

UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

A group design project report submitted for the award of
Master of Engineering Computer Science

Project Supervisor: Professor Mike Wald

Second Examiner: Dr. Gary B Wills

Interactive Learning within Online Media: A Framework for Investigating Video eAssessment

by Christopher Baines

Samuel Bennett

Harry Cutts

Christopher Hewett

Maria Lynch

January 28, 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

by Christopher Baines

Samuel Bennett

Harry Cutts

Christopher Hewett

Maria Lynch

This report details the creation of a framework for interactive learning and eAssessment within online videos.

Videos are often used as an educational tool within online learning systems, but limited information is known about the best way to include educational content and assessment within them.

The framework produced allows for the direct inclusion of instant feedback quizzes within videos to facilitate learning and eAssessment. There are now a number of published plugins and tools within this ecosystem. The primary focus was a question overlay allowing users to display arbitrary question sets and define the results of interactions with them.

To reduce the barrier to entry, an authoring tool that allows the creation of overlaid content has been developed.

To allow for further insight into the effect of this framework into eAssessment and learning an analytics plugin can record user interactions. This allows for future analysis of user behaviour.

Having finished the challenging and innovative project the team hopes that this framework will allow for a clear assessment of the viability of this learning model and integration into the Synote Annotation System.

Contents

Acknowledgements	xvii
1 Introduction	1
1.1 Problem	2
1.2 Goals and Scope	2
2 Previous Work	3
2.1 Background	4
2.2 Types of Assessment	4
2.3 eAssessment	6
2.3.1 Quiz Definition Standards	7
2.3.2 Usage of Video in eAssessment	9
2.3.3 Accessibility and Usability	9
2.4 Synote	10
2.5 Analytics	11
3 Overall Approach	13
3.1 Reviewing Previous Work	14
3.2 Client Requirements	14
3.2.1 AngularJS	15
3.3 Stakeholder Analysis	15
3.4 Requirements Analysis	16
3.4.1 Functional Requirements	16
3.4.2 Non-Functional Requirements	17
3.5 Modular Web server Approach	19
3.6 System Architecture	19
3.7 Deliverables	21
3.8 Testing	21
4 Project Management	23
4.1 Formation	24
4.1.1 Skills Audit	24
4.1.2 Roles and Responsibilities	24
4.2 Project Process	25
4.2.1 Statistical analysis of Project Progress	26
4.2.2 Collaboration and Issue Management	27
4.3 Client Management	28
4.4 Management Problems	28
5 Video Player	31

5.1	Introduction	32
5.2	Accessibility	32
5.2.1	Design and Implementation	32
5.2.2	Summary	33
5.3	Compatibility	33
5.4	Scrub Bar Extensions	34
5.4.1	Videogular Cuepoints	35
5.4.2	Videogular Heatmap	36
5.5	Summary	36
6	Videogular Questions	37
6.1	Introduction	38
6.2	Design	38
6.3	Implementation	38
6.3.1	Annotations	38
6.3.2	Front-End and User Interface	40
6.3.3	Back-End - WebWorkers	41
6.4	Summary	42
7	Authoring Tool	43
7.1	Introduction	44
7.2	Design	44
7.3	Implementation	45
7.3.1	Data Bubbling	45
7.3.2	Exporting of Data	47
7.3.3	Preview Tool	48
7.4	Summary	48
8	Analytics	51
8.1	Introduction	52
8.2	Design	52
8.3	Implementation	52
8.4	Summary	54
9	Testing	57
9.1	Introduction	58
9.2	Tools	58
9.2.1	Unit Testing Plan	58
9.2.2	JSHint	59
9.2.3	Karma	59
9.2.4	Jasmine	59
9.2.5	Python unittest Module	60
9.2.6	Locust	60
9.3	Videogular Questions Example	60
9.3.1	Videogular Questions	60
9.3.2	Videogular Cuepoints	61
9.3.3	Example Results Server	61
9.3.4	Accessibility	64

9.3.5	Deliverable Sign-Off Tests	64
9.4	Example Analytics Server	65
9.4.1	Cross-Origin Request Testing	65
9.4.2	Reloading Processed Data	66
9.5	Example Analytics Front-End	66
9.5.1	Videogular Heatmaps	66
9.5.2	Videogular Analytics Integration with Videogular Questions	68
9.5.3	Accessibility	68
9.5.4	Deliverable Sign-Off tests	69
9.6	Authoring Tool	70
9.6.1	Accessibility	70
9.6.2	Deliverable Sign-Off Tests	71
9.7	Summary	71
10	Future work	73
10.1	Introduction	74
10.2	Integration with Synote	74
10.3	Additional Video Players	74
10.4	Video Encoding	75
10.5	Mobile Interfaces	75
10.6	Second Screen	76
10.7	Angular 2	76
10.8	Accessibility	77
10.8.1	Cuepoints and Heatmap	77
10.9	Authoring Tool	77
10.10	Use of Graphs	77
10.11	Summary	78
11	Conclusions	79
11.1	Project Management	80
11.2	System Overview	80
11.3	Summary	81
	Glossary	87
	Appendix A Screenshots	89
A.1	Videogular Questions Example	90
A.2	Videogular Analytics	92
A.3	Authoring Tool	97
	Appendix B Skills Audit Results	99
B.1	Skills Audit Matrices	100
	Appendix C Coding Conventions	103
C.1	Introduction	104
C.2	General	104
C.3	CSS Coding Conventions	104
C.4	HTML coding conventions	105

C.5 Issue Tracker Usage Guidelines	105
C.6 JavaScript Coding Conventions	106
C.7 LaTeX	108
C.8 Python Coding Conventions	108
Appendix D Repositories	109
D.1 Introduction	110
D.2 Deliverables	110
D.3 Examples	112
D.4 Auxiliary	113
Appendix E Minutes of Meetings	115
E.1 Introduction	116
E.2 29th September 2014	116
E.3 6th October 2014	117
E.4 13th October 2014	118
E.5 20th October 2014	119
E.6 27th October 2014	120
E.7 10th November 2014	120
E.8 17th November 2014	122
E.9 24th November 2014	122
E.10 1st December 2014	123
E.11 8th December 2014	124
Appendix F Gantt Charts	125
F.1 Initial Plan	126
F.2 Actual	128
Appendix G Report Authorship	131
G.1 Introduction	132
G.2 Authorship	132
Appendix H Authoring Tool Wireframes	135
H.1 Introduction	136
H.2 Authoring Tool Primary User Interface	136
H.3 Question Creation Wireframes	140
H.4 Accessibility Tooltips	140
Appendix I Question Mockups	145
Appendix J User Study Results	149
Appendix K Code Fragments	157
K.1 Videogular Examples Code Fragments	158
K.2 Authoring Tool Code Fragments	159
Appendix L Deployment of Webservers	161
L.1 Introduction	162
L.2 Deployment of All Projects	162

L.3	Deployment of Flask Applications	162
L.4	Deployment of Node.js Servers	163
L.5	Deployment of AngularJs Files	164
Appendix M	Accessibility Standards	165
M.1	Introduction	166
M.2	Videogular Questions Example	166
M.3	Videogular Analytics Example	171
M.4	Authoring Tool	177
Appendix N	Firefox Bug Report	183
N.1	Introduction	184
N.2	What did you do?	184
N.3	What happened?	184
N.4	What should have happened?	184
N.5	Workaround	185
Appendix O	Deliverables Report	187
Appendix P	Project Brief	195

List of Figures

2.1	Components of an Managed Learning Environment (MLE) [25]	4
2.2	QTI components diagram [15]	7
2.3	Screenshot of a Synote replay correctly synchronised (http://www.synote.org/synote/recording/replay/114674/0 (Accessed: 15 Jan 15))	11
3.1	Stakeholder matrix for the project	16
3.2	System architecture diagram for the project	20
4.1	A punch card diagram showing the frequency of commits at different hours of the day. The area of each circle is proportional to the number of commits.	26
4.2	A bar chart of the number of GitHub issues being closed in each week of the project.	27
5.1	Screenshots of the accessibility improvements made to Videogular's HTML controls.	33
6.1	The architecture of Videogular Questions	38
6.2	A class diagram representing what attributes each question type needs	40
6.3	Sequence diagram showing interactions between the front end and WebWorker	41
7.1	Image showing a state diagram of how the authoring tool is used.	44
7.2	Bubbling of model data up controllers	46
7.3	State diagram illustrating decisions when bubbling data to root	46
7.4	Client-centric data binding updating the model in the parent controller by data bubbling illustrated in an state diagram	46
8.1	Sequence diagram showing the process of events being emitted by Videogular Questions (vgQuestions) plugin and sent to the REST service by the Videogular Analytics (vgAnalytics) plugin	53
9.1	The heat map has dynamically updated	67
9.2	Analytics events after being processed into viewer time periods	67
9.3	Graph showing results of questions sent from Videogular Analytics	68
A.1	Videogular Questions simple example poll	90
A.2	Videogular Questions Caesar cipher example with related questions	90
A.3	Videogular Questions example showing a single selection question	90
A.4	Videogular Questions example showing a multiple selection question	91
A.5	Videogular Questions example showing a stars rating question	91
A.6	Videogular Questions example showing a text question	91

A.7	Videogular Questions example showing a range question	92
A.8	Videogular Analytics example showing the video display	92
A.9	Videogular Analytics example showing the events display	93
A.10	Videogular Analytics example showing the statistics display	94
A.11	Videogular Analytics example showing the results display	95
A.12	Videogular Analytics example showing the results correlation display . . .	96
A.13	Final authoring tool user interface	97
F.1	Gantt Chart prior to the Christmas vacation.	126
F.2	Gantt Chart after the Christmas vacation	127
F.3	Gantt Chart prior to the Christmas vacation.	128
F.4	Gantt Chart after the Christmas vacation	129
H.1	Wireframe showing the accordion type design of showing/hiding options .	137
H.2	Wireframe showing the pop up type design of showing/hiding options before opening a pop up.	138
H.3	Wireframe showing the accordion type design of showing/hiding options after opening a pop up.	139
H.4	A wireframe showing the possible interface when creating a single choice poll type question	140
H.5	A wireframe showing the possible interface when creating a single choice quiz type question	141
H.6	A wireframe showing the possible interface when creating a multiple choice poll type question	141
H.7	A wireframe showing the possible interface when creating a multiple choice quiz type question	142
H.8	A wireframe showing the possible interface when creating a range or star poll type question	142
H.9	A wireframe showing the possible interface when creating a range or star quiz type question	142
H.10	A wireframe showing an example of how tooltips could be implemented .	143
I.1	An example of a single answer question	146
I.2	An example of a multiple answer question	146
I.3	An example of a text type question	147
I.4	An example of a range type question	147
I.5	An example of a stars type question	147
J.1	Page 1 of user study presentation	150
J.2	Page 2 of user study presentation	151
J.3	Page 3 of user study presentation	152
J.4	Page 4 of user study presentation	153
J.5	Page 5 of user study presentation	154

List of Tables

2.1	Perspectives on learning	5
2.2	Existing Video e-Quizzes	10
8.1	API of the emitted analytics events and their data payload	54
9.1	Time taken for each request to complete in milliseconds while stress testing the example results server	63
9.2	vgQuestions Example Deliverable sign-off tests	64
9.3	vgAnalytics Deliverable sign-off tests	69
9.4	Authoring tool Deliverable sign-off tests	71
M.1	Conformance to WCAG 2.0 Guidelines for Videogular Questions Example	166
M.2	Conformance to WCAG 2.0 Guidelines for Videogular Analytics Example	171
M.3	Conformance to WCAG 2.0 Guidelines for the Authoring Tool	177

Listings

7.1	Base template for authoring tool Definition File generation	47
7.2	The final Definition File is offered for downloading using a data blob URL	48
8.1	AngularJS demonstrating the message passing interface used in the Analytics plugin	53
9.1	Code showing appending Cross-Origin headers to all responses	65
K.1	Code for loading an annotation	158
K.2	Base template for a question asking if the viewer wishes to review the video section they answered incorrectly	159
L.1	Apache configuration	162
L.2	Apache configuration	163

Acknowledgements

The group would like to express their thanks to a number of people at the University who were instrumental in the success of this project.

Professor Mike Wald for both his exceptional supervision, and being a reasonable and communicative client.

Yunjia Li for his technical expertise on both Synote and software development in general.

Shameem Bajar for her help with gaining invaluable user feedback.

Dr Gary Wills for acting as an external source of help regarding the coursework.

Chapter

1

Introduction

“Without annotations a video is like a book without a contents.”
- Professor Mike Wald on Synote

1.1 Problem

Videos combined with quizzes and polls are used for educational purposes in many settings, including [Massive Open Online Courses \(MOOCs\)](#) and university lectures. For the authors of these resources, this often involves splitting videos up into sections with a poll or quiz after each section to gauge understanding. This approach requires the author to have video editing skills, and can be time-consuming. It would be useful if quizzes could be created and included directly into videos without the need for video editing. The resulting media could be analysed to discover how best to present information for learning.

1.2 Goals and Scope

This project will be split into three main sections:

1 Quiz Authoring Tool

The authoring tool (designed with accessibility in mind) will allow users to specify:

- sets of questions and polls to appear in the video,
- the time at which question sets should appear, and
- actions to be taken when questions are answered (e.g. skip back in the video if the answer is incorrect).

The export feature will produce a [Definition File](#) that contains all of the above information that is compatible with the video player and its plugins.

2 Questions Overlay

A library will be implemented to allow quizzes and polls to be overlaid on Web videos, playable in the major browsers (Mozilla Firefox, Microsoft Internet Explorer and Google Chrome) on different platforms (PC, Mac OS X and Android). These videos could be any of the main video formats (MP4, WebM, OGG). The library will read the [Definition File](#) and show the questions on the overlay at the appropriate time. Accessibility considerations for the overlaid videos will be investigated.

3 Video and Quiz Analytics

Metrics of user behaviour will be recorded and methods of displaying these to the author will be explored.

Previous Work

To avoid duplicating work done by other parties, an investigation into the current standing of related fields was undertaken, both in academia and the education industry. This showed where improvements could be made and identified methods and tools that could be used within the project. The areas studied were: types of assessment, eAssessment, quiz standards, video accessibility and usability, and analysis of video usage.

2.1 Background

A **Managed Learning Environment (MLE)** is a system used by an educational establishment to facilitate and manage learning. It is typically comprised of several subsystems (as seen in Figure 2.1).

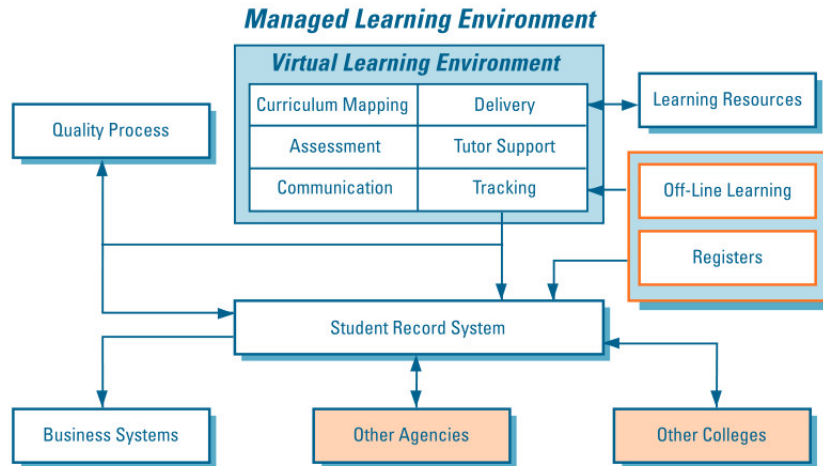


Figure 2.1: Components of an MLE [25]

This project focuses on the assessment system in the **Virtual Learning Environment (VLE)**.

2.2 Types of Assessment

In education, summative (or ‘assessment of learning’ [26]) and formative (or ‘assessment for learning’ [26]) assessments are often used to provide different types of feedback to students, and show their level of progression and competence. The method of assessment will depend on the perspective of the assessor (see Table 2.1).

Summative assessment is used when achievement levels need to be reported against a set of published criteria, often as an end of course evaluation. As all students are evaluated against the same criteria their results can be combined to form league tables etc. [12]. These assessments are generally high-stakes so “those being assessed are likely to do all they can to conceal ignorance and suggest competence” [16]. The feedback from such assessments is usually a mark, often given as a level of achievement or competence, rather than specific details on how the assessment could have been completed more successfully.

Table 2.1: Perspectives on learning and approaches to assessment and feedback (from [26])

Perspective on learning	Assumption	Assessment	Feedback
Associative	Learning as acquiring competence Learners acquire knowledge by building associations between different concepts. Learners gain skills by building progressively complex actions from component skills.	Concepts and competencies frequently assessed at micro level and in combination through macro-level tasks.	- Expert feedback focusing on weaknesses in skills and conceptual understanding - Interactive environments for knowledge and skills acquisition
Constructivist	Learning as achieving understanding Learners actively construct ideas by building and testing hypotheses.	Assessment by means of experimentation, discovery and inquiry-based tasks.	- Self-generated feedback arising from reflection and self-assessment - Interactive discovery environments with opportunities for self-testing
Social constructivist	Learning as achieving understanding Learners actively construct new ideas through collaborative activities and dialogue.	Collaborative and cooperative tasks involving shared expression of ideas. Participation by learners in the design of assessment tasks.	- Peer feedback arising from collaborative activities and dialogue - Interactive environments that support sharing and peer feedback

Continues on the next page...

Perspective on learning	Assumption	Assessment	Feedback
Situative	Learning as social practice Learners develop their identities through participation in specific communities of practice.	Holistic assessment in authentic or simulated professional contexts. Participation in social practices of inquiry and assessment.	- Socially produced feedback from multiple sources - Feedback derived from authentic real-life tasks - Interactive environments that simulate professional practice

Formative assessment is not strictly based on criteria but takes into account students' progress and effort [12]. The marks given for formative assessments may use the performance of students from several occurrences. These performances may be inconsistent but these differences in behaviour can be used as a diagnostic to locate where problems are [12]. Formative assessment must be student centred, as its two main aims are to get students to recognise the gap between the understanding level they have achieved and what is required, and for the students to take action to close this gap [3]. To identify the gap immediate feedback on how to improve is required [28] and the disclosure of concepts that are difficult needs to be encouraged [16] to be able to provide effective feedback.

The choice to use formative or summative assessment will depend on the purpose of the test. If a level of competence is required then summative assessment would be more appropriate; a test to highlight the areas that students are struggling with would be better as a formative assessment.

2.3 eAssessment

An **eAssessment** (also known as **Computer-Aided Assessment (CAA)**, **Computer Assisted Assessment (CAA)**, or **Computer Based Assessment (CBA)**) is the process of making, viewing and scoring assessments using a computer. In this report these processes will be carried out by two tools: the “**Authoring Tool**” will be responsible for creating the questions and assembling the test and the “**Assessment Delivery System**” will be responsible for displaying and scoring the assessments. These tools must be interoperable, meaning they must use data formats that are compatible with each other.

This project focuses on the use of quizzes and polls within videos as a means of assessment.

2.3.1 Quiz Definition Standards

In order to store and display the questions in an [eAssessment](#), a schema for the questions must be defined. The only formally defined standard found was IMS Global's [Question and Test Interoperability \(QTI\)](#) specification¹. Some other quiz definitions were found including Moodle's General Import Format Technology (GIFT)² and Aiken formats³.

GIFT is a question markup format by Moodle⁴ and allows the use of a text editor to create questions in some simple formats. There is a strict compliance to specific syntax required and for complex questions it is no longer intuitive [21]. Aiken is another markup format that is designed to be more human-readable, but again it has strict syntax that must be adhered to or the imports to Moodle would fail.

QTI is an industrial interoperability standard for [eAssessments](#). The specification overview [15] splits the [eAssessment](#) system into five subsystems: authoring tool, item bank, test construction tool, assessment delivery system and learning system (see [Figure 2.2](#)).

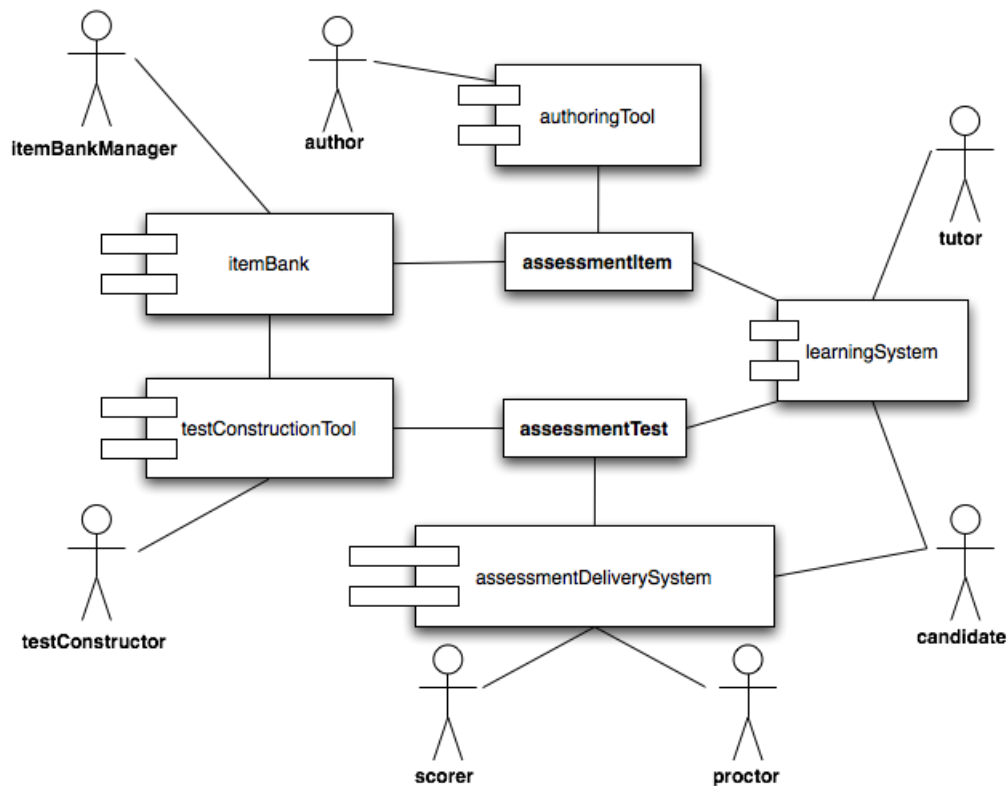


Figure 2.2: QTI components diagram [15]

¹<http://www.imsglobal.org/question/#version2.1> (Accessed: 24 Nov 14)

²https://docs.moodle.org/28/en/GIFT_format (Accessed: 24 Nov 14)

³https://docs.moodle.org/23/en/Aiken_format (Accessed: 24 Nov 14)

⁴<https://moodle.org> (Accessed: 24 Nov 14)

In the [QTI](#) definition the authoring tool is used to create individual questions. However, many systems combine this with the Item Bank and Test Construction Tool. For this report “[Authoring Tool](#)” refers to this combination and, as such, an authoring tool is used to create whole quizzes.

In an academic review [Hasani-Mavriqi et al. \(2010\)](#) [13] state:

The major promise of QTI is that by introduction of common format developers can concentrate on developing innovative tools, whereas teachers can focus on defining new and groundbreaking methods of how to apply those tools in an online environment.

