East Anglia
REC Approval System

First time here? Create an Account.

First Name* Last Name*

First Name Last Name

Known As UEA Email*

Known As abc13xyz @uea.ac.uk

School* Type of User*

School Type of User

Figure A.1: Index page (application home page)

A.2 Administrator Perspective

A.2.1 Dashboard

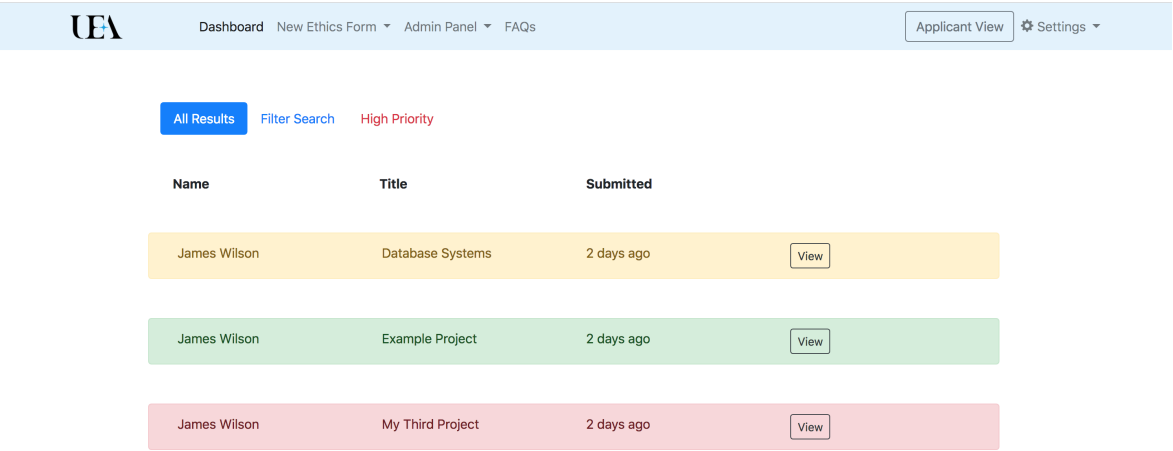


Figure A.2: Administrator dashboard - all results shown

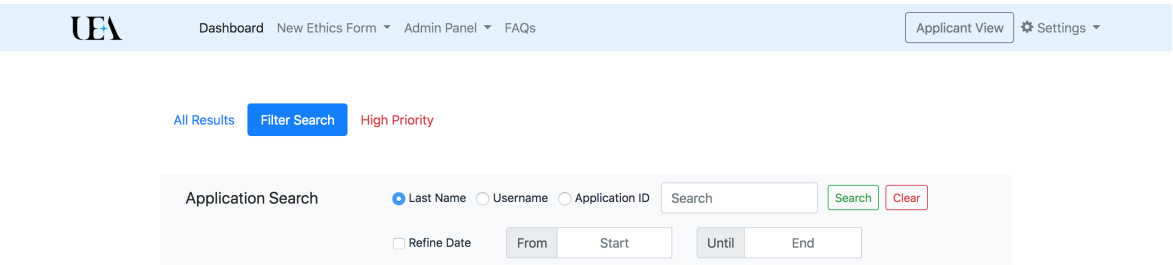


Figure A.3: Administrator dashboard - search bar

A.2.2 Form Management

UEA

[Dashboard](#) [New Ethics Form](#) [Admin Panel](#) [FAQs](#)

[Settings](#)

Create a Form

Form Type *

Individual

Form Title

+ Add Title

Form Subtitle

+ Add Subtitle

Add a Heading

Add an Input

Custom Select Field

Add Text

Add a Divider

Preview:

☐ Toggle Controls

Figure A.4: Form creation - component selector page

Preview:

☐ Toggle Controls

Name of Application

Subtitle

Title *

Section 1

Create Form

Figure A.5: Form can be created by administrator

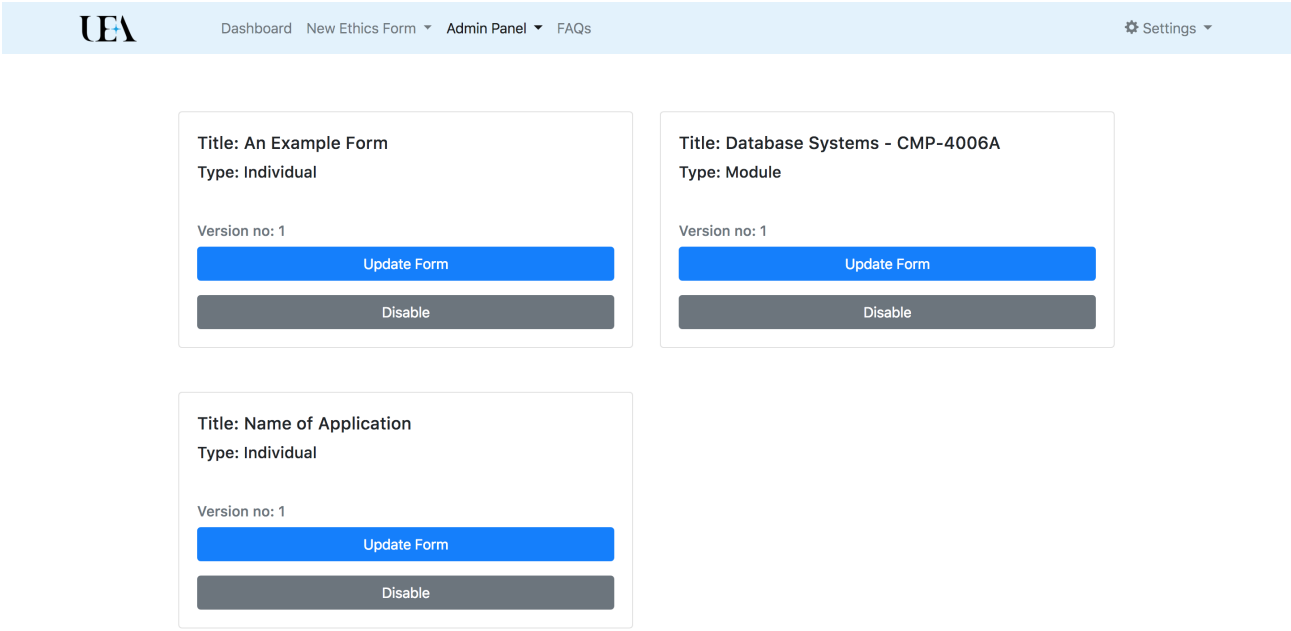


Figure A.6: Form created and ready for population by applicant

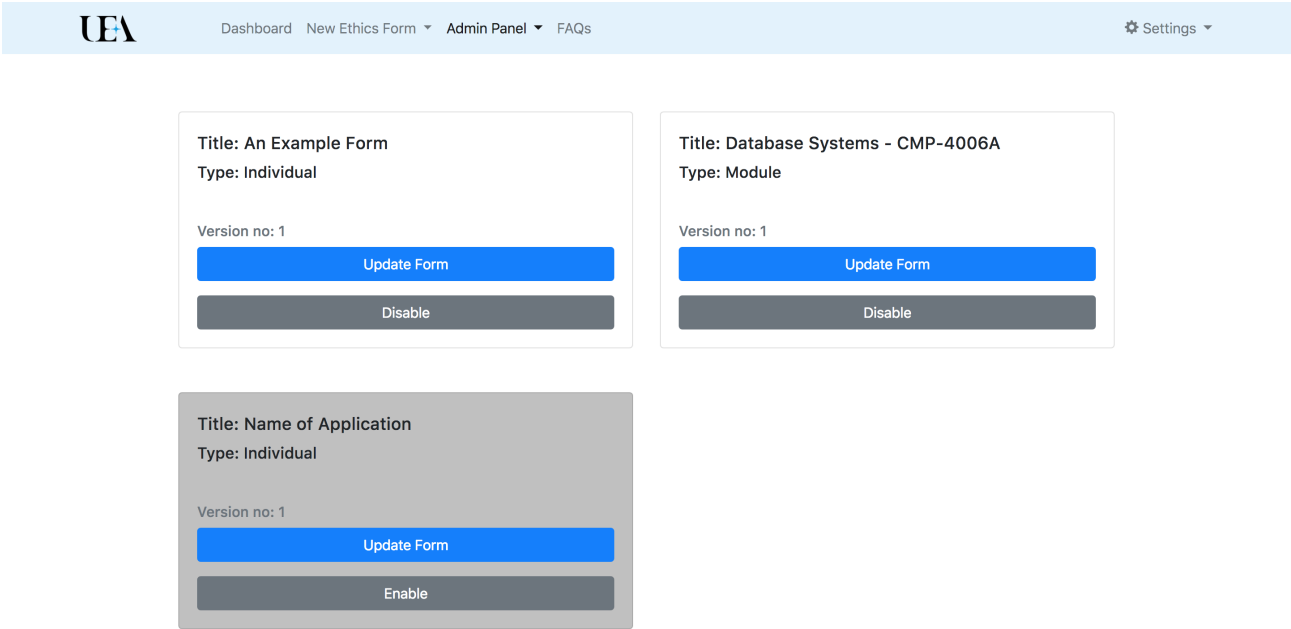



Figure A.7: Form can then be disabled by administrator

A.3 User Perspective

A.3.1 New Application

UEA

[Dashboard](#) [New Ethics Form](#) [FAQs](#)

 [Settings](#)

An Example Form

Subtitle

Title *

Header

An Input *

Another

Yes or No Compulsory *

No

Yes or No Example


No

Textarea

A.3.2 Dashboard

UEA

[Dashboard](#) [New Ethics Form](#) [FAQs](#)

 [Settings](#)

Hi Jim ,

Successful1

Title	Code	Submitted	
Example Project	CMP/18/PGT/2	2 days ago	<div><div>View</div><div>Update</div></div>

Pending4

Title	Code	Submitted	
Database Systems	CMP/18/PGT/1	2 days ago	<div><div>View</div><div>Update</div></div>
Example	CMP/18/PGT/4	11 hours ago	<div><div>View</div><div>Update</div></div>