It is a very complex specification with many ambiguous or optional elements. The complexity of the specification increases the likelihood for errors in the implementation as it opens up opportunities for developers to interpret the specification in different ways [21]. This means the intent of the original author may be lost.

[QTI](#) version 2.1 was finalised in 2012 [15] but up to this point the standard was not widely used [28] as there were several fundamental flaws in the standard. When the v2.1 draft was withdrawn in 2009 Rib Abel (IMS Global’s Chief Executive Officer) was quoted as saying (on v2.0) “its deficiencies are well known and IMS does not recommend implementation of it [...] the only version of QTI that is fully endorsed by IMS GLC is v1.2.1”⁵. This standard gave definitions in natural language. These could often be long-winded and ambiguous, making the standard difficult to implement [21, 24].

[QTI](#) v2.1 defines 18 types of interaction (question, answers and response patterns) [14]:

- | | | |
|------------------|-------------------|-----------------------|
| 1. Choice | 7. Text entry | 13. Graphic associate |
| 2. Order | 8. Extended text | 14. Graphic gap match |
| 3. Associate | 9. Hot text | 15. Position object |
| 4. Match | 10. Hotspot | 16. Slider |
| 5. Gap match | 11. Select point | 17. Drawing |
| 6. Inline choice | 12. Graphic order | 18. Upload file |

In addition, assessment items can have template and/or adaptive behaviour giving 72 options for each item’s type.

Template items have variables in the questions and answers. This allows similar questions to be defined using the same template item (e.g. by changing the numbers in a maths problem).

⁵<http://lists.ucles.org.uk/public/ims-qti/2009-March/001463.html> (Accessed: 25 Nov 14)

Adaptive items are scored over a series of questions, and allow different questions to be displayed based on the answer(s) given for the previous question(s).

2.3.2 Usage of Video in eAssessment

Video is often used as a medium to convey educational material as it appeals to different learning styles. However, academics have found that when video is streamed (viewed non-interactively) there is a lack of interactivity and user control [4, 9]. To involve the user more, interactive elements must be added. Cherrett et al. [4] found that 75% of users agreed or strongly agreed that interactive video had enhanced their learning experience, but at the time of the paper (2009) they could find no evidence of interactive video being used as a learning tool. There are now some interactive videos being used as learning tools [17] but not widely in MLEs.

Interactive videos have often been implemented in Flash [4, 20]. This is no longer supported on many devices in favour of [HyperText Markup Language version 5 \(HTML5\)](#) technologies [1]. The interactivity in [HTML5](#) video players is generally implemented using clickable areas or pop ups [22]. [HTML5](#) video players do not require an external plugin to run and so will be browser independent [10, 22], however some standards are not implemented in all browsers.

Including polls and quizzes in interactive videos is one method that could be used to give a formative assessment. This would give immediate, meaningful feedback, a key feature of formative assessment [28]. Having the questions integrated into the video allows the appropriate sections to be automatically re-watched, encouraging immediate action on feedback.

2.3.3 Accessibility and Usability

To allow interoperability between [eAssessment](#) systems the [QTI](#) specification involves both the model and the view of the data [13], making accessibility difficult to add in later. This means many systems that comply with the [QTI](#) specification are inaccessible.

[eAssessment](#) systems require an [Authoring Tool](#) that is easily understood by people who do not have in-depth technical knowledge (e.g. teachers). Often authoring tools that implement the [QTI](#) standard are too technical for this [13]. These tools need to improve their accessibility and usability, keeping the time commitments for users reasonable [2, 28] or they will not be used.

Kolodyazhnaya [17] did a check for accessibility and other features on existing applications in the education industry that use video e-quizzes and found that none of them were accessible from a keyboard (see [Table 2.2](#)). This means that they

do not conform to the [User Agent Accessibility Guidelines \(UAAG\)](#) of the [Web Accessibility Initiative \(WAI\)](#) [27] or the [Web Content Accessibility Guidelines \(WCAG\)](#) 2.0 Standards. Further research showed that Moodle also uses a video player to display content that is completely inaccessible to a user only using a keyboard⁶.

Table 2.2: A comparison between existing systems with interactive video e-quizzes (from [17])

Criteria	Adobe Captivate	Camtasia Studio	Edpuzzle	Educannon	Hapyak
Version	Desktop	Desktop	Web	Web	Web
HTML5 support	✓	None	✓	✓	✓
Replay function	None	✓	None	None	None
Flash fallback	✓	✓	✓	✓	✓
Accessibility	None	None	None	None	None
Overlay support	None	✓	None	None	✓

Keyboard-accessible and even clicker-accessible Web video players have been implemented by both Opera [7] and PayPal [18], using [HTML5](#) which conforms to the [WAI Accessible Rich Internet Applications \(ARIA\)](#) specification. PayPal's implementation is even open source.

There are many ways of conveying content in a video and none of the methods alone will generally convey the whole message. Most video searches stay on a whole resource level as there is a lack of semantic interlinking [19]. Annotation systems such as Synote (see [section 2.4](#)) aim to make videos accessible by adding transcripts and other annotations to them. The videos being annotated may not be owned by the annotator.

2.4 Synote

Synote⁷ is a Web multimedia annotation system designed and built with accessibility in mind.

⁶<https://tracker.moodle.org/browse/MDL-36081> (Accessed: 24 Nov 14)

⁷<http://synote.org/synote/> (Accessed: 24 Nov 14)

The application aims to make multimedia content accessible by synchronising the multimedia in many different formats (see Figure 2.3). The formats used are textual, video/audio, bookmarks and images.

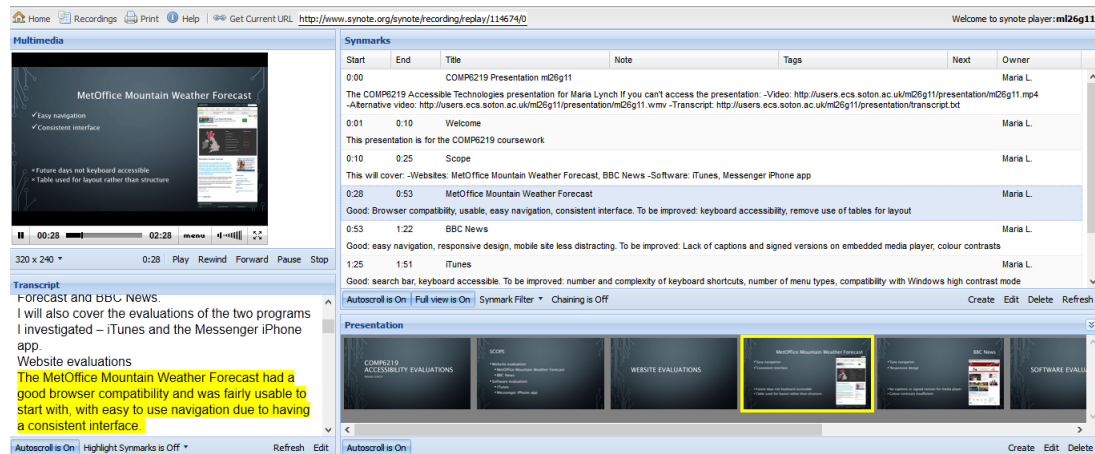


Figure 2.3: Screenshot of a Synote replay correctly synchronised (<http://www.synote.org/synote/recording/replay/114674/0> (Accessed: 15 Jan 15))

The textual element is a transcript of the video/audio. This aims to make the presentations more accessible for people with hearing impairments and cognitive difficulties. It also makes the presentation more easily searchable.

The bookmarks (known as Synmarks) can be used for many reasons. For example they could be used to provide summaries of the slides, make notes on the presentation or provide a direct copy of the text from the slides.

The images can be PowerPoint slides that are used in the presentation so that there can be other video to explain the slides at the same time.

Using the print function a screen reader friendly version of the presentation can be produced⁸. This provides the transcripts, bookmarks, and images in sequential order so that a screen reader can read all the text. This can be particularly useful to visually impaired users.

2.5 Analytics

Meaningful analysis of video data is difficult as you need to know the content of the video to understand why the sections may have been of interest to the user [23].

⁸http://www.synote.org/synote/recording/handlePrint?id=114674&_part=&part=on&from=&to=&_synmarked=&_synmarked-user-1771=&_transcript=&transcript=on&_presentation=&presentation=on&slideHeight=&_synmarks=&synmarks=on&_synmarks-user-1771=&_synmarks-user-1771=on&_synmarkId=&synmarkId=on&_synmarkTiming=&synmarkTiming=on&_synmarkTitle=&synmarkTitle=on&_synmarkNote=&synmarkNote=on&_synmarkTags=&synmarkTags=on&_synmarkOwner=&synmarkOwner=on&_synmarkNext=&synmarkNext=on (Accessed: 15 Jan 15)

Ronchetti [23] proposes that extra features can be added to the video to aid automatic analysis, including: multiple (parallel) cognitive channels, semantic marking, transcripts, and annotations. Video viewing styles have been researched for non-interactive videos (streaming) and in general four main styles were identified [9]:

- **Linear** – A student watches a video in one pass (without interruptions) from the beginning to the end
- **Elaboration** – A student watches a video again after finishing the first time in one pass
- **Maintenance rehearsal** – A student watches parts of a video repeatedly
- **Zapping** – A student skips through the instructional video at intervals of relatively short viewing times

With the growing popularity of [Massive Open Online Courses \(MOOCs\)](#) analysis of video tutorials has been important to the educational industry to increase their usage. [Guo et al. \[11\]](#) made several findings such as:

- Shorter videos are much more engaging
- Tablet drawing tutorials (like those on [Khan Academy](#)⁹) are more engaging than PowerPoint slides or code screencasting
- Students engage differently with lecture and tutorial videos

These findings would also apply to video quizzes but as the quizzes add a new element to the material the interactions will be different and there will be many more things that could be learned from studying their usage patterns.

Currently there is little research into viewing styles of interactive videos with quizzes and polls in them as there are few tools to capture the necessary data. [Chorianopoulos and Giannakos \[5\]](#) created an analytics tool that showed the quizzes separately to the video¹⁰ to create graphs of video usage patterns. They used the YouTube player to collect the usage data and Google Drive to include quizzes. They chose to display this information in graphs of video time vs frequency.

⁹<http://www.khanacademy.org/> (Accessed: 21 Jan 15)

¹⁰<http://www.socialskip.org/home> (Accessed: 01 Dec 14)

Overall Approach

Chapter

3

To create a high quality framework an overall approach for the project was decided on.

This was done by first summarising the research conducted into previous work, as well as defining where this project fits within that ecosystem. Specific requirements were gathered from the client and an analysis of the needs of other project stakeholders conducted. This led to a list of requirements that the project would need to meet. Additionally, thought was put into the general system design of the framework.

3.1 Reviewing Previous Work

Having reviewed the Previous Work ([chapter 2](#)), some design decisions for the project could be proposed to the client:

- A basic quiz standard would need to be defined to allow interoperability between the authoring tool and questions overlay. It was decided not to use the [Question and Test Interoperability \(QTI\)](#) standard to define the question sets due to the complexity of the standard, and the difficulty of implementing accessibility. The time constraint on this project would mean that more time would be spent on ensuring compliance with [QTI](#) than implementing the tools.
- The interactive video had to be able to provide immediate feedback, to allow effective formative assessment to take place.
- A [HyperText Markup Language version 5 \(HTML5\)](#) video player would be used to play the videos, enabling mobile devices that do not support Flash to play the interactive videos. Using [HTML5](#) for the control elements would also simplify the implementation of keyboard accessibility.
- The questions overlay would need to emit events that can be captured and used to analyse the users' viewing patterns. This would allow viewing patterns to be compared and contrasted with traditional videos.

3.2 Client Requirements

Because the long-term aim is to integrate this project into the new version of Synote, there were some specific requirements that the client had with regards to implementation.

The new version of Synote will be written in [AngularJS](#) and [HTML5](#). To make it more easily compatible the questions overlay should also be written using this framework. With this new requirement it was agreed that [Videogular](#)¹² should be used. [Videogular](#) is a [HTML5](#) video player for [AngularJS](#). The nature of the library makes it very easy to write plugins to add the required functionality.

In addition to quizzes, the inclusion of polls must be supported within the questions overlay.

¹<https://github.com/2fdevs/videogular>

²<https://github.com/2fdevs/bower-videogular-controls>

3.2.1 AngularJS

[AngularJS](#)³ is a JavaScript Framework that extends HTML attributes. It is an open source library made by Google.

The underlying architectural pattern used by [AngularJS](#) is client side [Model-View-Controller \(MVC\)](#). This is where the data (Model), appearance (View) and actions that can be applied (Controller) are separated out to make encapsulation and code reuse easier.

The framework allows custom HTML tags and attributes to be created using directives. Directives allow the user to specify the behaviour of specific elements they have created.

One of [AngularJS](#)'s most prominent features is its use of two-way data binding. This synchronises the views with the data held in the model. This is especially useful when dealing with dynamic content, as when extra content is added it is automatically shown in the view.

3.3 Stakeholder Analysis

To decide on an approach for the project the stakeholders needed to be identified and analysed. Their requests have been prioritised by categorising each stakeholder (see [Figure 3.1](#)).

The project supervisor and client, Professor Mike Wald, had the most interest and power in the project. As the project proposer it would be his ideas that would form the basis of the project.

Yunjia Li is the lead developer for the new version of Synote. As the aim of the project was to provide code for Synote he had a lot of interest in the project. His requirements would need to be taken into account to ensure interoperability between the two systems, which gave him power in the project.

Due to the nature of the project the developers themselves would be major stakeholders. Work would need to be evenly distributed to give all members of the group the same opportunities to obtain marks.

The University of Southampton sets the mark scheme and guidelines for the project. These would need to be satisfied to make the project successful.

2fdevs are the development team who created the [Videogular](#) player. The project would be creating plugins for this player, and so would rely on certain attributes of [Videogular](#), such as accessibility.

³<https://angularjs.org/>

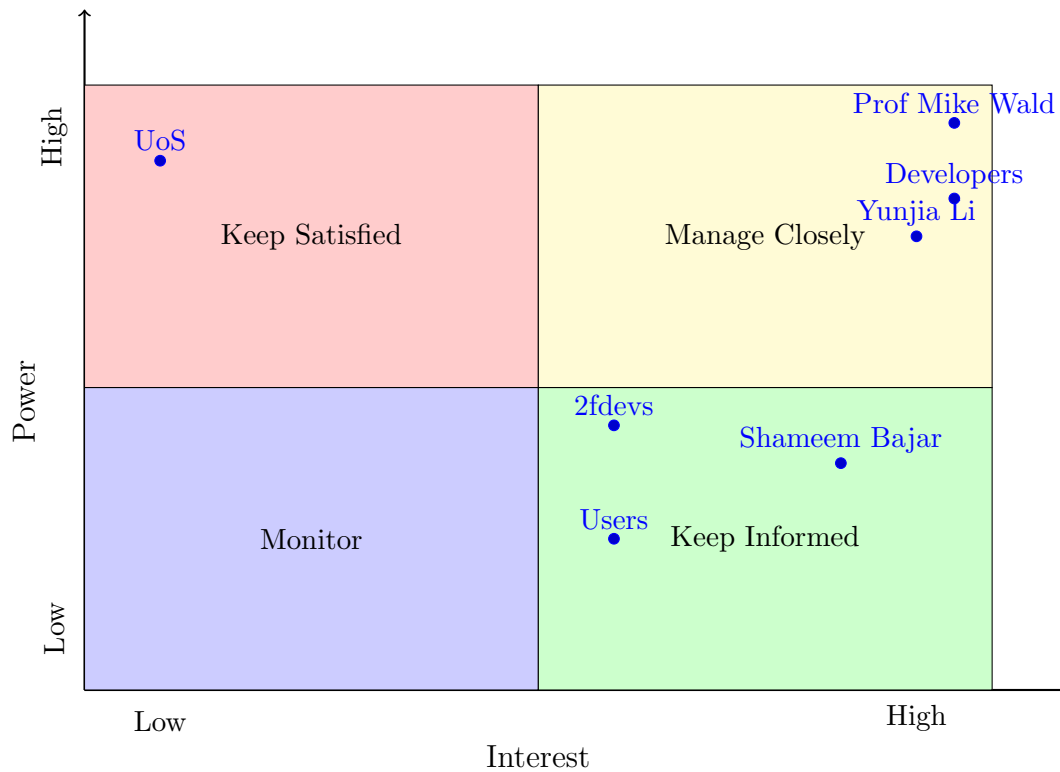


Figure 3.1: Stakeholder matrix for the project

Shameem Bajar is a 3rd year student who has an Individual Project based around this project. Her interest levels would be high as her work depends on the output of this project.

The users would be stakeholders, as the decisions made will affect how easy the system is to use and whether the desired functionality is available.

Requirements from all stakeholders will be considered but any conflicts will be dealt with by prioritising the needs of the most influential party (closer to the top right of Figure 3.1).

3.4 Requirements Analysis

Considering the needs of all stakeholders, a set of requirements for the project were agreed. The reasons for each being included are specified in *italics*.

3.4.1 Functional Requirements

F1 Question types

The questions overlay must be able to display multiple choice (with one or more

selectable answers), range selector, rating, and text based answer types. The stakeholders for each are described in italics after each one.

Authors wish to ask questions that require set responses.

F2 Jumping to content

There should be a feature to jump to specific content if a question is answered incorrectly.

This is a method of giving instant feedback, which was one of the things found to be important in the Previous Work (see [chapter 2](#)).

F3 Analytics events

Video and question events should be emitted from the questions overlay so that analytics data can be calculated.

Analysis of existing work showed a lack of analysis of video usage behaviour, so this would help facilitate new analysis.

F4 Cuepoints

The locations of the questions should be marked visually on the [scrub bar](#) of the video.

Users wish to see where the questions are in the video.

F5 Poll responses

The responses to polls must be able to be sent to a server, and results from the server must be able to be displayed.

This is one of the client's requirements.

F6 Instant feedback

Once a question or set of questions have been answered there must be a way of receiving instant feedback on performance.

Found to be important in the analysis of existing work.

3.4.2 Non-Functional Requirements

N1 Video player used

[Videogular](#) should be used as the video player, and components should be written in [AngularJS](#) and [HTML5](#).

One of Yunjia's requirements.

N2 Standalone

The plugins created should be standalone, but still easy to integrate into the new version of Synote.

One of the client's requirements.

N3 Browser compatibility

The questions overlay must work on Microsoft Internet Explorer 8, Mozilla Firefox and Google Chrome.

One of the client's requirements.

N4 Operating System Compatibility

The questions overlay must work on Windows, Mac OS X and Android.

One of the client's requirements.

N5 Extensibility

The questions overlay must be extensible so that further features can be added.

Makes incremental development easier for the developers, and allows any missing features that are wanted for integration into Synote to be added easily.

N6 Keyboard Accessibility

The questions overlay and authoring tool must be completely accessible to a keyboard-only user.

Users may not be able to use a mouse.

N7 Use of colour

Any colours used should be customisable.

Users may have visual impairments such as colour blindness, so they may only be able to use the application if it is in certain colours.

N8 User interface

A Graphical User Interface (GUI) must be provided for the Authoring Tool so that non-technical users can create quizzes.

One of the client's requirements is to reduce the barrier to entry.

N9 User friendliness

To allow maximum use of these tools, no tools should not require extensive training.

One of the client's requirements is to reduce the barrier to entry.

N10 Documentation and Testing Examples

The plugins built should have usage examples, to allow less technical users to implement their own system.

One of the client's requirements is to reduce the barrier to entry, and to ensure that the framework is easy to integrate.

N11 Server architecture

Any server behaviour should be accessed by [Representational State Transfer \(ReST\)](#) calls, which should work across hosts.

One of Yunjia's requirements to allow easier integration into Synote.

N12 Scalability

All tools should scale when they are used by large numbers of people.

One of the client's requirements is to ensure performance does not degrade when use of the tools are scaled.

3.5 Modular Web server Approach

It is important for this project to be easily integrated into other projects, particularly the latest version of Synote (see [section 2.4](#)). To accomplish this the design of the back-end systems ensures that they can be run without depending on any other modules. All of the functionality will be accessible via [ReST](#) calls as set out by [Requirement N11](#).

Abstracting all calls using [ReST](#) means that other services are able to interact with the back-end services in a language independent way, which also makes the application standalone ([Requirement N2](#)). To facilitate this, all [ReST](#) responses are returned with a Cross-Origin header. This allows servers which are not on the same machine to communicate with the [ReST](#) service.

These features allow an external application to run the Web server separately and communicate with the server side code. The only dependency added by those who use the code in this project is that they need to be able to make [ReST](#) calls.

3.6 System Architecture

The following set of components were decided on, the interactions of which are shown in [Figure 3.2](#):

- **Videogular Questions** displays questions over a [Videogular](#) video player according to a quiz defined in a [Definition File](#). If any of the questions are polls, it sends responses of the poll(s) to a results server, and displays the results that are sent back.
- **Videogular Analytics** logs events from [Videogular](#) (e.g. when the user plays or pauses the video) and Videogular Questions (e.g. when the user answers a question) and sends them to an analytics server.
- **Videogular Cuepoints** displays marks on the [scrub bar](#) indicating when questions will appear.
- The **results server** receives responses from Videogular Questions and returns the aggregated results on request.

- The **analytics server** receives event logs from Videogular Analytics, and provides a separate Web front-end, the **analytics front-end**, which displays analysis of the events.
- **Videogular Heatmap** overlays a heat map onto the [scrub bar](#) of the [Videogular](#) player, for visualisation of analytics results.
- An **authoring tool** allows a user to create [Definition Files](#) without having to write them manually.

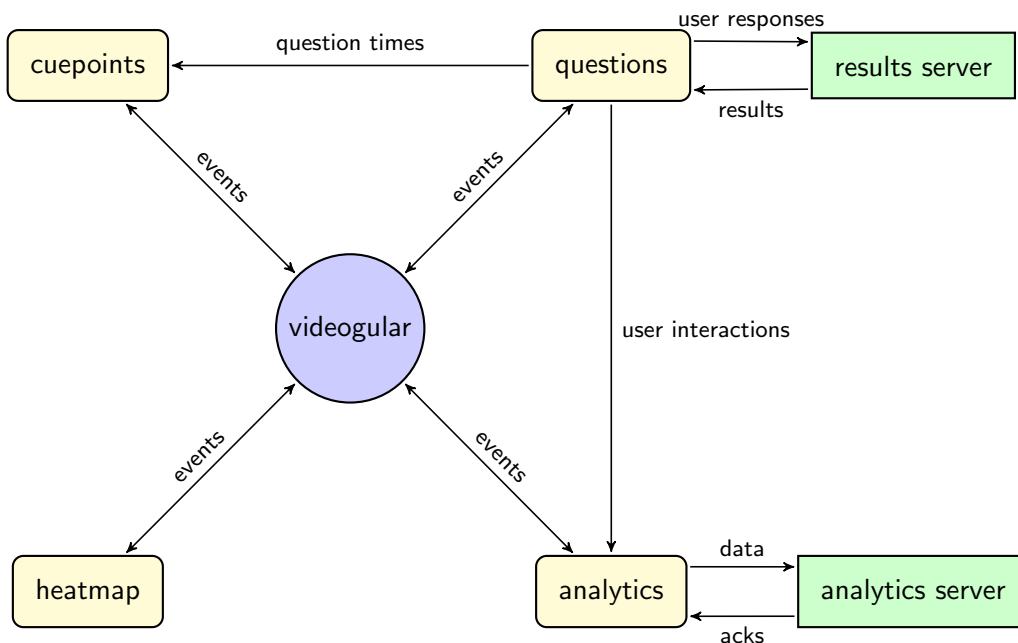


Figure 3.2: System architecture diagram for the project

The results and analytics servers (including the analytics front-end) are examples only, and not themselves deliverables (see [section 3.7](#)).

As the expected behaviour of the plugins depends on the application they are used in, an example site was required to test the other components.

The server components were separated as they do completely separate jobs, and because some deployments may use one but not the other. Similarly, Videogular Analytics is separate from Videogular Questions as it only depends on [Videogular](#). Videogular Cuepoints and Videogular Heatmap are separate components as they can be used in other, completely unrelated projects.

3.7 Deliverables

The following set of deliverables was agreed on in order to meet the requirements:

- Videogular Questions: a plugin for adding polls and questions to videos
 - Source code, under the MIT licence
 - An example proof-of-concept website
- Videogular Cuepoints: a plugin which displays informational marks on the [scrub bar](#) of a video
 - Source code, under the MIT licence
 - Demonstration of usage in Videogular Questions example site
- Videogular Analytics: a plugin for reporting of events within the Videogular player
 - Source code, under the MIT licence
 - An API specification for Videogular Analytics
 - An example site showing basic usage of the analytics data collected by the plugin
- Videogular Heatmap: a plugin which displays heat map information on the [scrub bar](#) of a video
 - Source code, under the MIT licence
 - Demonstration of usage in Videogular Analytics example site
- Authoring tool: a Web application to produce the [Definition File](#) to be used with Videogular Questions
 - Source code, under the MIT licence

3.8 Testing

Software testing is necessary on projects of all sizes, especially on large projects such as this. To ensure quality throughout the project each part of the framework has been thoroughly tested.

In this project, this was done by creating example sites, satisfying [Requirement N10](#). Due to the free-form nature of AngularJS's project structure one of the industry standards, angular-seed⁴, was used. angular-seed is a skeleton with which one can create

⁴<https://github.com/angular/angular-seed>

AngularJS Web apps. It includes two testing frameworks (Karma and Jasmine) already set up. More details about Karma, Jasmine and other tests performed can be found in [chapter 9](#).

Project Management

Chapter

4

Due to the large and complex nature of the project, care was taken to ensure the group's time and effort was managed effectively. This chapter gives details of how the group formed as well as the general positions within the team each member took. Information about the group's working procedure is given, along with details of how the group communicated internally and with project stakeholders.

*“Management is doing things right;
leadership is doing the right things.”
- Peter Drucker in Leadership*

4.1 Formation

Although as a whole the group had never worked together in the past, some existing close working relationships existed due to previous group coursework. As such it was easy to form the group and begin producing impressive results.

4.1.1 Skills Audit

Many of the strengths and weaknesses of the group members were clear from previous work, but a skills audit was undertaken to ensure that the group was competent to complete both the coursework and the project without fault.

To begin with, a requirement analysis was performed to determine exactly what skills were needed. These split into three clear subsections: management and leadership (from the perspective of both the coursework and the project); technical; and communication.

Next, an audit of each team member's skills was performed, the results of which can be found in [appendix B](#). From these an analysis of the group's total skills was determined, and the group's weaknesses could be found. It was found that the group had no clear weaknesses that could found using self-audit. Any members with a weakness were complemented by other members strengths.

4.1.2 Roles and Responsibilities

Throughout the project all members took on a number of technical and management roles. Care was taken to allow for flexibility between and within roles. Roles and responsibilities were assigned dynamically, with members volunteering when they felt they were the best fit.

Christopher Baines

Christopher focused on architectural design and component interaction. Much of his technical work was in-depth and complex in nature. His particular focuses were the Videogular Questions plugin and the analytics server.

Samuel Bennett

Samuel took a leadership role early in the project. He spent much of his time removing technical blockers for other members and acting as a central hub, keeping all members abreast with the work of others. Technically, he used his previous [AngularJS](#) knowledge to begin work on the code bases for the project, as well as working closely with Christopher Baines in determining the general architectural design. Much of his technical work was done on Videogular Questions and the [Definition File](#) format.

Harry Cutts

Harry's main technical contribution was the Videogular Cuepoints plugin, and the accessibility improvements that were contributed to [Videogular](#). He also ensured that the project work was being completed to a good standard, using coding conventions and tools such as JSHint. He also took on part of the role of Scrum Master, ensuring the group conformed to the agile process, and took minutes of group meetings.

Christopher Hewett

Christopher's technical work focused initially on the Videogular Questions plugin (in particular compatibility with mobile devices), and then later on the authoring tool. One of his earlier key themes was completing work on the styling of all front-ends. In the last few weeks prior to Christmas Christopher took on a leadership role in ensuring everything was completed on time.

Maria Lynch

Maria's technical contributions focused on the initial design and implementation of the authoring tool, and the design and implementation of the Videogular Heatmap plugin. Maria took responsibility for organising the coursework deliverables. She managed the report production, and also contributed to the user interface design and accessibility of many of the deliverables.

Generally the group felt that all members completed similar amounts of work over the course of the project, as well as sharing equal responsibility.

4.2 Project Process

The project was run using an agile methodology based on Scrum. This involved having weekly sprints where each person completed a number of tasks, either alone or collaborating with others. Planning meetings were held at the start of each sprint where the issues in the backlog were discussed, prioritised, and given a point value related to how much time it would take to resolve. The issues were then assigned to one or more group members, in order of priority. Each person had a maximum capacity of points for each week. At the end of each sprint, a retrospective was held to discuss the previous sprint and the way in which the project management was being carried out.

This structure meant that the planning and monitoring of the project happened formally on a sprint-by-sprint basis. When required the group adapted their working practices, without being constrained by the Scrum model. For example, when preparing for progress presentations "mini-sprints" were run both for the presentation and for work to be completed in the remainder of the week.

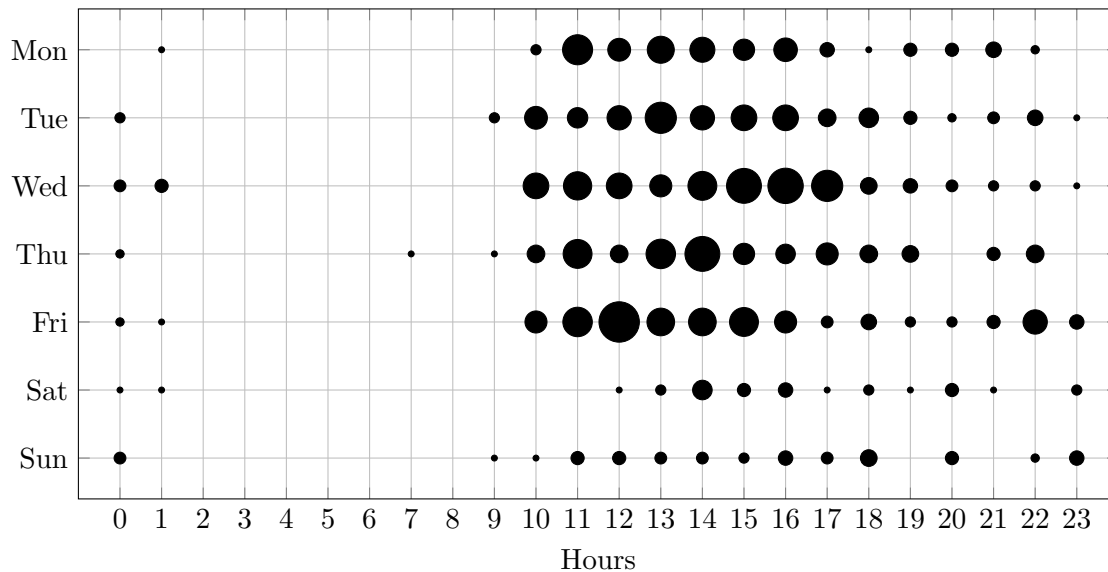


Figure 4.1: A punch card diagram showing the frequency of commits at different hours of the day. The area of each circle is proportional to the number of commits.

At the beginning of the project a project plan was agreed with the client, this can be seen in [Figure F.1](#) and [Figure F.2](#) as a Gantt chart. At the time of the production of this document an updated Gantt chart has been produced and can be seen in [Figure F.3](#) and [Figure F.4](#). As seen in the updated Gantt chart, the authoring tool has been started earlier than planned. During the sprint planning meeting in week 4 spare resources were present and allocated to beginning the mockups for this tool. The Analytics tool was started later than originally planned after its allocated time spend was re-adjusted due to the team gaining further experience with AngularJS.

4.2.1 Statistical analysis of Project Progress

Due to the use of version control and issue tracking, it is easy to produce retrospective statistical analysis.

By tracking commits within the GitHub organisation (over 1000) one can determine when the week's work was normally completed. This data is visualised in [Figure 4.1](#). It is important to remember that work occurs prior to a commit and as such the commit time is an ending marker. Looking at [Figure 4.1](#) one can see the group generally completed most of their work as it would have been done in an industrial setting, during standard office hours, with extra work being done late into the evenings and on weekends.

By tracking closed issues on a week by week basis it is possible to determine when work was completed throughout the project. Although issues are of different sizes, over the sample of 100s of issues a general grasp can be achieved. A graph of this can be seen in [Figure 4.2](#). It shows that work was completed steadily throughout the semester. Week

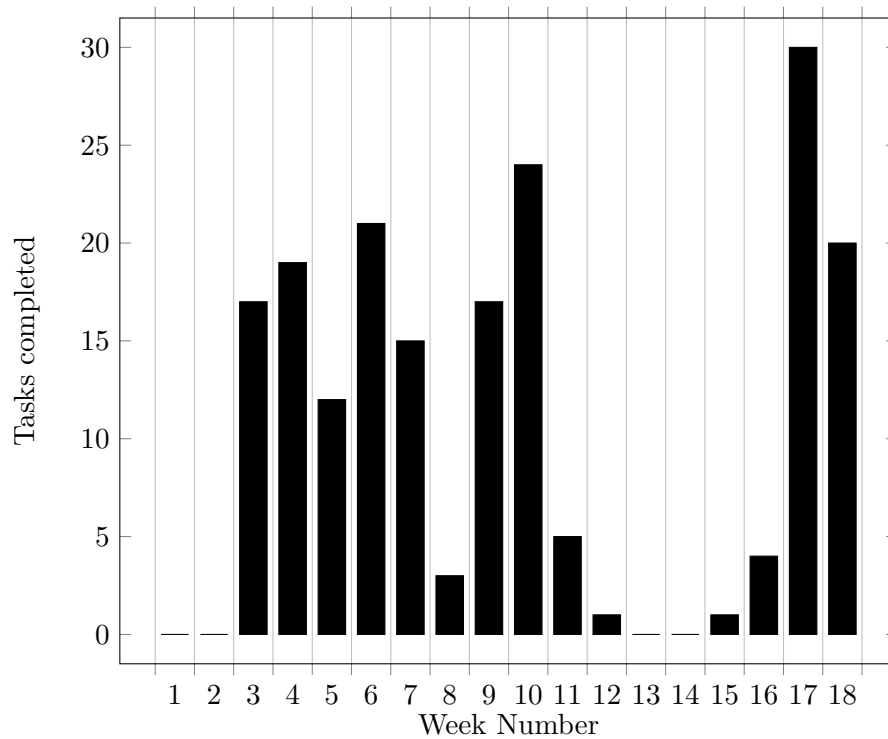


Figure 4.2: A bar chart of the number of GitHub issues being closed in each week of the project.

8's deficiency can be explained by the issues being more time consuming, as well as the second progress presentation and other coursework needing to be completed that week. The dip between weeks 12 to 17 is explained by the Christmas break and examination period.

4.2.2 Collaboration and Issue Management

To allow for collaborative work a GitHub organisation¹ was used. This acted as a central repository not only for code and conventions documents, but also for presentations and reports. Separate repositories for each area of the project are located here for ease of access. Using GitHub made it easier to interact with other open source projects, and all pull requests and releases were directed from here.

GitHub also provided an issue management system on a repository by repository basis. This allowed tracking of conversations around different bugs and enhancements as the project progressed. The group used the Waffle.io² dashboard to manage the issues backlog from a project wide perspective as this would interface directly with GitHub.

¹<https://github.com/soton-ecs-2014-gdp-12/>

²<https://waffle.io>

4.3 Client Management

The project had a reasonably large number of stakeholders who needed to be managed and kept informed. Each stakeholder required different levels of communication via different methods.

The most important stakeholder was the client, Professor Mike Wald. Weekly meetings were organised, minutes of which can be found in [appendix E](#), and emails were used for urgent issues.

Final project sign-off to check for client satisfaction was gathered using a deliverable report. It was delivered prior to the sign-off meeting and then discussed. It can be found in [appendix O](#).

Another important stakeholder was Yunjia Li. He attend the weekly meetings as much as possible so as to ensure his future interests in the framework were protected. He was also contactable via email when necessary.

2fdevs, the Videogular development team, were communicated with via Gitter³, an online chat system integrated with GitHub. They had an ongoing interest with the framework due to its role of advertising the [Videogular](#) project. Their advice and code proved invaluable throughout the project.

Shameem Bajar provided valuable user feedback. She came to meetings and communicated with the group via email.

4.4 Management Problems

Throughout the project a number of management problems were encountered and worked around.

Two weeks before the Christmas break, in week 10, Samuel Bennett suffered a bereavement in the family. As such he was unavailable for the whole of week 10. Upon learning this the group quickly restructured its leadership with Christopher Hewett taking responsibility. The team planned for Samuel to not return before Christmas, although this was not the case.

In the first few sprints the group found that the point value each member was supposed to complete had to be adjusted. Initially it was too low, meaning that all members did not have enough work assigned to them. This meant that people had to be flexible and use their own judgement in prioritising additional tasks for them to complete. In later weeks the point value allocation was increased.

³<https://gitter.im/2fdevs/videogular>

At times the agile methodology was rather restricting. In particular issues were found around the presentation and report deliverables. To combat this the group adapted to “mini-sprints”. This was needed as the presentation dates were halfway through our sprint window (Monday to Sunday). During these “mini-sprints” the week was broken up, with the first “mini-sprint” working towards the presentation and the second doing smaller pieces of work and documentation tasks.

One difficult problem was the inability to plan around other coursework for other modules. Within the group there were 14 additional pieces of coursework with ranging deadlines. Sometimes these deadlines were rather short or the amount of work required unclear. This made the weekly planning process difficult at times, requiring the group to be sympathetic to its members and accommodate change as needed. However, the agile project planning methods meant we were able to be flexible and successful.

Video Player

Specific improvements to [Videogular](#) were made in three areas: accessibility, compatibility and scrub bar extensions. Each area was conducted independently, and the contributions were readily accepted back into the [Videogular](#) project or published as separate plugins. These were made to ensure that the [Videogular](#) player met the accessibility needs and to add required features to the player.

5.1 Introduction

After reviewing the [Videogular](#) project it was found that it was lacking in a number of aspects. Therefore to fulfil requirements such as [Requirement N6](#) the following goals were identified for this part of the project:

- investigate [Videogular](#) to allow for its use within the new version of Synote
- improve the accessibility of [Videogular](#) for users with reduced input options
- ensure that [Videogular](#) was usable on as many devices as possible
- add the ability to mark locations on the [scrub bar](#)
- produce a plugin to allow the display of statistical data along the [scrub bar](#)

5.2 Accessibility

[Videogular](#) has the option to display HTML controls for the video player, replacing the browser's built-in controls. However, when the project began there was no way for the user to access these controls without a mouse. The main problem was that the controls did not identify themselves as interactive elements, and so could not be 'focused' (selected with the tab key). This also prevented them from being picked up by a screen reader.

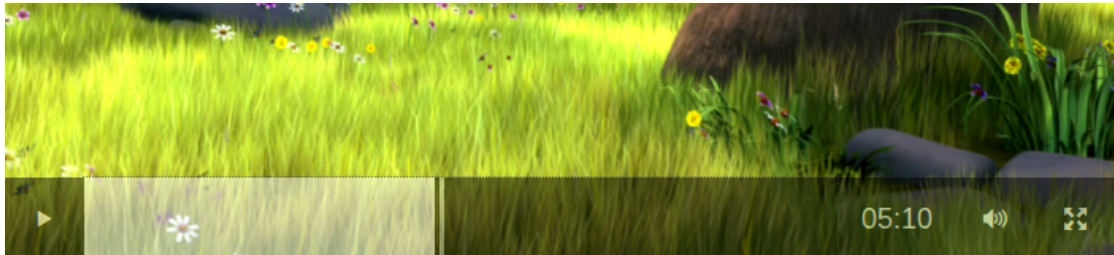
5.2.1 Design and Implementation

Some online video players allow the player as a whole to be focused, and then controlled with various keyboard shortcuts. The problem with this method is that accessibility aids, such as screen readers, will not have information on these controls, and so users of those technologies will require further instruction. It is also not accessible to users of a clicker, who can only press one key and so require individually focusable controls.

Instead, the controls were made individually focusable by using semantic markup where possible, and [Accessible Rich Internet Applications \(ARIA\)](#) attributes otherwise. When a control is in focus, a visual indicator is displayed to highlight it. Buttons can then be pressed using the space bar (Figure [5.1a](#)); the scrub bar can be moved using the left and right arrow keys (Figure [5.1b](#)); and the volume can be adjusted using the up and down arrow keys when the mute button is focused (Figure [5.1c](#)).



(a) The controls with the play button focused. In this state, pressing the space bar plays or pauses the video.



(b) The controls with the scrub bar focused. In this state, pressing the left or right arrow keys skips backwards or forwards in the video.



(c) The controls with the mute button focused. In this state, the volume control appears, pressing the space bar toggles mute, and the up and down arrow keys change the volume.

Figure 5.1: Screenshots of the accessibility improvements made to [Videogular](#)'s HTML controls.

5.2.2 Summary

This solution is not completely clicker-accessible, but could be made so by introducing buttons to skip forwards or backwards, and to raise or lower the volume. It is a significant improvement on the original state, as keyboard users can now use the player. The improvements were submitted to the [Videogular](#) project¹, and are now part of [Videogular](#) 0.6.1.

5.3 Compatibility

One of the main focuses during this part of the project was to make sure the application was compatible with a range of platforms, operating systems and devices

¹<https://github.com/2fdevs/videogular/pull/108>

([Requirement N3](#), [Requirement N4](#)). The first challenge was to ensure that the player and overlay displayed correctly at a variety of resolutions and screen/window sizes.

Apple iPhones caused a problem, as Safari forces the video to be played in full-screen mode. This is a design choice and there is no way to play the video inline, as directives to do this are ignored by the browser.

This meant that the video paused at the correct time but as the video is in the native player the overlay was not visible unless the player was closed. There is no way of notifying the user that this needs to be done. A possible solution involves telling the user to quit out of the video when it is paused so they may answer the question, but this is not very convenient. However, this is not an issue on iPads.

Android phones correctly interpret the inline directive and will play the video properly, so the questions overlay is correctly applied and usable. This has been the primary focus of testing on mobile devices as it represents a very large portion of the market.

The overlay which displays questions appears above the video and obstructs it. Placing the overlay in the middle of the screen requires complex [Cascading Style Sheets \(CSS\)](#) to correctly locate it on small screens. There are still some issues with the pop-up location on some mobile devices where the browser reports an incorrect screen size. Until these devices are compliant with the specification, it is too time consuming to fix each individual case.

All tablets tested, running iOS or Android, display the pop-up correctly. This is because they allow a video to be played inline and have content appear above the video. They also do not need additional [CSS](#) to position the pop-up as they normally have laptop-size screens (>1024 pixels wide) which do not require any video scaling. (The video was 700 pixels wide at its maximum size.)

The best user experience with this application is using it on a desktop with an up-to-date browser. This will definitely work and display as intended. As the screen size of the device gets smaller the user experience is significantly reduced as the video and poll overlay will be much smaller than recommended. To support a number of browsers and operating systems a range of video formats are supplied to the [Videogular](#) plugin so that the browser can pick one that is supported.

5.4 Scrub Bar Extensions

[Videogular](#) has been designed and implemented in such a way as to allow for directive-based plugins to be developed to extend its functionality. When using the [Videogular Controls](#), the native browser controls are replaced with an HTML based interface. All the interface changes are for [Videogular Controls](#), not the native browser controls.

The team behind [Videogular](#), 2fdevs, are actively seeking members of the community to develop new plugins for their system, and so were willing to discuss the plugins and their architectural design.

5.4.1 Videogular Cuepoints

[Videogular](#) Cuepoints is a [Videogular](#) plugin for displaying ‘cuepoints’, marks on the scrub bar which can be positioned at different times. For example, cuepoints could indicate the start of a section in the video, or (in this case) a time when a pop up will appear.

Design and Implementation

Like the rest of Videogular Controls, each cuepoint is represented by an HTML element which is positioned using [CSS](#). This allows users of the plugin to style them with their own stylesheets.

Cuepoint positions are specified in the `config` object which is given as a parameter to cuepoints, and a stylesheet for a custom theme can be specified in the same way.

One problem found during implementation was that the cuepoints cannot be displayed before the video has begun playing. This is due to [Videogular](#) not setting the video length variable until the video plays, and so this variable cannot be used to calculate the positions of the cuepoints until then.

Further Work

In future, the ability to specify a different [CSS](#) class for each cuepoint could very easily be added, allowing visual differentiation of the markers. It would also be useful to make the cuepoints more accessible, for example by providing alternative text for screen readers.

There may also be scope for investigating why the [Videogular](#) player does not set the length variable before it starts playing and if there are any fixes for this. Initial investigation concluded that until the video starts playing it is impossible to determine the length of the video. However, loading the video via a second hidden [HyperText Markup Language version 5 \(HTML5\)](#) element and playing it may give access to the video length.

5.4.2 Videogular Heatmap

[Videogular](#) Heatmap is a [Videogular](#) plugin for giving areas of the scrub bar different colours. In the examples this is used to give a visual representation of the number of times each section of the video has been watched.

Design and Implementation

An early design decision was to use [CSS](#) to colour the sections as this would allow different colours to be specified (see [Requirement N7](#)). This also would allow developers to develop their own colour schemes with meanings if they desired.

Sections and colours can be specified in a `config` object which is given as a parameter when the heat map is used.

The format for defining the configuration parameters was made consistent with Videogular Cuepoints (see [subsection 5.4.1](#)) to improve usability.

The same issue that was encountered with the Cuepoints plugin, which prevents display until the video plays, was a problem for the heat map plugin. This is less of an issue for this plugin as the colouring does not have any context for a user until they have watched the video.

An extra option that was added for improved accessibility was the ability to label each heat map section with the frequency. Extra hidden text was added so that a screen reader user could still access all the data shown in the heat map.

5.5 Summary

Several improvements to [Videogular](#) were made during the course of the project, primarily focussing on accessibility. With these changes the [Videogular](#) player is now accessible to users who only navigate websites using the keyboard. All changes made have been adopted by the [Videogular](#) project. Compatibility with a number platforms and devices was reviewed and the only major problem found was that Safari forces video to be full-screen on iPhones.

Two [Videogular](#) plugins were created to allow the display of markers and statistical data on the [scrub bar](#). Both of these have been designed for use in the example websites, cuepoints for points where questions appear and the heat map to demonstrate how often sections of a video have been watched.

Videogular Questions

Chapter

6

A [Videogular](#) plugin was written to provide the core functionality of the project. It allows for questions to be overlaid on top of the video and handles user interaction. This plugin is called [Videogular Questions](#). It also provides utility functions for the [Definition File](#) messaging passing interface, making it easy to create content.

6.1 Introduction

This plugin was one of the primary focuses of the project, and is an accessible system presenting the user with questions overlaid on a video. A number of features were explicitly requested by the client, such as question types ([Requirement F1](#)) and the ability to let users re-watch content easily if questions were answered incorrectly ([Requirement F2](#)).

6.2 Design

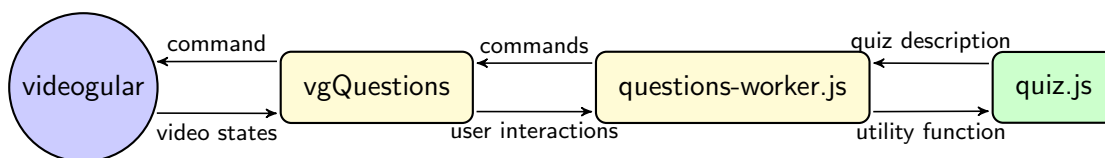


Figure 6.1: The architecture of [Videogular Questions](#)

The core of [Videogular Questions](#) is a [Definition File](#) which defines the question sets. The [Definition File](#) is loaded in a [WebWorker](#) that will interact with [Videogular Questions](#) as shown in [Figure 6.1](#). This uses a well-defined message passing interface to allow easy communication. Most users will not use the message passing interface and will instead use the authoring tool (see [chapter 7](#)) to create a [Definition File](#) that is run by the [WebWorker](#). This makes the plugin exceptionally flexible.

One of the issues discussed for this section was accessibility. It was decided that [Cascading Style Sheets \(CSS\)](#) should be used for the styling to allow users to alter the styling to suit their needs. Keyboard accessibility would be an additional focus.

6.3 Implementation

The implementation of the questions overlay was split into three sections: first, representing the question sets internally; next, displaying the questions within the user interface; and finally, implementing the back-end communication between the question representations and the video.

6.3.1 Annotations

One of the main issues to address early on was how to represent the questions. The [Question and Test Interoperability \(QTI\)](#) specification was investigated, but was found to be complicated and made implementing accessibility features difficult (see [section 3.1](#)).

It was decided to design a new format for representing the data and logic that is used for a particular application of [Videogular Questions](#).

This new format separates the front end of the library (responsible for interacting with the [Document Object Model \(DOM\)](#)) from the data and logic describing the questions, by means of a message passing interface. This is achieved in a rigorous way in the browser by using a [WebWorker](#), a sandboxed thread that runs independently of other scripts, and without access to the [DOM](#) or JavaScript functions which could compromise security. More information about their use within this project can be found in [subsection 6.3.3](#).

Using JavaScript rather than a pure data representation (e.g. JSON or XML) allows the inclusion of logic to decide on an action to take (e.g. a question to display, or video location to skip to). This makes it extremely flexible and concise, and the isolation provided by the [WebWorker](#) mitigates many security concerns with having an application that executes data given as input as code.

Every item in an [annotation](#) can have an action and condition function. The action function is called when a item finishes, and is aware of the state of the [annotations](#), allowing it to make decisions and affect the state of the video accordingly. For example, in [Listing K.1](#) this is used to show the “skip back” question only if the previous question is answered incorrectly.

The condition function is called to determine if the respective item will show. By default, when an [annotation](#) is shown, each item is shown in sequence. However, if an item has a condition function the item only shown if the condition function returns [true](#). If the condition function returns [false](#), the item is skipped. This functionality is used in [Listing K.1](#) to have the video skip back on the request of the user.

An early decision was to define the difference between a poll and a quiz question. The decision made was that a poll is a type of quiz question that does not have a correct answer.

In the code this is captured in the `record_response` attribute. If the `record_response` attribute is set to [true](#), the question becomes a poll and sends the results to a server. If the `correct_answer` attribute is set then the question has a correct answer which will be shown to the user, if requested. This allows a user to possible have a poll type question that stores the data collected and also has a right answer. This provides maximum flexibility, and the underlying code doesn’t distinguish between a quiz or a poll.

Basic question types (single choice, multiple choice and scale questions) were implemented first. A variety of visualisations were implemented including check boxes, radio buttons and sliding scales. Validation was needed to ensure that the specified minimum and maximum limits were followed.

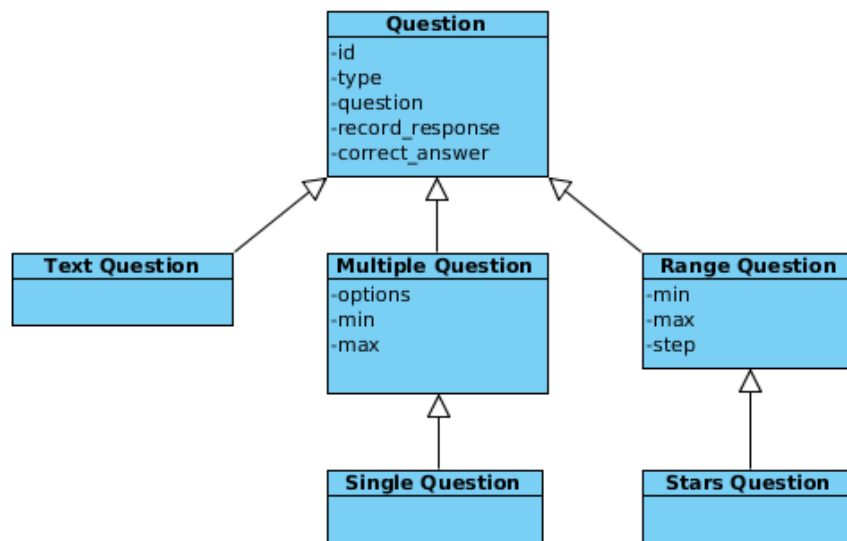


Figure 6.2: A class diagram representing what attributes each question type needs

In reviewing the types of questions to support a class diagram was produced ([Figure 6.2](#)) to describe the type of data that the questions would have. Once the additional question types of textual and range were implemented [Requirement F1](#) was satisfied.

As seen in [Figure 6.2](#) Single Question is a sub type of Multiple Question where min and max (the minimum and maximum selected number of answers) is 1. Similarly, the stars question is similar to the range question where the `step` value is 1. These decisions were made to simplify the code to make a consistent interface easier to produce.

By having a standardised [Definition File](#) that uses JavaScript functions, it was possible to write template functions that could be used by the authoring tool.

6.3.2 Front-End and User Interface

The appearance of the overlay depends on the question type to be shown. The layout of these different types was carefully considered for accessibility and ease of use. Mockups were made (see [appendix I](#)) and user studies were done in collaboration with a third year project student (see [appendix J](#)).

A set of example [CSS](#) files are supplied with the project that layout the questions according to the feedback received. Developers using any part of the project can use these styles as is, or modify/develop their own. The styles provided had a high contrast to meet the [Web Content Accessibility Guidelines \(WCAG\)](#) 2.0 colour contrast standards. Keyboard accessibility was ensured throughout.

The final user interface produced can be viewed in [Figure A.1](#), [Figure A.3](#), [Figure A.4](#), [Figure A.5](#), [Figure A.6](#) and [Figure A.7](#).

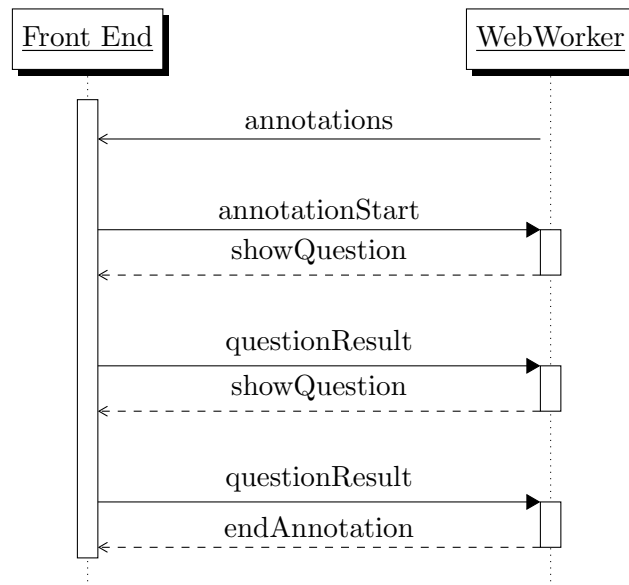


Figure 6.3: Sequence diagram showing interactions between the front end and WebWorker

6.3.3 Back-End - WebWorkers

WebWorkers give a good way of running (sandboxed) background scripts that are computationally intensive. They are a way of multithreading - allowing multiple scripts to run simultaneously, avoiding the problem of unresponsive pages due to long running scripts. This is done by using message passing.

For example, when an answer is submitted by the user this would send a message to the **WebWorker** containing the answer. The **WebWorker** can then process this information without affecting the responsiveness of the page. Once the processing is complete the **WebWorker** can send a message back to the page to tell it what to do next.

Figure 6.3 illustrates messages that could pass between the **WebWorker**, and the front end when the Listing K.1 is used. Initially, the worker sends an `annotationStart` message which contains the times which **annotations** will occur.

When the first time point is reached, the front end sends a `annotationStart` message to the **WebWorker** containing the id of that **annotation**. The worker then replies with a `showQuestion` message containing the contents of the first question. Note that here the functions are not sent to the front end and it is just the JSON representation of the question is sent.

When the user responds to the question, that response is sent to the **WebWorker** in a `questionResult` message. In this case the user responded incorrectly, therefore when the **WebWorker** evaluated the condition on the second question it evaluated to `false`. This meant that the **WebWorker** replied with another `showQuestion` message.

Once the user responds to the second question, the response is again sent to the [WebWorker](#). In this case, this was the final question in the [annotation](#), so the [WebWorker](#) responds with an `endAnnotation` message.

6.4 Summary

An internal representation of the questions is given by the [Definition File](#) that is used to display the questions to the user in the overlays. This allows all aspects of the questions presented to the user to be customised. This [Definition File](#) can be manually written or it can be created by an authoring tool ([chapter 7](#)).

[CSS](#) was used to keep the appearance of the questions consistent and accessible but this also allows a developer to quickly change how the questions are shown with little technical knowledge of the system helping to fulfil [Requirement N7](#).

[WebWorkers](#) are used to communicate with the front end to provide the necessary messages for the expected actions to occur. By abstracting this to another thread utilisation of additional processing has been enabled and it is possible to ensure that the user has a better experience since the page is not as likely to freeze due to the additional thread sharing some of the processing. In addition, separating the front-end and back-end and only communicating via a message passing system provides low coupling which will allow customising one without affecting the other.

Authoring Tool

The authoring tool chapter details the motivation for creating a tool that is able to produce the [Definition File](#) used by the [Videogular Questions](#) plugin. One of the key features was that the tool should be accessible and is able to describe the wide range of features the [Videogular Questions](#) plugin supports.

7.1 Introduction

An accessible [Authoring Tool](#) was required to allow users to create their own question sets. Within the authoring tool users can create polls and quizzes and overlay them at chosen locations in a video. This avoided the necessity for users of the framework to learn the JavaScript quiz format, thus significantly reducing the technical barrier to entry.

One key consideration during the implementation of the authoring tool was accessibility. The [Web Content Accessibility Guidelines \(WCAG\)](#) 2.0 guidelines were kept in mind at all times and adhered to as much as possible.

Care has been taken to ensure that the authoring tool can be easily used within other AngularJS projects, and thorough documentation has been provided to ease this process.

7.2 Design

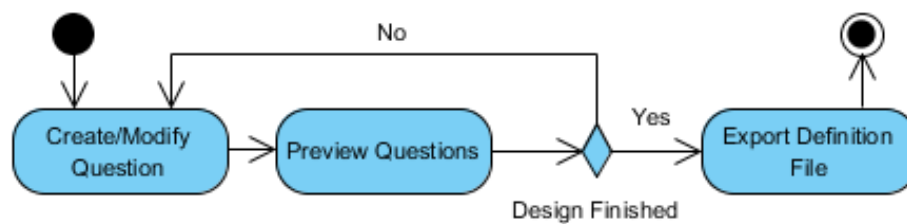


Figure 7.1: Image showing a state diagram of how the authoring tool is used.

Within the authoring tool users have the ability to create questions sets within a video. There could be any number of questions in a set and these could be of many different types. A range of options are given to modify the functionality of each question to allow the user to create exactly what they need.

At any point the user can preview their creation on the inbuilt video player. Once they are satisfied with the quiz they can export it for use with [Videogular Questions](#) externally from the tool.

It was decided to use Bootstrap¹ to give a consistent appearance to all of the user interfaces components within the project. This is a HTML, [Cascading Style Sheets \(CSS\)](#), and JavaScript framework for developing webpages.

As an Angular framework is used, any additional functionality is required to be written as directives and controllers. UIBootstrap² provides some Bootstrap components written in [AngularJS](#). This provided some of the components required.

¹<https://github.com/twbs/bootstrap/>

²<http://angular-ui.github.io/bootstrap/>

Wireframes were created using this style so that they could easily be directly coded as static HTML to enable a demonstration to be provided quickly to the client and to allow for feedback from users and the client. They can be found in [appendix H](#).

Two different approaches were suggested for showing the advanced options for the video. A pop up approach hid these from general view to avoid confusing users with lots of options that may be unnecessary. The other approach was to use the accordion structure. This would be collapsed by default so the options would still be hidden. This was chosen as it was more consistent with the rest of the tool.

7.3 Implementation

Once the wireframes were drawn up they could be turned into static HTML. Elements of this webpage were then incrementally given the functionality they required. The main challenge was ascertaining how the functionality fitted in with the [AngularJS](#) framework.

Each section of the page had to have its own directive so that independent behaviours could be achieved. This kept the HTML pages short and readable as the behaviour was dealt with by the JavaScript and the styling dealt with by the [CSS](#).

Angular allowed the dynamic nature of the data to define the appearance of the page. Sections could be set to appear only when certain attributes were set using `ng-switch` statements. This meant one area of the page could have different content decided by selections made elsewhere on the page.

7.3.1 Data Bubbling

Within the authoring tool the ability to use two way binding and watches with AngularJS was used. This allows templating of a number of reusable elements such as the time picker and means there is no necessity to rewrite the class each time it is used. This reduced the time taken to produce the authoring tool.

The root controller holds the global state of the authoring tool. This global state is updated by the child controllers that each own specific parts of the global model. This bubbling of data through the controllers is the key to how the authoring tool was built.

[Figure 7.2](#) shows the general structure of the controllers. There are a large number of controllers and levels to encapsulate the different aspects of the question sets being created. In this diagram controllers A and B will bubble any modified model data up to the root controller. If model data in controller B1 or B2 changes, their new model will bubble up to controller B. Controller B will then bubble its model combined with the new model data to the root controller.

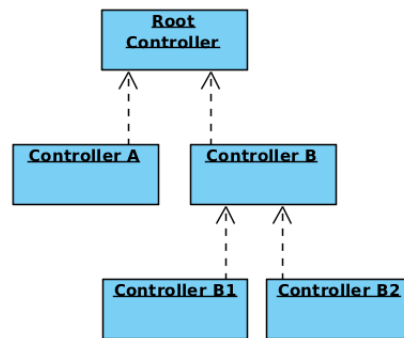


Figure 7.2: Bubbling of model data up controllers

This bubbling technique works for any number of or levels of children, provided the model data is correctly bubbled up through each layer. It also makes adding new controllers at any point simple as only the bubbling of the data up the levels needs to be handled.

Each controller may have one or more children who may or may not hold model data.

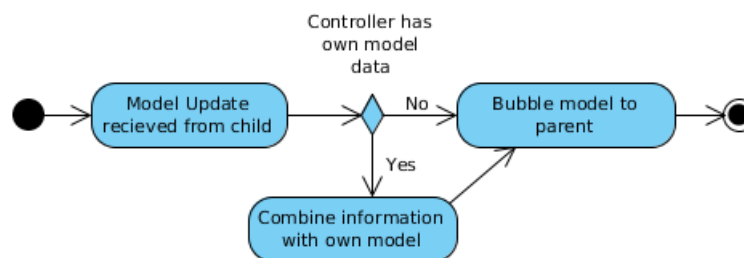


Figure 7.3: State diagram illustrating decisions when bubbling data to root

Figure 7.3 demonstrates the decision making process to determine how the data is bubbled by the controller. Any controller that does not hold important model data will only bubble any information sent to it to the root controller. Any controller that has a model will also bubble its own model and model data received upwards. Therefore the global model is slowly accumulated through the data that has been bubbled up. When it reaches the root controller the global model is built and is saved for later use.

A controller holding model data stores the state of the controller which is represented by a HTML view. Each view is bound to its specific model using two way data binding. This means when the view is changed by the user the model will be automatically updated.

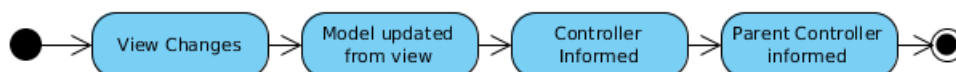


Figure 7.4: Client-centric data binding updating the model in the parent controller by data bubbling illustrated in an state diagram

Figure 7.4 shows the process of the changing the view and the parent updating accordingly. Each time the model is updated the controller runs any appropriate action to change the display of the view then sends the updated model to its parent. This

change is bubbled upwards at each level until it reaches the root controller which stores the new data in its global model.

7.3.2 Exporting of Data

As the user is selecting the options for their question sets, the data will be bubbled up as described in [subsection 7.3.1](#). Once they have finished they will be able to press the [Export](#) button. When this is pressed the global model is reviewed, and the corresponding [Definition File](#) is generated.

First the global options are processed and the basic file is created. This uses a default template which is stored in the authoring tool. The template used is shown in [Listing 7.1](#). This has two sections that are replaced with content that are generated in a later stage. These are `WORKER_URL` and `ANNOTATION_DATA`. `WORKER_URL` is replaced by the URL. This may need to be modified if the [Videogular Questions](#) plugin has been modified to move the [WebWorker](#) location. `ANNOTATION_DATA` is replaced with the generated question data that is generated below.

```
1  /* jshint worker: true */
2  "use strict";
3
4  importScripts("WORKER_URL");
5
6  /* global loadAnnotations */
7  loadAnnotations(ANNOTATION_DATA);
```

Listing 7.1: Base template for authoring tool [Definition File](#) generation

Once the template has been created, each question set is processed. This involves taking all the settings for the question set, storing it the question set format, and then processing and generating JavaScript for each question in the question set.

When generating the JavaScript for each question the question type is used to generate a set of attributes specific to that question type. The attributes generated are as defined in the class diagram [Figure 6.2](#) and these are added to the question.

For some questions, there will be a number of different additional questions added to the question set. One example is when the author has selected that a user may return to a specific point if they fail a question. This will add a question asking the user if they wish to review the section of the video containing the answer to the question. This template is shown in [Listing K.2](#) and has the variables `QUESTION_ID` and `TIME` which will be replaced with the data from the question the user has enabled the option on.

Once an individual question has been generated the next question in the question set will be processed.

Once all of the questions in a question set has been processed it will continue onto the next question set.

Some options allow you to add results at the end of the video. These settings are stored up and once the all question sets have been processed this is added onto the end of the `ANNOTATION_DATA` string.

Once all of these options have been calculated the final template is produced. The template string is processed and content is inserted to generate the finished [Definition File](#). To allow the user to download the file easily a `blob` URI is constructed with the code in [Listing 7.2](#). Without using a data URL it would require the user to copy the text generated and save it. This was not a user friendly option so there was investigation into possible other uses. Offering a file for download was problematic as JavaScript does not have any write permission for files due to its client side nature. A data `blob` URL provides an acceptable solution by mocking up a file URL. This may encounter issues with larger files due to the max length permitted in a URL but these issues have not yet been seen.

```
1 var url = "data:text/json;charset=utf8," + encodeURIComponent(data);  
2 window.open(url, "_blank");  
3 window.focus();
```

Listing 7.2: The final [Definition File](#) is offered for downloading using a data `blob` URL

7.3.3 Preview Tool

The preview tool works in a similar way to the export tool ([subsection 7.3.2](#)) by generating the [Definition File](#) and creating the `blob` URI. Once this has been done this is passed to the [Videogular Questions](#) plugin embedded in the authoring tool, which loads up the newly created set of questions.

This allows for immediate feedback when creating the set of questions and makes it easier to create a set for non-technical users ([Requirement N9](#)). Since the [Videogular Questions](#) plugin has been embedded it is sure that this will react the same when added to a website using the plugin.

7.4 Summary

The tool that has been produced allows a user to create the [Definition File](#) in the correct format that can be used [Videogular Questions](#) to display user created question sets.

[Requirement N6](#) outlines the importance of accessibility so the authoring tool has been designed to be as accessible as possible and [section M.4](#) details the conformance to the

WCAG standards. In [subsection 9.6.1](#) a number of testing methods are applied to the authoring tool to demonstrate that it is accessible and outlined any further issues.

[Figure A.13](#) is an image showing the final user interface. The colour scheme was chosen to be high contrasting to fulfil [Requirement N7](#). In addition the [CSS](#) used to set this colour scheme is in a single place and is easy to change as to meet [Requirement N7](#).

To further decrease the barrier to entry ([Requirement N9](#))the [Videogular](#) and [Videogular Questions](#) plugin were used to allow a preview of the question set they have created. One of the useful side effects of [Requirement N2](#) has meant that it has been easy to integrate [Videogular Questions](#) into the authoring tool.

Some more work may needed to investigate how large a [Definition File](#) may be produced when using the data [blob](#) URI approach to allow the user to download the file. While no problems have occurred during testing, [subsection 7.3.2](#) identifies possible issues that may occur for exceptionally large question sets.

Analytics

Being able to track how the application is being used will allow further improvements in the user experience and is a key requirement for the client. This will also enable research to be performed once use statistics have been gathered. To facilitate research applications, the ability to add events to the plugin has been made exceptionally simple.

“I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.”

- Sherlock Holmes, A character written by Sir Arthur Conan Doyle

8.1 Introduction

One of the main requirements states that video and question events should be emitted from the Videogular player ([Requirement F3](#)). It was decided to make a new plugin to handle analytics rather than putting analytics events directly into the [Videogular Questions](#) plugin.

This plugin will monitor how the user uses the [Videogular](#) video player and the [Videogular Questions](#) plugin and store events related to this. If a server is configured this will send the users data to a server where it can be aggregated and processed.

8.2 Design

The reason for designing the plugin to be separate from the main [Videogular Questions](#) plugin is that not all users of the core functionality will also want to perform analytics. This means that the overhead of collecting the data is removed for users who are not interested. This can be important on mobile devices that have less processing power available. In addition, implementing this as a separate plugin means that it is not dependant on any of the other plugins and therefore fulfils [Requirement N2](#).

Since the plugin is not able to be directly demonstrated it will be used with an external server to capture data and illustrate its operation. The aggregates the data and stores it in some format. It is configured to post data via [Representational State Transfer \(ReST\)](#) calls ([Requirement N11](#)) to a server defined in the analytics configuration file.

There is a published [Application Programming Interface \(API\)](#) to be used with the analytics plugin that details all events published from the plugin. A well documented [API](#) is important ([Requirement N10](#)) so that developers can easily create a service that collects and uses the data.

In the event that the server is unable to be contacted it should queue the results until it comes back online. Once a send request fails the plugin pauses sending events and queues them. It tries sending all queued events at intervals until the server comes back online or the application is closed. If the application is closed before the server comes back online the data is lost.

8.3 Implementation

To ensure that [Videogular Questions](#) does not need [Videogular](#) Analytics to work and is still able to communicate it was decided to use the publish/subscribe model. This allows messages to be sent to specific channels. Any modules listening to the channel

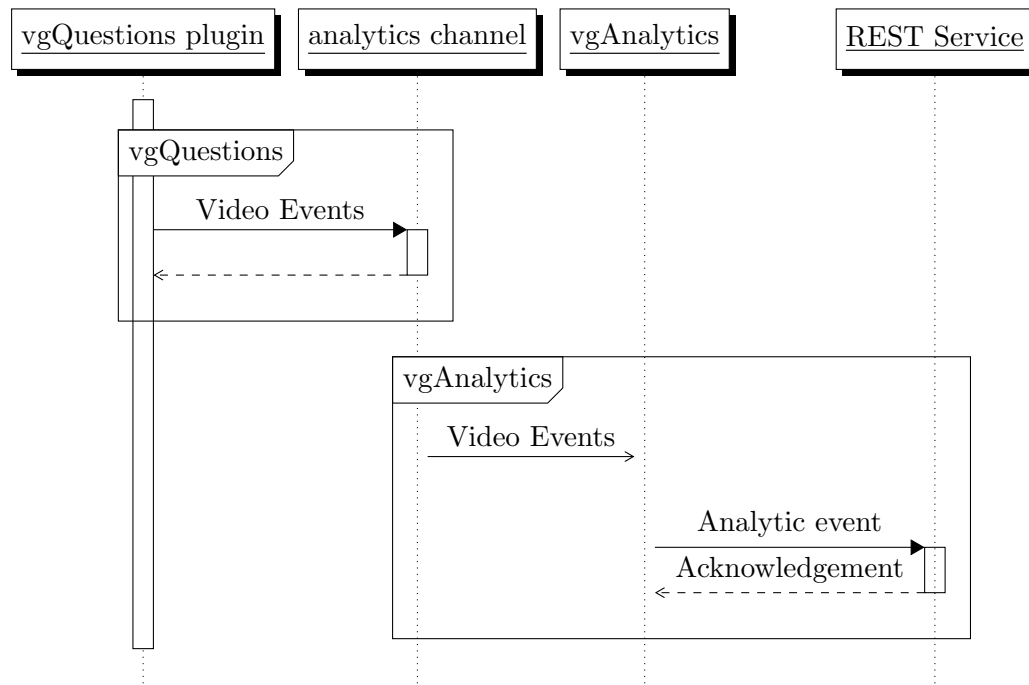


Figure 8.1: Sequence diagram showing the process of events being emitted by **Videogular Questions** (vgQuestions) plugin and sent to the REST service by the **Videogular Analytics** (vgAnalytics) plugin

receive all messages sent to the channel. There is a specific Angular broadcast system which is used to facilitate creation of this publish/subscribe model.

In [Listing 8.1](#) the `analytics` channel is broadcasted an event with the name `show_question` with the content of the data variable. This is used in **Videogular Questions** to detail how people are using the overlay.

```
1 $rootScope.$broadcast('analytics','show_question', data);
```

Listing 8.1: AngularJS demonstrating the message passing interface used in the Analytics plugin

The **Videogular Analytics** plugin listens to the `analytics channel` and receives all messages published to this channel. Therefore **Videogular Questions** can publish information which will be received by the **Videogular Analytics** plugin. If it has not been instantiated then these messages will not cause any errors which is one of the important factors in picking a communication method between the various plugins. The sequence diagram in [Figure 8.1](#) shows an illustration of the messages sent between each of the different modules. Here you can see two distinct parts of the system joined by the `analytics channel`.

By using the publish/subscribe model the plugins are designed so that they will work together when both are used but have no dependencies on each other. This allows any of the plugins to be used together or separately as the user wishes.

In addition to listening to the messages sent by the [Videogular Questions](#) plugin the [Videogular](#) Analytics plugin also listens to a number of [Videogular](#) video events. This is to track information such as when the user pauses, continues and stops the video. These hooks are placed onto the [Videogular](#) object and then are emitted on the `analytics channel`.

By ensuring that all events that will be sent to the server are published to the `analytics channel` further plugins will be easier to introduce. If there is a wish to process the statistics generated on the client this channel can be listened to without interacting with the [Videogular](#) Analytics plugin. This keeps the plugin a very separate unit ([Requirement N2](#)) while allowing easy access to the data that it uses.

The deliverable [API](#) is listed in [Table 8.1](#) and describes all messages that are sent from the [Videogular Questions](#) and [Videogular](#) Analytics plugin to the configured server.

Table 8.1: API of the emitted analytics events and their data payload

Event name	Emitted when	Expected Payload
show_question	a question is shown to the user	All of the associated question data
end_question	the annotation being shown has finished	None
show_results	there are results that will be shown	The results data being shown
submitted_question	the user submits a question	The chosen question response
skipped_question	the user skips a question	None
continue_question	a results page is closed by pressing the continue button	None
play	the video starts to play	The time the video plays from
pause	the video is paused	The time the video pauses at
stop	the video is stopped	The time the video was stopped at

8.4 Summary

The [Videogular](#) Analytics plugin has been designed to interact with the [Videogular Questions](#) plugin only by communicating via the publish/subscribe model. This means that it is simple to access the events with another plugin or include additional events to be sent to the server without modification of this plugin.

A well documented analytics [API](#) has been created (see [Table 8.1](#)) to ensure that the events sent are able to be understood and used easily in another application. The ability

to add more events makes this a dynamic plugin which can be easily adapted for specific research uses.

Testing

Software testing is necessary on projects of all sizes, especially on large projects such as this. To ensure quality throughout the project each part of the framework has been thoroughly tested. This chapter details different testing methodologies along with the results of the tests.

9.1 Introduction

During development number of testing methods were used for each section of the project to reflect the different types of module. All sections that have requirements that directly specify features such as [Requirement N6](#) will have user acceptance tests performed. These tests directly form part of the deliverables report ([appendix O](#)).

A number of sections have a deliverable sign-off table. While working in an agile method a working program that could be presented to the client at any time was kept. This was usually done during the client/supervisor meetings. During final tests number deliverable sign-off tests were created that were used to prove to the client that the software worked correctly. These were presented in the deliverable report (see [appendix O](#)) that was presented to the client and signed off.

These deliverable sign-off tests were used as manual regression tests. These were used during the final changes and bug fixes to ensure the functionality that the client wished were still present and working correctly.

9.2 Tools

A number and range of different tools have been used during testing to ensure that the quality of the code is to the high standard the client required. For each repository the types of tests that are applicable were decided and can be implemented to properly test the application.

9.2.1 Unit Testing Plan

Where possible unit test style tools have been used to automate repeatable types of tests. Anyone working with a repository that has unit tests should follow the following policies:

- **Running unit tests after checking out new code** - This is to ensure that all code checked out is currently working. If it does not they are encouraged to look into fixing the problem themselves or determining which commit broke the build and contact the committer to find the reason for this.
- **Create unit tests for new functionality** - The unit tests form part of the continuous testing plan with the aim to ensure all code on master branches runs properly. Adding unit tests for new features helps ensure this as others will be able to run them to check your new features are working correctly after other code modifications have been made.

- **Run unit tests before committing new code** - As a final check before submitting new code, all unit tests should be run on the repository. This acts as a final confirmation to ensure that no new code has broken old code. This forms part of the regression testing. Any older code that has been broken must be reviewed and fixed to ensure that the master branch runs properly.
- **Fix broken unit tests** - When a unit test is identified that it is not working as intended it should be reviewed to see if it is still applicable and ensure that it is modified to work if so. There may be multiple reasons why a unit test has been broken so it may be needed to contact the person who originally broke the unit test to find out what was changed and why.

The tools that were used in the testing sections below are described fully in the following subsections.

9.2.2 JSHint

JSHint is a tool for static code analysis of JavaScript, which can detect many instances of bad coding practice in source files before they cause errors. Included in each of the repositories containing JavaScript is a configuration file for JSHint, which enforces use of ECMAScript 5 Strict Mode, and prohibits use of undeclared variables or type coercion. JSHint was run at regularly during the project and any issues it highlighted were fixed. For repositories that had JSHint set up for use it was recommended to run this before committing.

9.2.3 Karma

Karma allows tests to be run in browsers (or headlessly) in an automated manner. It then also aggregates and displays the results making it easy to see if the tests have passed or failed.

Karma is included within Angular seed (see [section 3.8](#)), which means it was available from the start. The inclusion in Angular seed was one of the reasons it was used as it is an industry standard for AngularJS.

9.2.4 Jasmine

Jasmine is a testing framework for JavaScript. It has simple syntax and works well with Karma.

Jasmine was also included in Angular seed (see [section 3.8](#)) and therefore already set up for use.

This has been used as the unit test framework for AngularJS code within the project.

9.2.5 Python unittest Module

Python has included in its standard packages, a module called `unittest`. This module is designed to be similar to the JUnit testing framework for Java and this has been used for all python based projects as a unit test framework.

9.2.6 Locust

Locust¹ is a distributed load testing tool that allows definition of a site access pattern. By designing a site access pattern you specify how often sites are visited relative to others and therefore can tailor the access patterns to your likely use case.

This tool has been used as it is able to give percentile data to see statistics on how fast all requests are fulfilled. Many other similar tools only give an average request time which will cause problems when some users are kept waiting for a much longer time. Having access to a full set of statistics allows the running of performance checks and improvement of features so no users are left waiting.

9.3 Videogular Questions Example

The Videogular Questions Example site was written as testing and documentation for how Videogular Questions, Videogular Cuepoints and Example results server could be used. This helps to fulfil [Requirement N10](#).

9.3.1 Videogular Questions

There are several tests written for the [Videogular Questions](#) plugin. These use Jasmine, and can be run in an automated way using Karma.

These tests exercise the [WebWorker](#) component of the [Videogular Questions](#) plugin over the message passing interface. A script is used, which contains messages to send, and expected responses. A test passes if the script runs as expected with no missing or extra messages.

The examples created in the Videogular Questions Example repository use individual or collections of features. These were used in the development of these features, and in the

¹<http://locust.io/>

demonstration of these features to the client. These examples complement the Jasmine tests, as they exercise the display elements of the plugin.

There was no user experience testing done directly as part of the project, as the client had reservations about user experience testing due to the toolkit style nature of the project. Shameem performed some user experience testing for the Videogular Questions Example site and these were discussed in a meeting with both Shameem and the client [subsection E.7.1](#). The consensus was that while the results were informative, they were outside the scope of this project.

9.3.2 Videogular Cuepoints

[Videogular](#) Cuepoints was integrated into the example site, and configured to mark the times at which pop ups were displayed. The issue where marks could only be displayed once the video began to play (as described in [subsection 5.4.1](#)) was the only problem encountered.

9.3.3 Example Results Server

For the example results server there was a small set of functions to test with a small number of possible inputs and a well defined set of responses. This is, therefore, well suited to individual unit tests.

Flask² is a python web application framework that allows you to create web applications with simple routing patterns in python.

The Flask library had a test client and recommended test skeleton³ which was made use of to run the unit tests. This used the `unittest` standard python library which meant it was easy to test with but also allowed calling of methods by simulating HTTP requests.

The testing was split into 6 areas to test the main components of the application: cross-origin resource sharing, routing, database setup, voting, getting results and load testing.

Cross-Origin Resource Sharing Tests

One of the main requirements by the client was that these units should be able to be accessed via [Representational State Transfer \(ReST\)](#) calls (see [Requirement N11](#)). In addition the requirement stated that there should be no reliance that these servers are on the same host. To ensure that these [ReST](#) calls will not fail Cross-Origin Resource Sharing (CORS) headers were implemented as discussed in [section 3.5](#).

²<http://flask.pocoo.org/>

³<http://flask.pocoo.org/docs/0.10/testing/>

This checks to ensure that the CORS header is correctly sent in the HTTP reply. If this is not set the web browser will likely reject the loading of the page and the application will fail.

In addition, it ensures that the response is the one expected and not an error state to ensure that the web application is also sending the correct content.

Routing Tests

The routing tests are to ensure that the application correctly starts and is available.

If this fails to load up the testing URL which returns "Hello World!" then it is unlikely to be able to perform some of the more complex functions.

Database Setup Tests

Before the application can be used it needs have its database set up. This is performed by accessing '/setup'. This set of unit tests test setting up a database and ensure that if this URL is visited twice, it successfully detects that the database is already set up and does not recreate it.

This is important to ensure that the database is set up correctly and that when setup is visited again data in the database is not lost.

Voting Tests

These tests try a number of different ways to vote by sending a number of different formats of invalid and valid data to check if the application correctly deals with the data. All return codes and responses are checked to ensure no invalid vote is accepted or valid one rejected.

Getting Results Tests

A number of valid votes are constructed and then sent into the system. Then these are attempted to be retrieved. The returned values are checked to ensure that they have not been corrupted. This submits one and multiple votes to ensure that all votes are correctly collated and returned.

If the ability to vote does not work then this will fail as it relies on being able to put votes into the system.

Table 9.1: Time taken for each request to complete in milliseconds while stress testing the example results server

Name	Requests	Percentile							
		50%	75%	80%	90%	95%	98%	99%	100%
GET /	535	5	50	67	110	170	240	360	601
GET /results/*/	1599	6	26	46	100	136	233	326	326
POST /vote	1129	120	150	170	210	250	360	460	638
Total / Average	3263	32	110	120	160	210	270	400	638

Load Testing

For the load testing Locust was used and three access patterns were chosen to test the example results server:

- Visiting the root page - This is to determine that the webserver is still responding to basic HTTP requests
- Voting - This is to test that the voting functionality is still accessible
- Requesting data - This is to test that the server is still able to return data

These three URLs were visited repeatedly by the user clients over a period of time.

Table 9.1 shows the time taken for a response to be handled and returned.

Here you can see that the time taken for result requests takes less than 6 milliseconds 50% of the time and all requests were dealt with in at most 326 milliseconds. The voting ReST point takes slightly longer as it needs to store data to the sqlite database but still took at most 638 milliseconds. All these tests were performed when 50 Locust users were using the website continuously and therefore it was agreed that the performance should scale. This is because one user will only ever make one vote and one results request per question and this tested for 50 users continuously making these requests.

Although this is only an example this ties in with Requirement N12 that the results server should perform normally when a lecture room full of people is using it. The testing shows that the application is able to cope with 50 people continuously voting and therefore should be able to cope with a much less strenuous 300 people casting votes over a period of several minutes.

There is also scope to move the back-end database to something that is purely in memory to further increase the speed, as the major factor will be writing the data back to the hard disk.

9.3.4 Accessibility

The Videogular Questions Example was mostly accessible by the WCAG 2.0 standards (see Table M.1). However the lack of alternatives for the video and the cuepoints using only colour may cause issues for some users.

The lack of captions or signed versions of the videos was down to the videos used so not was an issue with the page overall as it is a proof-of-concept.

A textual alternative was not provided because the idea is to integrate this into Synote which provides a transcript (see section 2.4). Thus, this feature would be duplicated.

The cuepoints are only conveyed through the changing of colour on the scrub bar. In future extra behaviour could be added to these to make them accessible through other means.

As a proof-of-concept this site showed that it is possible to create an accessible questions overlay when combined with Synote.

9.3.5 Deliverable Sign-Off Tests

These tests in Table 9.2 were presented to the client as the deliverable items for the Videogular Questions and Videogular Cuepoint plugins.

These tests were ran on the Videogular Questions Example website. This had examples of all types of questions and therefore was used as a deliverable for the client.

Table 9.2: vgQuestions Example Deliverable sign-off tests

Detail of test	Pass
Single Question provides radio buttons to select at most one answer.	✓
Question Multiple provides checkboxes to select one or more results. The min/max values should be shown and <input type="button" value="Submit"/> should be disabled if these requirements are not met.	✓
Questions Star should allow you to click on the stars and select one rating. The Min/Max values should be shown and <input type="button" value="Submit"/> should be disabled if these requirements are not met.	✓
Questions Text should allow you to enter a textual value before submitting.	✓
Question Range should present a slider which allows you to choose a range of values. The min/max values should be shown on the slider.	✓

Continues on the next page...

Detail of test	Pass
The tabs Single Question, Question Multiple, Question Stars, Question Text, Question Range all should show one demonstration of the specific question type.	✓
The poll simple example should show an example poll.	✓
Caesar Example should have Videogular Cuepoints enabled so you will be able to see where the questions will be shown.	✓
Caesar Example's second question at the end of the example will have a poll to see the results of all those who have answered the question.	✓

The pass marks show each test was tested by the developers on this project and was successful, and also accepted by the client after review of the deliverable report.

9.4 Example Analytics Server

The analytics back-end server receives the [ReST](#) calls from the [Videogular](#) Analytics plugin and is then able to process them. For this example implementation it was chosen not to store the data but forward it to the front-end. This will demonstrate that the Videogular Analytics plugin is able to make [ReST](#) calls. Storage could be used to collate the information for later recall.

9.4.1 Cross-Origin Request Testing

[Requirement N2](#) states that each plugin should operate in a standalone manner. It needed to be ensure that the example server and analytics plugin works correctly when performing cross origin requests. To permit this on the example server testing code has been included to permit all cross origin requests, see [Listing 9.1](#).

```

1 app.all('*', function(req, res, next) {
2   res.header("Access-Control-Allow-Origin", "*");
3   res.header("Access-Control-Allow-Headers",
4     "Origin, X-Requested-With, Content-Type, Accept");
5   next();
6 });

```

Listing 9.1: Code showing appending Cross-Origin headers to all responses

To test that the analytics worked cross-domain the analytics example site and the Videogular Analytics plugin were set up on two different testing servers. This was able to correctly pass data between the two and therefore passed cross-origin testing.

9.4.2 Reloading Processed Data

The back-end acted as a testing platform for the front-end. A large set of data was recorded from the Videogular Analytics plugin and saved to allow for repeated testing of the same pieces of data. The back-end plugin has a tool to specify a data file that has already been processed to be loaded.

Any front-end client connecting to the back-end Videogular Analytics server will be sent all events in the loaded piece of data. This allows the front-end website to be quickly tested with a set of data that encompasses all features of the Videogular Analytics plugin. This data set allows comparison between a known valid set of data and data that Videogular Analytics creates allowing you to easily identify issues that occur.

This helps to faster identify cases where Videogular Analytics is not conforming to the [Application Programming Interface \(API\)](#) standard.

9.5 Example Analytics Front-End

The example analytics front-end loads up events from the Videogular Analytics plugin. These events are obtained by a websocket to example analytics server. Screenshots of the user interface produced can be found in [section A.2](#).

In the current model the example analytics server acts as an intermediary between this and the Videogular Analytics plugin that can serve additional events as described in [subsection 9.4.2](#).

This front-end tests the Videogular Analytics plugin to ensure that the data received is not missing or corrupting events. This is done by loading the data into different formats and processing it. The data processing assumes a number of standard data elements expected from the Videogular Analytics based on the [API \(Table 8.1\)](#).

9.5.1 Videogular Heatmaps

One of the analytics used in the example site is the Videogular Heatmap. When running an instance of Videogular Questions Example this can be used to dynamically update the heat map in the example analytics page (see [Figure 9.1](#)).

This is used to test that the Videogular Analytics plugin is able to properly export the timing data. The processing on the client side matches up start and end marks. These marks shown in [Figure 9.2](#) are then further processed to produce the data that the analytics can use.

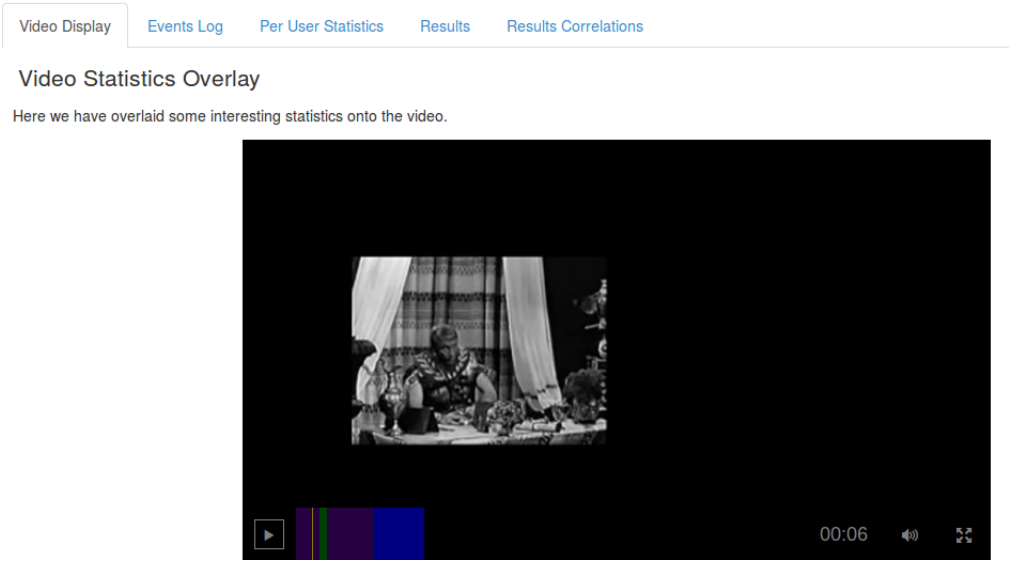


Figure 9.1: The heat map has dynamically updated

If these are not able to be processed then this indicates an issue with the events being sent from the [Videogular](#) Analytics plugin. If they are able to be sent but the heat map does not display then there is an issue with the heat map.

Running this against the predefined data is used as a regression test for [Videogular](#) Heatmap to ensure it is working correctly.

By running an instance of the Videogular Questions Example site and voting with the analytics end-to-end testing is performed as this data is passed through a number of the plugins to reach the Analytics front-end to be loaded by the [Videogular](#) Heatmap plugin.

Number of times each section was viewed

This table divides the video into sections, and shows the number of times that the user has viewed each one.

Start time	End time	Frequency
0.00 seconds	18.19 seconds	1
18.19 seconds	18.19 seconds	2
18.19 seconds	18.24 seconds	3
18.24 seconds	2.94 seconds	6
2.94 seconds	50.03 seconds	0

Figure 9.2: Analytics events after being processed into viewer time periods

The added feature of displaying the frequencies was also tested. These displayed well and matched the frequencies represented by the colours used. A screen reader could still access the data as it was declared in hidden text.

9.5.2 Videogular Analytics Integration with Videogular Questions

To test integration with [Videogular Questions](#) a set of dynamic graphs were included that update as events about question results are sent. [Figure 9.3](#) shows an example of the results of questions that have been sent from the plugin.

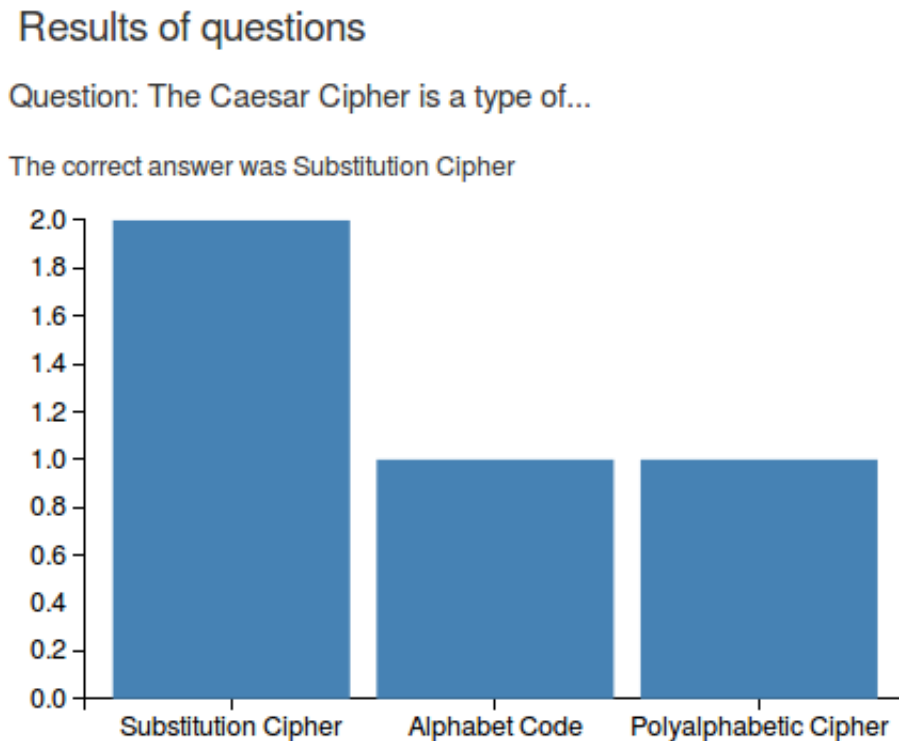


Figure 9.3: Graph showing results of questions sent from Videogular Analytics

If the question data is not being received this means that the [Videogular Analytics](#) plugin has not been able to properly access the [Videogular Questions](#) events. This test was used as a regression test to ensure that the Videogular Analytics plugin is correctly accessing all published analytics events and forwarding them to the [ReST](#) server.

9.5.3 Accessibility

The example analytics site generally conformed to the [Web Content Accessibility Guidelines \(WCAG\)](#) 2.0 standards (see [Table M.2](#)). The non-conformances found were a lack of alternatives for the video and graphs.

The video issues were the same as for the Videogular Questions Example site (see [subsection 9.3.4](#)). However the cuepoints were not used. The heat maps had a textual option that could be displayed which gave additional methods for conveying the information.

Using graphs has created many accessibility issues for visually impaired or cognitively impaired users as the data is not described in text. This would be an important consideration for any developer using the plugins in their own site. As a proof-of-concept the example site purely shows that graphs could be used as a method of automatically producing a visual representation of the data.

9.5.4 Deliverable Sign-Off tests

The tests in table [Table 9.3](#) were presented to the client as the deliverable items for the example analytics tool.

These tests were ran on the analytics tool example using preloaded data so that there was had a repeatable set of data that was always valid.

Table 9.3: vgAnalytics Deliverable sign-off tests

Detail of test	Pass
All events are listed at the top of the events log page, including the type and details of the events	✓
The sections that the users have watched are correctly calculated and shown at the bottom	✓
Watched video segments are calculated correctly and show how many times all users have viewed a section.	✓
The results page shows the user responses.	✓
The results page also shows correct answers and whether users rewatched sections.	✓
The “% watched by correct answers” and “Time watched by correct answers” graphs show two ways you could represent the data in a different way.	✗

Here the final requirement was queried by the client. The example graphs shown did not have an vertical axis. This was fixed by adding an axis after the client meeting however during the meeting this did not affect sign-off since it was an example of possible graphs that could be created.

9.6 Authoring Tool

The authoring tool is primary deliverable as it is the only item that is designed as a tool and not as a proof-of-concept example site deliverable. Therefore the primary types of testing are deliverable sign-off tests and accessibility testing.

9.6.1 Accessibility

The authoring tool was found to be generally accessible conforming to most of the [WCAG 2.0](#) standards (see [Table M.3](#)). The areas that were lacking were providing textual alternatives and error checking.

There were no alternatives provided for the video. However as this would be used with Synote transcripts could be used from there. In future, support for extra accessibility features such as transcripts, captions or signed versions could be included in the authoring tool.

In creating the questions there is no help provided for error checking. This is because it was decided not to narrow the options available to the users. In the occurrences where errors are identified these are generally highlighted by surrounding the component with a red border. More information on how the system would be used would be required in order to create helpful error messages.

Two keyboard accessibility issues were found, both of which were due to browser bugs. The first was that Google Chrome and Chromium did not allow multiple items to be selected in a `<select multiple>` element using only the keyboard. During the course of the project, a fix was made to Google Chrome and Chromium which allowed multiple *adjacent* items to be selected with the keyboard [6], which is an improvement but still not a complete fix.

The second keyboard accessibility issue was that collapsible section headings caused “focus loops” in Mozilla Firefox, where pressing the Tab key from the last button on the heading caused the focus to move back to the heading itself, preventing the user from focussing any content after the heading. This was reported on the Mozilla bug tracker [8] (see [appendix N](#)), where another user pointed out that the HTML causing the bug was invalid. The HTML was rewritten to be valid, which fixed the issue.

Another issue found was in the use of the `<select>` element as the colour of highlighted options is not editable in the [Cascading Style Sheets \(CSS\)](#). To solve this issue the elements would all need rewriting to take a completely different structure to allow the [CSS](#) to modify the colours used.

Accessibility in the authoring tool was good but there are improvements that could be made in future including adding support for accessibility features such as subtitles, providing error correction support and rewriting the `<select>` elements to allow editing of colours.

9.6.2 Deliverable Sign-Off Tests

The tests in [Table 9.4](#) were presented to the client as the deliverable items for the authoring tool.

Table 9.4: Authoring tool Deliverable sign-off tests

Detail of test	Pass
Pressing <code>Export</code> will provide you with a Definition File which can be used.	✓
Pressing <code>Preview</code> will reset the video back to the start, load the questions into the preview, and begin playing	✓
All 5 question types are implemented and, as suggested, the Single choice question (with radio buttons) is merged with the Multiple choice question.	✓
A question set is able to be added at a specific time.	✓
Once a single question has been added the <code>Preview</code> button is used to ensure this is shown properly	✓
Each question type is tested along with common options and displays as expected.	✓
Skipping is tested for one question and works as expected.	✓
When checked, the record button sends the poll data to the poll server.	✓
A full walkthrough is attempted, all question types are able to be added and previewing, and exporting works	✓

The pass marks show each test was tested by the team and was successful, and also accepted by the client after review of the deliverable report.

9.7 Summary

Working in an agile manner meant that each week work was presented to the client. This meant that the primary types of testing are ensuring functionality works through integration tests and the final deliverable reports. The example websites have been

created to demonstrate that the plugin works and to allow testing of them in the environment they are planned to be used in.

Some repositories have been tested using a unit testing strategy([subsection 9.2.1](#)) which is focussed on ensuring that regression tests are ran and detection of errors before committing code. For these repositories the master branch should always build and function correctly. This has allowed meetings with the client to always demonstrate working projects while at any stage of development.

The analytics front-end server is an example of integration testing performed. Here [Videogular](#), Videogular Analytics, and Videogular Heatmap is used to demonstrate a number of statistics about how a user is using the interface. There are tests for each of these modules but identified a number of issues that occurred after linking these plugins together. Videogular Questions, Videogular Analytics, and results server have been tested in this manner using the Videogular Examples website.

In server applications that require a fast response load testing has been performed to ensure that performance degradation would not affect a users experience. This has been shown ([Table 9.1](#)) to cause any user to have high latency and suggestions have been made to improve it if needed.

These tests have demonstrated to the client that this has been thoroughly tested and they have signed off the deliverable report ([appendix O](#)) presented to them. The documentation provided also includes recommended practices for continued work to satisfy [Requirement N10](#).

Future work

This framework is just the beginning of the work that could be undertaken in this area, there are a number of interesting improvements that can be made to enhance and extend the project. This chapter details a number of these potential improvements.

10.1 Introduction

During the project a number of areas were found that could have been expanded. Some of these have been looked into but not implemented due to time constraints and others have been suggested as points of future work by the client. A number of the important items of future work have been detailed below.

10.2 Integration with Synote

The most important part of the project is ensuring that the work done is able to be easily integrated into Synote.

Therefore during the process there was communication with the current developer of Synote to ensure that the work done will be able to be used.

One key requirement was that the all work done should be able to standalone and not require any additional Synote framework but that Synote should be able to communicate with these pieces of work easily (see [Requirement N2](#)).

To ensure this all, all services use an open standard to communicate using either [Representational State Transfer \(ReST\)](#) calls ([Requirement N11](#)), with associated documentation of each call and how it can be used, or protocols such as Web sockets. These services can be run as small Web servers which will allow Synote to communicate with them easily. In addition, since these are standalone, the best language for the associated service could be chosen. This has decreased the complexity which should reduce the time needed to learn how the software works in order to continue from this project.

The core of the project uses [AngularJS](#) and the [Videogular](#) player as this was recommended as the latest version of Synote will be written in Angular. All of the plugins were written for the [Videogular](#) player and therefore will be able to be easily integrated into the primary Synote codebase. The plugins are the only part that will be integrated into Synote directly and therefore were best suited to be written in Angular. The other services will communicate with Synote and therefore did not need to be written in Angular.

10.3 Additional Video Players

Since the plugins are built upon the [Videogular](#) system there was no concern with how the video is played. The base [Videogular](#) system plays the common video formats.

To further expand the system the [Videogular](#) YouTube Plugin¹ could be fixed. This would allow users to specify a YouTube video to use in the authoring tool and removes the problems with getting all formats of video for all browsers.

This was not worked on due to the time requirement and was decided out of scope by the client. However some work was done looking into the changes required and it appeared to be possible to fix for the latest version.

10.4 Video Encoding

To display video using the default [Videogular](#) plugin the video needs to be in a number of formats to ensure all browsers can play the video. Therefore if the user wishes to use their own video they will need to manually convert the video to all required formats before it will be usable on all operating systems/browsers.

The process of converting the video to each of the required formats is non-trivial. In addition, finding out that video formats are required requires knowledge of what to look for. This has a high barrier to entry and therefore could prevent possible users being able to use the tools.

There are a number of open source video conversion tools that could be used. An idea for an improvement would be to modify the authoring tool to allow a user to upload a video. Once this has been uploaded it will be automatically stored and converted to the correct formats. The server would then store these new formats and either set up the quiz with the video or give the user the newly converted video URLs for inclusion to their quiz.

This would be helpful to reduce the barrier to entry ([Requirement N8](#)) as the user would not need to convert the video to the correct format which can be highly technical.

10.5 Mobile Interfaces

Creating interfaces that work well with mobile interfaces is a difficult task and therefore minimal time was spent to ensure that all basic examples work on most mobile devices.

Further work would be concerned with getting the plugin to work on all devices as iPhones have shown to have significant issues with playing video and overlaying content (see [section 5.3](#)).

Tablets have been shown to work fine with the plugins as they generally have near desktop size resolutions and full capabilities to play video without forcing the user to

¹<https://github.com/NamPNQ/bower-videogular-youtube>

maximise the screen. Therefore work in the area would be focussed on display this information on a mobile phone.

There may be other improvement that can be made for small screen applications such as making buttons much larger (easier to press) and hiding the video while the popup is shown (so the phone does not need to render the video and popup in front of part of it).

10.6 Second Screen

A previous project worked on the possibility of creating a second screening application that worked with Synote. This allowed two views of Synote to be viewed and display different data on a second screen. The second screen could often be a mobile device.

If further time were available extension of the plugins developed to have a "second screen" option for users could be investigated. This would allow users to turn their mobile device into a second screen for the question popups to allow them to answer the questions on their device.

This would also allow a user to take advantage of additional lecture features such as subtitling. Another project is currently being run to get in-lecture subtitles and this could be a feature added as another plugin.

10.7 Angular 2

At the 2014 ng-europe conference during a talk about the future development of Angular details about the 2.0 release were given. It is set for an early 2016 release, with Angular 1.x receiving bug fixes for another 2 years.

Currently it seems that Angular 2.0 is a complete rewrite of the Angular ecosystem and is only related to Angular 1.x by name. Upon its release applications wishing to stay upto date will need to be rewritten from scratch. As this project is currently built around Videogular, it would first need to be updated and rewritten. The framework's main client, Synote, is currently in development and as such is using Angular 1.x.

As such there is potential future work for investigating Angular 2.0 upon its final release. At that time much more information regarding potential upgrade paths would exist.

10.8 Accessibility

With the focus on accessibility there were several improvements that could be made to the plugins to create a much more accessible system for users.

10.8.1 Cuepoints and Heatmap

Currently the only way of getting information from the [Videogular](#) Cuepoints is through the use of colour. [Videogular](#) Heatmaps also has the option to include the frequency as text. Although the colours are customisable these plugins are not very accessible. An improvement that could be made in future is to have information available when the marked section is in focus (on hover and when it is playing) that gives the details used by the plugin. For example an area of a heat map could display the number of times the section has been viewed, and the start and end times of the section.

10.9 Authoring Tool

The authoring tool accessibility could be improved by adding support for accessibility features such as captions and transcripts. This would aid the improvement of the accessibility of the content when using it in the questions overlay.

The elements used to create dropdowns and multiple select boxes could also be rewritten to allow the editing of colours in [Cascading Style Sheets \(CSS\)](#). This would increase the ease of modification to fit a specific users requirements or styling towards a website theme.

10.10 Use of Graphs

In the Example Analytics site graphs have been used as a representation but they have no textual alternative. In future, any graphs used should be made accessible to screen readers and users with cognitive impairments.

Since the analytics website is an example making this entirely accessible was not a high priority the deliverables that the client explicitly requested were prioritised rather than this example website.

10.11 Summary

The primary focus of this project towards future work has always been easing the integration into Synote. The requirements to ease integration ([Requirement N2](#)) were followed to aid this.

As the project was based on the Videogular player as a plugin, fixing the YouTube plugin should be a high priority for future work. This will lower the barrier to entry ([Requirement N9](#)) as users will not need to upload video and encode it correctly for different browsers. This was decided out of scope by the client. Another way of doing this would be automatically re-encoding the video on upload, which is also a possible improvement.

Some research into mobile interfaces was done and found a number of problems with browser and differing phone behaviour and sizes. This is a very hard problem and therefore to focus on the original problem this was not looked at.

Previous work with second screening could be reviewed and re-implemented in this project but this would primarily rely on having the tools working on mobile devices (which are generally used for the second screening application).

Angular 2 does not look like it will present a problem as Angular 1 is likely to be support for a reasonable period of time. Therefore the support of the application will not become problematic as deprecation of Angular 1 will not be soon. More information will be available when Angular 2 is released in the future.

Accessibility is a general topic that can be improved in all areas of the applications but much of these problems fall down to making the videos more accessible by adding captions and reducing the use of colours as the sole means of conveying information. Keyboard accessibility ([Requirement N6](#)) has been focussed on in terms of accessibility as this was one of the original requirements.

Conclusions

The conclusions outlines the project and the degree of accomplishment focussing on client satisfaction as the primary measure of success.

“It was a ‘Jump to Conclusions’ mat. You see, it would be this mat that you would put on the floor; and would have different conclusions written on it that you could jump to!”
- Richard Riehle as Tom Smykowski in Office Space (1999)

11.1 Project Management

Throughout the project, work was completed at a steady rate ([Figure 4.2](#)) and work periods were distributed evenly over a 40 hour working week with some additional work at weekends and later at night ([Figure 4.1](#)). The Gantt charts produced ([appendix F](#)) were used to track progress during meetings to ensure that client's priority deliverables had enough time.

The key to this project's success was the use of the agile software development process. It allowed maximum reactivity and flexibility to ensure client satisfaction. Regular meetings occurred between the developers and the client. These were to keep the client apprised of the progress, and obtain the client's feedback on the work done ([appendix E](#)). The development team also met regularly with additional meetings when necessary. During the start of the project, and some integration phases, it was beneficial to meet to work collaboratively. However, work was also done individually.

11.2 System Overview

During the initial stages of the project, considerable effort was spent on fully comprehending the problem and investigating potential technical solutions. This resulted in an approach that, while being quite complicated, drew on a large number of tools and technologies. This provided the appropriate flexibility and functionality.

The plugin ([Videogular Questions](#)) central to this project involved overlaying quizzes and polls onto videos. From the review of previous work it was found that instantaneous feedback is key to learning, therefore this was an important feature of the [Videogular Questions](#) plugin. Another important client requirement was that all plugins should be extensible. Using [AngularJS](#) for the user interface of the [Videogular Questions](#) plugin, gives the required extensibility with the additional benefits of efficient reuse of code and high levels of customisability.

In order for a dynamic set of questions to be displayed to the user, [WebWorkers](#) (JavaScript sandboxed threads) have been utilised. This allowed for the use of a programmatic [Definition File](#), rather than using an exhaustive data format. This was a very different approach to using the [Question and Test Interoperability \(QTI\)](#) definition standard; making accessibility considerations easier to implement without restricting the options available to authors.

The creation of the [Definition File](#) is a non-trivial task. Thus, an authoring tool was produced to reduce the barrier to entry, enabling non-technical users to use the system. Accessibility considerations were a key factor in the development of the authoring tool to satisfy the client's requirements ([Requirement N6](#) and [Requirement N7](#)).

In user generated sites that use the [Videogular Questions](#) plugin, there is the option to include analytics functionality provided by the [Videogular](#) Analytics plugin. This enables future work researching the different ways users interact with eAssessment and learning systems to develop the next generation of interactive learning videos.

There have been a number of areas that are promising for further development in this system. The client is interested in additional plugins to handle additional video formats. The [Videogular Questions](#) abstraction supports different video formats without the need to modify any code in the plugin.

Further work towards accessibility (as outlined in [section 10.8](#)) will increase the range of users able to use the system. A fully accessible system would allow new research that could improve eAssessment for people with disabilities.

11.3 Summary

During the project process, a number of important papers and research books have been reviewed to provide the necessary background in eAssessment ([chapter 2](#)). Using the information gathered from both these academic and industrial works the team has produced an innovative product that has fully satisfied the client.

Creating [Videogular Questions](#) and the associated tools in the framework has been a challenging project. It now allows users to go from the creation stage, through the publishing stage, and on to the analytics stage. Having met all goals the team looks forward to the client using the project within Synote, and hopefully seeing it used elsewhere online.

Bibliography

- [1] Charles Arthur. Adobe kills mobile Flash, giving Steve Jobs the last laugh. *The Guardian*, 9 November 2011. Available: <http://www.theguardian.com/technology/2011/nov/09/adobe-flash-mobile-dead> (Accessed: 26 Nov 14), 2011.
- [2] David Bacigalupo, Bill Warburton, E.A. Draffan, Pei Zhang, Lester Gilbert, and Gary Wills. A Formative eAssessment Co-Design Case Study. In *The 10th IEEE International Conference on Advanced Learning Technologies*, [online], July 2010. <http://eprints.soton.ac.uk/271236/> (Accessed: 01 Dec 14).
- [3] Susan M. Brookhart. Successful Students’; Formative and Summative Uses of Assessment Information. *Assessment in Education: Principles, Policy & Practice*, 8(2):153–169, [online], 2001. <http://www.tandfonline.com/doi/pdf/10.1080/09695940123775> (Accessed: 01 Dec 14).
- [4] Tom Cherrett, Gary B. Wills, Joseph Price, Sarah Maynard, and Itiel E. Dror. Making training more cognitively effective: making videos interactive. *British Journal of Educational Technology*, 40(6):1124–1134, [online], November 2009. <http://eprints.soton.ac.uk/267281/> (Accessed: 01 Dec 14).
- [5] K. Chorianopoulos and M.N. Giannakos. Merging learner performance with browsing behavior in video lectures. In *Proceedings of the Workshop on Analytics on Video-based Learning (WAVE 2013)*, volume 983, pages 37–41. CEUR-WS, [online], 2013. <http://ceur-ws.org/Vol-983/paper9.pdf> (Accessed: 01 Dec 14).
- [6] Chromium contributors. Using the keyboard to select multiple items in a select html tag, April 2012. Google Code [Online]. Available: <https://code.google.com/p/chromium/issues/detail?id=125585> (Accessed: 23 Jan 2015).
- [7] Ionut Colceriu. A more accessible HTML5 <video> player, December 2010. Dev.Opera [Online]. Available: <https://dev.opera.com/articles/more-accessible-html5-video-player/> (Accessed: 22 Jan 2015).
- [8] Harry Cutts and Firefox contributors. <input>s within <div>s within <a>s create focus cycles, November 2014. Bugzilla@Mozilla [Online]. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=1106045 (Accessed: 23 Jan 2015).

- [9] Jelle De Boer, Piet A. M. Kommers, and Bert De Brock. Using Learning Styles and Viewing Styles in Streaming Video. *Comput. Educ.*, 56(3):727–735, April 2011.
- [10] Francesco Epifania and Politecnico di Milano. Design and development of multimedia interactive systems for digital learning. In *DMS*, pages 238–241, [online], 2011. http://www.ksi.edu/seke/Proceedings/dms11/DET/19_Epifania_Francesco.pdf (Accessed: 01 Dec 14).
- [11] Philip J. Guo, Juho Kim, and Rob Rubin. How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 41–50, New York, NY, USA, [online], 2014. ACM. ISBN 978-1-4503-2669-8. <http://doi.acm.org/10.1145/2556325.2566239> (Accessed: 01 Dec 14).
- [12] Wynne Harlen and Mary James. Assessment and Learning: differences and relationships between formative and summative assessment. *Assessment in Education: Principles, Policy & Practice*, 4(3):365–379, [online], 1997. <http://www.tandfonline.com/doi/pdf/10.1080/0969594970040304> (Accessed: 01 Dec 14).
- [13] Ilire Hasani-Mavriqi, Christoph Portschi, Christoph Trattner, Brigita Kacjan, Denis Helic, and Hermann Maurer. Using Wiki for Rapid Authoring of Huge Numbers of E-Assessments. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2010*, pages 3046–3053, Toronto, Canada, June 2010. AACE.
- [14] IMS Global. *IMS Question & Test Interoperability Implementation Guide Version 2.1 Final*, 2012.
- [15] IMS Global. *IMS Question & Test Interoperability Overview Version 2.1 Final*, 2012.
- [16] Peter Knight. *A briefing on key concepts: Formative and summative, criterion and norm-referenced assessment*. Learning and Teaching Support Network, 2001.
- [17] Nadezhda Kolodyazhnaya. Accessible interactive video quizzes for e-learning systems. Master’s thesis, Electronics and Computer Science, University of Southampton, 2014.
- [18] Dennis Lembree. Introducing an accessible HTML5 video player, September 2014. PayPal Engineering [Online]. Available: <https://www.paypal-engineering.com/2014/09/05/introducing-an-accessible-html5-video-player/> (Accessed: 22 Jan 2015).
- [19] Yunjia Li, Mike Wald, and Gary Wills. Applying linked data in multimedia annotations. *International Journal of Semantic Computing*, 6(3):289–313, [online], September 2012. <http://eprints.soton.ac.uk/273063/> (Accessed: 01 Dec 14).

- [20] M. Magdin, M. Capay, and M. Mesarosova. Usage of interactive video in educational process to determine mental level and literacy of a learner. In *Interactive Collaborative Learning (ICL), 2011 14th International Conference on*, pages 510–513, [online], September 2011. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6059637> (Accessed: 01 Dec 14).
- [21] Michael Piotrowski. QTI: A failed e-learning standard? In Fotis Lazarinis, Steve Green, and Elaine Pearson, editors, *Handbook of Research on E-Learning Standards and Interoperability: Frameworks and Issues*, pages 59–82. IGI Global, Hershey, PA, USA, 2011.
- [22] H. Prima Dewi Purnamasari and N. Syifana. Clickable and interactive video system using html5. In *Information Networking (ICOIN), 2014 International Conference on*, pages 232–237, [online], February 2014. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6799697> (Accessed: 01 Dec 14).
- [23] Marco Ronchetti. Videlectures ingredients that can make analytics effective. In *Proceedings of the Workshop on Analytics on Video-based Learning (WAVE 2013)*, volume 983, pages 15–20. CEUR-WS, [online], 2013. <http://ceur-ws.org/Vol-983/paper4.pdf> (Accessed: 01 Dec 14).
- [24] Niall Sclater. The Demise of eAssessment Interoperability? In Hugh C. Davis, Erik Duval, Brandon Muramatsu, Su White, and Frans Van Assche, editors, *WWWrong*, volume 317 of *CEUR Workshop Proceedings*. CEUR-WS.org, [online], 2007. <http://ceur-ws.org/Vol-317/paper07.pdf> (Accessed: 01 Dec 14).
- [25] The Joint Information Systems Committee. MLEs and VLEs Explained, [online], 2001. http://www.jisc.ac.uk/uploaded_documents/bp1.pdf (Accessed: 01 Dec 14).
- [26] The Joint Information Systems Committee. Effective Assessment in a Digital Age, [online], 2010. http://www.jisc.ac.uk/media/documents/programmes/elearning/digiassass_eada.pdf (Accessed: 01 Dec 14).
- [27] W3C. User Agent Accessibility Guidelines (UAAG) 2.0, [online], 2014. <http://www.w3.org/TR/UAAG20/> (Accessed: 01 Dec 14).
- [28] Gary Wills, Lester Gilbert, Jonathan Hare, Jiri Kajaba, David Argles, and David Millard. Assessment Delivery Engine for QTIv2 Tests. In *4th Ten Competence Open Workshop*, [online], April 2008. <http://eprints.soton.ac.uk/265979/> (Accessed: 01 Dec 14).

Glossary

AngularJS A client side Model-View-Controller architectural pattern.

Annotation In the Videogular Questions plugin, a set of questions that appear at a particular time.

API A defined interface that allows the access of features of a program or service.

ARIA Accessible Rich Internet Applications. A Web Accessibility Initiative technical specification from the World Wide Web Consortium (W3C) that specifies how to make web pages more accessible.

Assessment Delivery System Tool responsible for displaying and scoring the assessments.

Authoring Tool Tool responsible for creating the questions and assembling the test.

Blob A blob represents some immutable raw data. From this you can create a URL..

CAA Computer-Aided Assessment. *see [eAssessment](#).*

CAA Computer Assisted Assessment. *see [eAssessment](#).*

CBA Computer Based Assessment. *see [eAssessment](#).*

CSS Cascading Style Sheets. Language used for specifying the styling of a document written in a markup language.

Definition File The definition file is used by Videogular Questions plugin to create the overlaid quiz/poll.

DOM Document Object Model. A convention for accessing and updating objects in HTML and XML documents. It is cross-platform and language-independent.

eAssessment The process of making, viewing and scoring assessments using a computer.

HTML5 HyperText Markup Language version 5. A language used to describe Web pages, which has come to refer to the combination of HTML5, [Cascading Style Sheets \(CSS\)](#) and JavaScript.

MLE Managed Learning Environment. The overall system used by an educational establishment to facilitate and manage learning.

MOOC Massive Open Online Course. A free study course available over the internet.

MVC Model-View-Controller. An architectural pattern used in creating user interfaces.

QTI Question and Test Interoperability. Quiz interoperability standard by IMS Global.
<http://www.imsglobal.org/question/#version2.1>.

ReST Representational State Transfer. An architectural pattern focusing on simple interfaces, scalability, portability, reliability and modifiability of components.

scrub bar A bar commonly found on video player user interfaces which shows the current position in the video, and often how much of the video has been downloaded so far.

UAAG User Agent Accessibility Guidelines.

Videogular HTML5 video player for [AngularJS](#).

Videogular Questions A plugin to overlay quizzes and polls on videos.

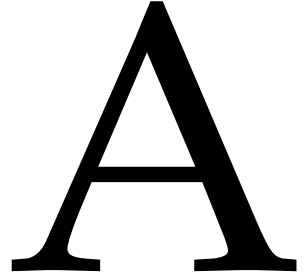
VLE Virtual Learning Environment. Component of a Managed Learning Environment responsible for the online interactions of students and tutors.

WAI Web Accessibility Initiative.

WCAG Web Content Accessibility Guidelines: <http://www.w3.org/TR/WCAG20/>
(Accessed: 15 Jan 15).

WebWorker A sandboxed separate thread that runs in the background of the webpage.

Appendix



Screenshots

An appendix containing screenshots of the software produced.

A.1 Videogular Questions Example

Amazing Video Quizzes and Poll A Videogular-Questions example - [github](#)

[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Simple Poll Example Video

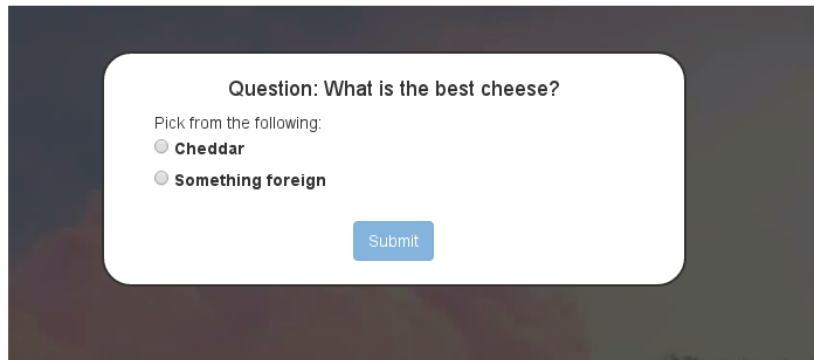


Figure A.1: [Videogular Questions](#) simple example poll

Amazing Video Quizzes and Poll A Videogular-Questions example - [github](#)

[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Caesar Cipher Video



Figure A.2: [Videogular Questions](#) Caesar cipher example with related questions

Amazing Video Quizzes and Poll A Videogular-Questions example - [github](#)

[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Single Choice Question Example

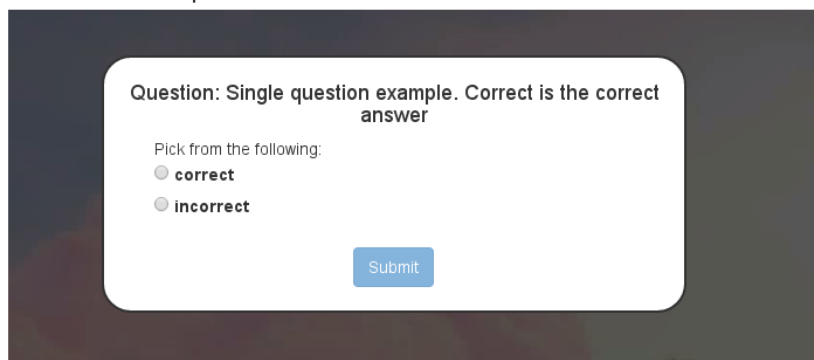
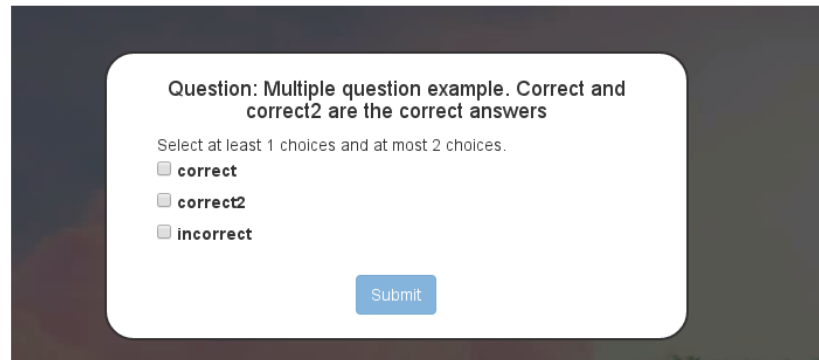


Figure A.3: [Videogular Questions](#) example showing a single selection question

Amazing Video Quizzes and Poll

A Videogular-Questions example - [github](#)[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Multiple Choice Question Example



Question: Multiple question example. Correct and correct2 are the correct answers

Select at least 1 choices and at most 2 choices.

☐ correct

☐ correct2

☐ incorrect

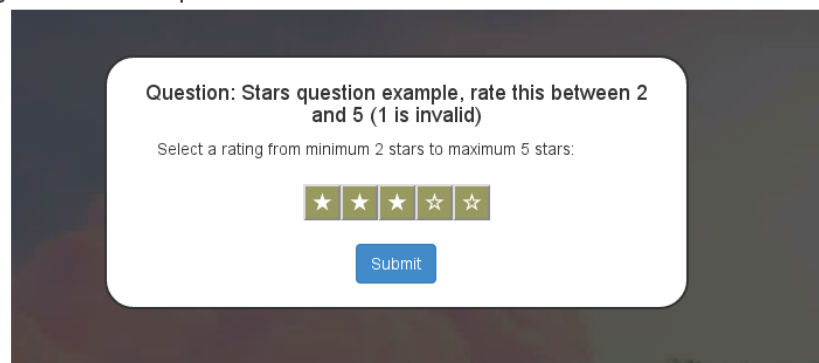
Submit

Figure A.4: [Videogular Questions](#) example showing a multiple selection question

Amazing Video Quizzes and Poll

A Videogular-Questions example - [github](#)[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Stars Rating Question Example



Question: Stars question example, rate this between 2 and 5 (1 is invalid)

Select a rating from minimum 2 stars to maximum 5 stars:

★ ★ ★ ★ ★

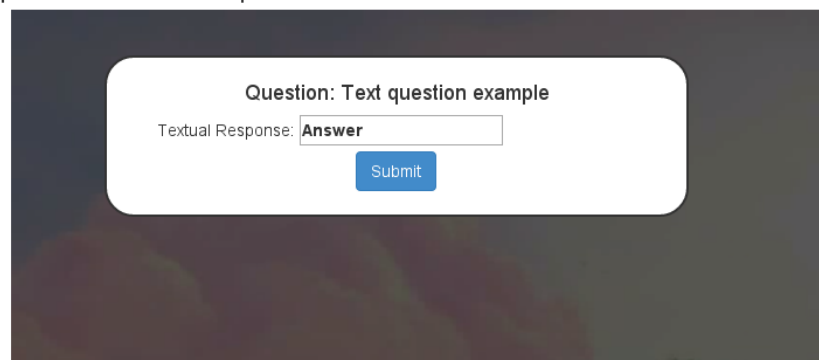
Submit

Figure A.5: [Videogular Questions](#) example showing a stars rating question

Amazing Video Quizzes and Poll

A Videogular-Questions example - [github](#)[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Textual Response Question Example



Question: Text question example

Textual Response:

Submit

Figure A.6: [Videogular Questions](#) example showing a text question

Amazing Video Quizzes and Poll

A Videogular-Questions example - [github](#)

[Simple Example](#) | [Caesar Example](#) | [Single Question](#) | [Question Multiple](#) | [Question Stars](#) | [Question Text](#) | [Question Range](#) | [Poll Simple](#)

Range Question Example

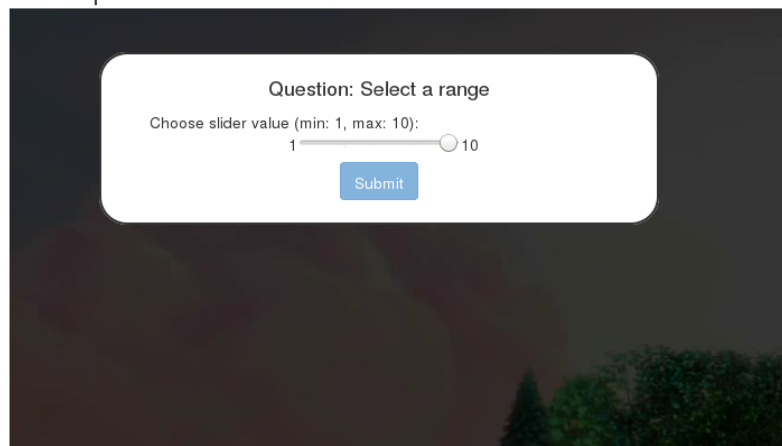


Figure A.7: [Videogular Questions](#) example showing a range question

A.2 Videogular Analytics

Videogular Analytics

Analytics on the use of the Videogular player created by GDP group 12 - [github](#)

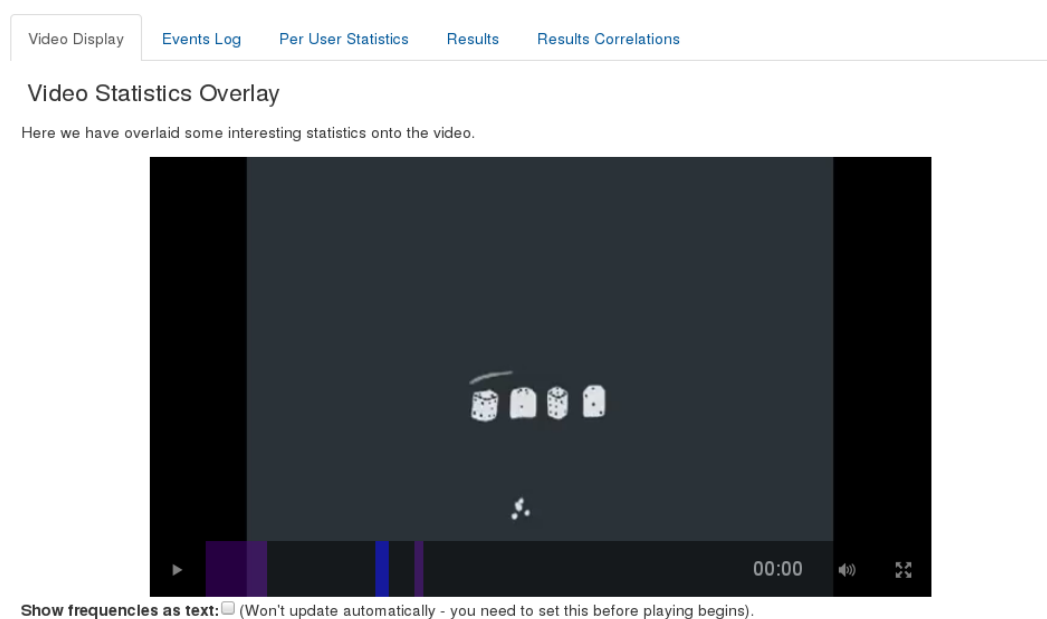


Figure A.8: [Videogular](#) Analytics example showing the video display

Videogular Analytics

Analytics on the use of the Videogular player created by GDP group 12 - [github](#)

Video Display Events Log Per User Statistics Results Results Correlations

Full Events Log

This tab lists all of the events as they are received from users using the site. Only events that are sent after this page has been loaded are currently recorded.

This table lists the event name, when it was received and the content of the event. This is a low level view of all the possible events received and demonstrates the range of data that is being collected. This data is used by a number of metrics demonstrated on this example webpage.

Time	UUID	Name	Content
2014-12-03T12:29:02.409Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	play	{ "time": 0 }
2014-12-03T12:29:13.414Z	36f66e09-8c36-4e3c-8078-b7cc5bdf1af3	play	{ "time": 0 }
2014-12-03T12:29:20.175Z	94a28b5d-3e30-4e87-9e2e-f951fcadb51	play	{ "time": 0 }
2014-12-03T12:29:20.732Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	show_question	{ "showQuestion": { "id": "name", "type": "single", "question": "The Caesar Cipher is a type of...", "options": [{ "name": "Substitution Cipher" }, { "name": "Alphabet Code" }, { "name": "Polyalphabetic Cipher" }], "correctAnswer": "Substitution Cipher" } }
2014-12-03T12:29:20.738Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	pause	{ "time": "1970-01-01T00:00:18.224Z" }
2014-12-03T12:29:24.641Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	submitted_question	{ "result": "Substitution Cipher" }
2014-12-03T12:29:24.650Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	end_question	{ "endAnnotation": "first-question" }
2014-12-03T12:29:24.664Z	3743a9a1-dcf0-4922-8b43-6f84d0ca1d32	play	{ "time": "1970-01-01T00:00:18.224Z" }
2014-12-03T12:29:31.734Z	36f66e09-8c36-4e3c-8078-b7cc5bdf1af3	show_question	{ "showQuestion": { "id": "name", "type": "single", "question": "The Caesar Cipher is a type of...", "options": [{ "name": "Substitution Cipher" }, { "name": "Alphabet Code" }, { "name": "Polyalphabetic Cipher" }], "correctAnswer": "Substitution Cipher" } }
2014-12-03T12:29:31.744Z	36f66e09-8c36-4e3c-8078-b7cc5bdf1af3	pause	{ "time": "1970-01-01T00:00:18.241Z" }

Figure A.9: Videogular Analytics example showing the events display

Videogular Analytics

Analytics on the use of the Videogular player created by GDP group 12 - [github](#)

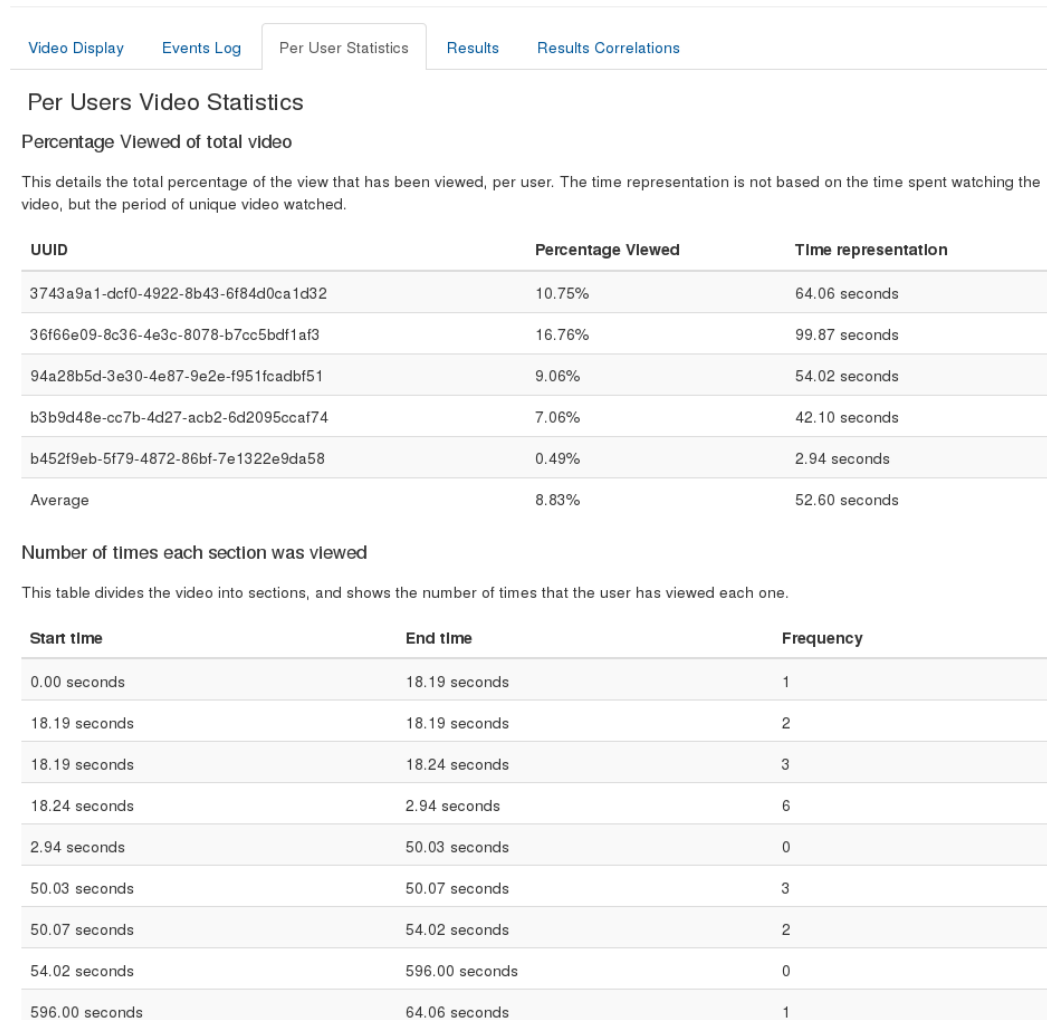


Figure A.10: [Videogular](#) Analytics example showing the statistics display

Videogular Analytics

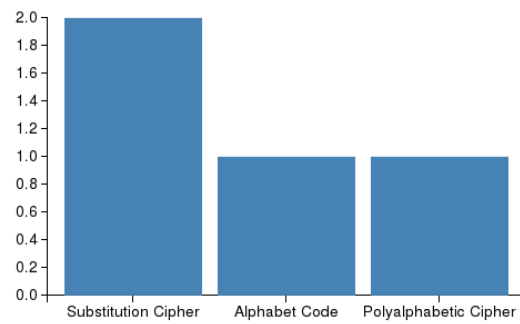
Analytics on the use of the Videogular player created by GDP group 12 - [github](#)

[Video Display](#) [Events Log](#) [Per User Statistics](#) [Results](#) [Results Correlations](#)

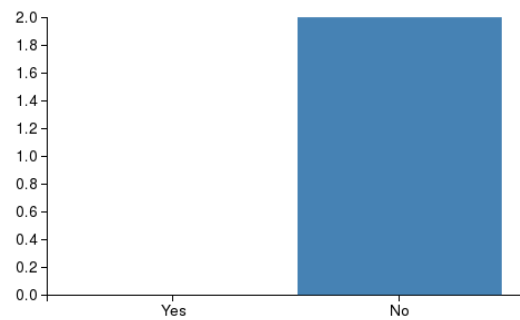
Results of questions

Question: The Caesar Cipher is a type of...

The correct answer was Substitution Cipher



Question: Answer incorrect, do you want to review the video



Question: How long did the Caesar Cipher remain unbroken...

The correct answer was 800 years

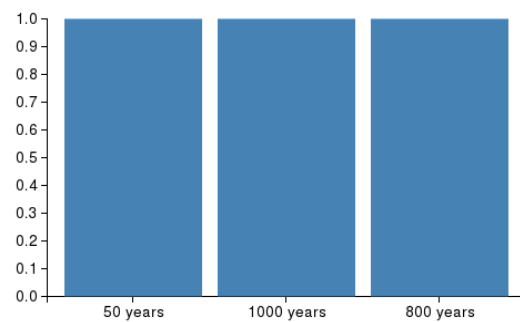


Figure A.11: [Videogular](#) Analytics example showing the results display

Videogular Analytics

Analytics on the use of the Videogular player created by GDP group 12 - [github](#)

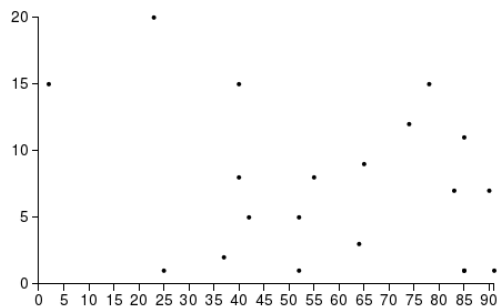
[Video Display](#) [Events Log](#) [Per User Statistics](#) [Results](#) [Results Correlations](#)

Results Correlation

This section shows a number of scatter plots that show the correlation between marks and the time spent on the video.

% watched by correct answers

This shows how much of the video watched in percent compared with the score the student achieved. This does not take into account sections of the video rewatched and therefore the highest value is 100%.



Time watched by correct answers

This shows how much of the video was watched in time compared with the score the student achieved. This takes into account if the video was rewatched and therefore a value higher than the video length is possible.

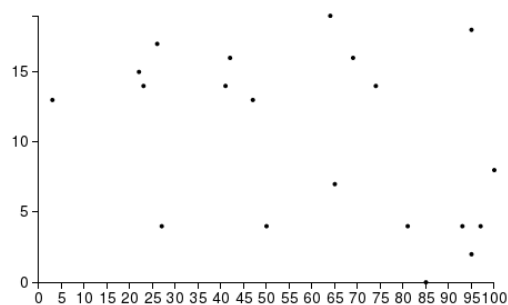


Figure A.12: Videogular Analytics example showing the results correlation display

A.3 Authoring Tool

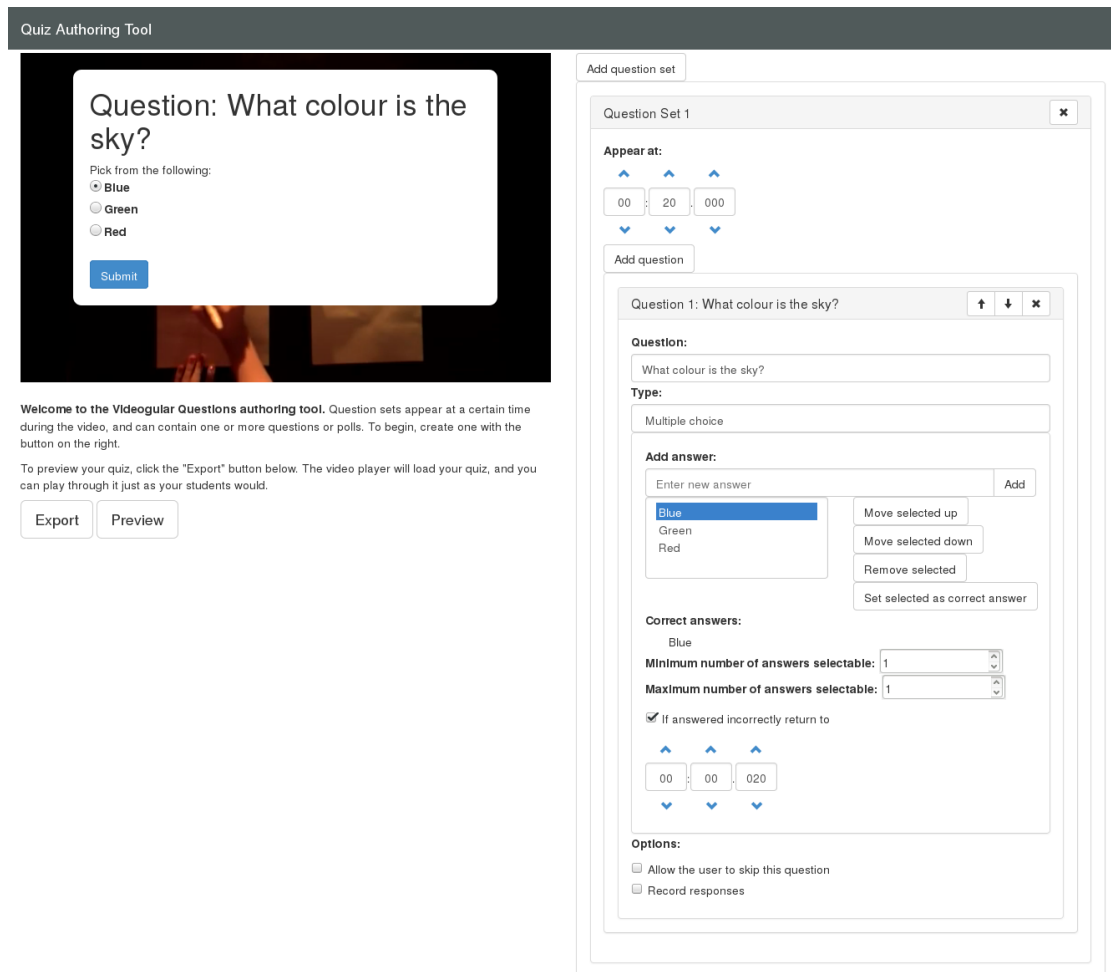


Figure A.13: Final authoring tool user interface

Skills Audit Results

Appendix

B

An appendix detailing the skills audits that were performed.

*“Debugging is twice as hard as writing the code in the first place.
Therefore, if you write the code as cleverly as possible, you are, by
definition, not smart enough to debug it.”*

- Brian W. Kernighan

B.1 Skills Audit Matrices

A skills audit of the group members was performed and skills audit matrices was produced for all tasks relevant to the project.

Below acronyms for each group member are used in the tables. Each acronym relates to a below member of the group:

- S - Samuel Bennett
- H - Harry Cutts
- CB - Christopher Baines
- CH - Christopher Hewett
- M - Maria Lynch

B.1.1 Project Management

Task	S	H	CB	CH	M	Average
Planning	5	3	5	4	5	4.4
Progress tracking	3	4	5	4	4	4.0
Time management	5	4	5	4	4	4.4
Contingency planning	4	4	5	5	5	4.6

B.1.2 Technical

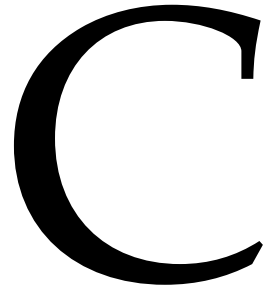
Task	S	H	CB	CH	M	Average
Technical Research	4	4	4	5	5	4.4
Analysis	5	4	4	5	5	4.6
Architecture Design	3	5	4	5	4	4.2
API Design	3	5	4	4	4	4.0
HCI / Interface Design	4	4	4	5	5	4.4
Implementation						
- Client Side	5	4	4	4	5	4.4
- Server Side	4	5	5	5	4	4.6
Evaluation	5	4	4	4	5	4.4

B.1.3 Communication

Task	S	H	CB	CH	M	Average
Technical Writing						
- Software Documentation	4	5	4	5	5	4.6
- Report	5	4	5	4	5	4.6
Academic Research	5	4	4	5	5	4.6
Presentation	5	5	4	4	4	4.4
Critical and comparative evaluation	4	5	3	4	5	4.2
Reflection	4	4	3	5	4	4.0

Coding Conventions

Appendix



An appendix detailing the coding conventions and standards for the project.

“How standards proliferate:

Situation: There are 14 competing standards

*Researcher: 14?! Ridiculous! We need to develop one universal
standard that covers everyone’s use case!*

Soon:

Situation: There are now 15 competing standards”

- XKCD - web comic¹

¹XKCD Standards - <http://xkcd.com/927/>

C.1 Introduction

This appendix contains the coding conventions, from the GitHub repository².

C.2 General

File names should be in Unix form (lower-case, words separated by hyphens, e.g. `a-directory/my-file.type`).

C.3 CSS Coding Conventions

Tabs should be used for indentation.

C.3.1 Property Ordering

In general, properties further towards the top should affect layout, while properties towards the bottom should affect appearance.

Specifically, the most common properties should be in this order:

```
1  .class {  
2      position  
3      display  
4  
5      flex /* etc. */  
6  
7      top  
8      bottom  
9      left  
10     right  
11     z-index  
12  
13     width /* also min-, max- */  
14     height /* also min-, max- */  
15  
16     padding  
17     border  
18     margin  
19  
20     font /* etc. */  
21     text-align  
22     background /* etc. */
```

²<https://github.com/soton-ecs-2014-gdp-12/conventions>

```
23     color
24 }
```

C.4 HTML coding conventions

All full HTML pages should specify `<!DOCTYPE html>`.

C.4.1 Indentation

Tabs should be used up to the indent level, with spaces for lining up tags which break over multiple lines.

C.4.2 Attributes

Attribute values should be quoted.

The `id` attribute should always be first after the tag name, followed by the `class` attribute. For meta tags, the `name` should be specified first.

C.4.3 Scripts

Where possible, scripts should be imported at the end of the body. For JavaScript files, the optional `type` attribute should be omitted.

C.5 Issue Tracker Usage Guidelines

If an issue doesn't seem to fit with any particular repository, file it against the [Videogular Questions](#) repository.

The issue type should be described with a label (one of bug, enhancement, or investigate).

C.5.1 waffle.io and Sprint Management

There is a waffle.io board³ for tracking the Sprints.

Each Sprint will have a milestone on each repository, named "Sprint ", with the due date set to that of the Sprint Retrospective meeting. When an issue is added to a Sprint, it should be added to that Sprint's milestone.

³<https://waffle.io/soton-ecs-2014-gdp-12/videogular-questions/>

C.6 JavaScript Coding Conventions

Strict mode should be used (`'use strict';`). Semicolons should never be omitted.

In object definitions, trailing commas should always be used. For example:

```
1 obj = {  
2   foo: 'bar',  
3   baz: 'quux',  
4 }
```

C.6.1 Indentation

Indentation should be done with tabs up to the indent level, and then spaces for lining up multi-line statements. For example (where a `>` is a tab and a `.` is a space):

```
1 function foo() {  
2 >   if (bar === 4) {  
3 > >     baz("Some really ridiculously long string that should be "  
4 > >       ....+ "avoided, even mentioned in the coding conventions.");  
5 >   }  
6 }
```

This allows other developers to choose indent sizes without messing up neatly lined-up parts.

C.6.2 Naming

Names should be camel case. The first letter should be lower-case, except for class names or constructors:

```
1 LightBulb = (function() {  
2   function LightBulb() {  
3     // ...  
4   }  
5  
6   return LightBulb;  
7 })();  
8  
9 var numberOfEngineers = 5;  
10 function changeLightBulb(bulb) {  
11   // ...  
12 }
```

File names should still be in Unix form (e.g. `light-bulb.js`). Unit test files should have the same name as the file which they test, followed by `_test` (e.g. `light-bulb_test.js`).

C.6.3 Types

The section of JavaScript Garden on Types ⁴ gives a number of guidelines (in the conclusion paragraphs) which should be followed.

C.6.4 JSHint

JSHint directive comments should be kept to a minimum, with configuration moved into the `.jshintrc` file where possible. If file-specific configuration is necessary, it should go at the top of the file (before `'use strict';`) unless:

- it is specific to a section of the file, or
- it describes the changes made to the environment by a particular line of code, for example by the `importScripts` method in a [WebWorker](#). In this case the directive should be on the next line, indented. For example:

```
1 importScripts("../app/bower_components/videogular-questions/  
  questions-worker.js");  
2 /* global loadAnnotations */
```

C.6.5 AngularJS

Dependency Injection

When defining a directive, service, view, controller, etc., and dynamically injecting dependencies, make sure to pass the parameter name as a string into the array. For example:

```
1 angular.module("com.example.fooBar", [])  
2   .directive(  
3     "ngFooBar",  
4     ["$window", "VG_STATES", function($window, VG_STATES) {  
5       ...  
6     }])
```

This stops minifiers from breaking the dependency information when they rename parameters.

⁴<https://bonsaiden.github.io/JavaScript-Garden/#types>

C.7 LaTeX

C.7.1 TODOs

For todo notes, use the `\todo` command. For example:

```
1 \todo{Discuss farming in the Middle East.}
```

C.7.2 References

When making references, use `\autoref`, like so:

```
1 \autoref{my-label}
```

C.7.3 Labelling Scheme

Figure labels should begin with `Figure:`, chapter labels with `Chapter:`, and section labels with `Section:`.

C.8 Python Coding Conventions

PEP8 ⁵ should be followed.

C.8.1 File Names

File names should be python style, that is, words separated by underscores (e.g. `voluminous.octopus.py`).

When a virtual environment is used, it should be called `venv`. Any dependencies should be defined in `requirements.txt`.

⁵<http://legacy.python.org/dev/peps/pep-0008/>

D

Repositories

An appendix detailing the main contributors to each project and the GitHub location of the source code repository.

*“I was going to tell a joke about svn, but I don’t think anyone would
git it.”*

- Unknown

D.1 Introduction

All of the source code for this project was managed on GitHub. Each section in this appendix relates to a repository used during the project. For each, the primary contributors have been named.

The appendix is split into deliverables, examples and auxiliary repositories based on their usage.

D.2 Deliverables

These repositories are modules that the client requested.

D.2.1 Videogular Questions

A plugin for the [Videogular](#) video player that allows the display quizzes and polls during the playback of videos

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/videogular-questions.git>

Main Contributors

- Christopher Baines
- Christopher Hewett
- Samuel Bennett
- Harry Cutts

D.2.2 Videogular Analytics

A [Videogular](#) plugin to report back analytics events to an analytics server for later review.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/videogular-analytics.git>

Main Contributors

- Christopher Baines
- Harry Cutts
- Christopher Hewett

D.2.3 Videogular Cuepoints

Videogular plugin for displaying marks on a video's scrub bar.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/videogular-cuepoints.git>

Main Contributors

- Harry Cutts

D.2.4 Videogular Heatmap

Videogular plugin to display a heat map representing arbitrary data in the videos scrub bar.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/videogular-heatmap.git>

Main Contributors

- Maria Lynch

D.2.5 Authoring Tool

A website that allows you to create [Definition File](#) for use with the Videogular Questions plugin.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/authoring-tool.git>

Main Contributors

- Christopher Hewett
- Maria Lynch
- Christopher Baines
- Harry Cutts

D.3 Examples

These repositories are examples that demonstrate what can be done with the deliverables. They have also been used to test the deliverables are working correctly.

D.3.1 Example Results Server

An example website illustrating a possible implementation of the results server for the Videogular Questions results data storage.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/example-poll-backend.git>

Main Contributors

- Christopher Hewett
- Samuel Bennett

D.3.2 Example Analytics Site

This is an example site demonstrating the usage of the Videogular Analytics and Videogular Heatmaps plugins.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/example-analytics-backend.git>

Main Contributors

- Christopher Hewett
- Maria Lynch
- Christopher Baines
- Harry Cutts

D.3.3 Videogular Questions Example Site

An example website illustrating the Videogular Questions plugin.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/videogular-questions-example.git>

Main Contributors

- Christopher Hewett
- Harry Cutts
- Christopher Baines
- Samuel Bennett
- Maria Lynch

D.4 Auxiliary

These repositories contain non code related material relevant to the project.

D.4.1 Reports

This contains source code for the project reports.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/reports.git>

Main Contributors

- Christopher Hewett
- Maria Lynch
- Samuel Bennett
- Christopher Baines
- Harry Cutts

D.4.2 Presentations

Progress reports given at each progress presentation

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/presentations.git>

Main Contributors

- Christopher Hewett
- Christopher Baines
- Maria Lynch
- Harry Cutts
- Samuel Bennett

D.4.3 Conventions

This holds the coding conventions. Harry Cutts was appointed as the owner of this repository and kept the conventions in order.

Repository available at:

<https://github.com/soton-ecs-2014-gdp-12/conventions.git>

Main Contributors

- Harry Cutts

Minutes of Meetings

Appendix

E

An appendix containing the minutes of meetings held during the project.

E.1 Introduction

This appendix contains the minutes of meetings held with the group's supervisor, client and second examiner, as recorded by Harry Cutts. It also contains minutes for the Sprint Retrospectives, where they happened.

Unless otherwise stated, all five group members attended all meetings. Mike Wald attended all client/supervisor meetings as both client and supervisor.

E.2 29th September 2014

E.2.1 Client/supervisor meeting

Yunjia Li (lead developer on Synote) also attended.

History of Synote

Mike Wald started by giving a brief history of Synote.

The 2008 version (which is the version running on synote.org) was built with Google Web Toolkit and Grails.

The 2010 version replaced the front-end, but re-used the back-end. A mobile interface was later added to this version.

A version for the Raspberry Pi was produced.

The new version of Synote is to be built with a Node.js back-end, which will interact with an AngularJS front-end via a [Representational State Transfer \(ReST\) Application Programming Interface \(API\)](#). The code is to be hosted on GitHub.

Our task

Our task is to work on software for interactive video quizzes to be integrated with Synote at a later date. The solution must be accessible. We may wish to use the [Question and Test Interoperability \(QTI\)](#) standard for defining quizzes.

This has been done before by a Masters student called Nadia, but her work was not compatible with AngularJS. Mike will send Nadia's implementation and report to us. There is also an existing AngularJS library for interactive video quizzes, which we could fork.

Other requirements:

- Use ReSTful [APIs](#) to connect the front-end and back-end
- Define a data structure which we can statically serve for the proof-of-concept
- Consider mobile browsers
- Make proper tests
- We'll need to choose a framework
- Later in the project, create a quiz authoring tool

The project should be managed in an agile way. Harry suggested that we follow Scrum. Yunjia recommended the RallyDev application for issue management.

We will hold these meetings with Mike regularly at 14:00 on Mondays. The group will meet after the GDP briefing on Wednesday.

E.2.2 After-meeting

Sam suggests that we create a very basic prototype and then ask whether this is what they want.

Harry suggested not scheduling anything over Christmas.

We should find some users and talk to them

Scrum (with 1 week sprints) would be good to try.

We each should:

- Read Nadia's report
- Have a look at the AngularJS Video Quiz module (Chris will email us the link)

Harry will create the Google Drive folder, upload the minutes, create a GitHub organisation and add everyone in the group.

E.3 6th October 2014

E.3.1 Client/supervisor meeting

Shameem Bajar, a third-year student of Information Technology in Organisations, also attended.

We presented a prototype system built on [Videogular](#) over the last week.

Videogular Quiz wasn't worth building on.

Shameem is doing a third-year project, and will be interacting with users.

Aspects which we haven't covered in our prototype:

- Jumping back to sections when questions are answered incorrectly
- Analytics on user behaviour around the quiz

A big use case for this project is for the live version. Another is for [Massive Open Online Courses \(MOOCs\)](#).

Our technical goals are to overlay quiz questions and polls over online videos, with analytics support.

Over the next week, we will:

- write the brief (sending a draft to Mike),
- make the prototype code more robust, and
- test the prototype on multiple platforms, building up a list of issues.

As soon as possible, we will get a demo working which can be used by Shameem to get user feedback and stories.

E.3.2 After-meeting

We could draw some state machines of possible quizzes.

Some questions for us to answer:

- Would MediaElement.js be worth investigating?
- Is Videogular's accessibility up to Mike's standards?

E.4 13th October 2014

E.4.1 Client/supervisor meeting

We demonstrated our progress on the prototype, including it running on a phone and a tablet. Mike reiterated that the user should be able to jump back in the video when a question is answered incorrectly.

We asked how we are to acquire realistic data with which to test the analytics system. Mike rejected our selection of him using a prototype system for a lecture, but suggested that Shameem could use focus groups to acquire data.

He also clarified that the actual analysis of the data is basically as a proof-of-concept. The important part is to have the front-end collect the data (in the form of event logs).

We attempted to demonstrate some accessibility improvements which we have made and contributed to Videogular, but they had not yet been deployed on the demonstration machine.

We asked whether we could access the code for the new version of Synote, and learned that Yunjia hasn't started on the new Synote yet.

E.5 20th October 2014

E.5.1 Client/supervisor meeting

We demonstrated:

- Videogular accessibility improvements
- Skipping back in the video when a wrong answer is given
- The new stars question type

We explained the architecture on which we had decided for the questions front-end, in which a [WebWorker](#) (a sandboxed JavaScript thread) is used to execute potentially unsafe quiz definitions. This allows us to use JavaScript to define quizzes in a flexible manner without a complicated markup language.

Mike said he would think about organising a focus group to collect data from.

We reported on our first sprint of the project.

We said we would prepare a stable demo for Wednesday, and give the link to Shameem for user feedback.

The presentation

We discussed how to pitch the idea of our project in the up-coming progress presentation.

Chris Hewett suggested we 'sell' the idea of teaching in small chunks and testing on each individual chunk. Small chunks can be revisited easily, but splitting a video into those

small chunks is very time-consuming. Instead, our project will allow questions to be inserted at the end of each chunk in a longer video.

E.5.2 Sprint retrospective

In the traditional Scrum way, we discussed how well our first sprint had gone.

The good

- We got everything that was assigned in the planning meeting done.
- We communicated well.

The bad

- We underestimated the amount of work we could get done.
- Our points allocations were out of whack quite a bit.

E.6 27th October 2014

There was no meeting this week as Mike was away.

E.6.1 Retrospective

We didn't all make our points targets this week. This was mostly due to the presentation.

Poll and Cuepoints issues weren't well defined enough.

Chewett is blocking on lack of documentation and schemas when creating examples. We should create issues to finalise schemas.

E.7 10th November 2014

E.7.1 Client/supervisor meeting

Yunjia Li and Shameem Bajar also attended.

We demonstrated:

- Charts showing results of in-video polls.
- The heat map plugin for Videogular.
- Our Videogular Analytics plug-in acquiring analytics data, and it being processed into a heat map.

We mentioned that we need to use HTML5 events (including seek) instead of watches to detect changes in the video state. We also need to display the heat map data using the plug-in we've created for Videogular.

Yunjia raised concerns about the perceived difficulty of deploying our server-side code. We said we would write a document describing the deployment process to an Apache server.

Shameem's study

Shameem presented some initial results of her user studies. The presentation will be sent to us shortly.

Feedback on the idea was positive. No users mentioned mobile devices, except that a participant would submit poll responses from a phone during a lecture (which is out of the scope of our project).

Many suggestions were made by users, some of which were out of scope. The following were in scope:

- Add number of attempts field to authoring tool.
- Give a grade at the end of the quiz.
- Recommend further videos on the same topic after the video is finished (which we could display as custom HTML on a results page).

We discussed the upcoming presentation, during which we are planning to demonstrate the analytics view.

E.7.2 Retrospective

- We have inconsistency between tabs and spaces for indentation in some files.
- Some external libraries are stored inconsistently (e.g. Bootstrap is a submodule in the analytics back-end, d3 is a file in the repository).
- We can use `bower link` instead of the shell script for using Bower with modules which are in development. This will be done in the next sprint.

E.8 17th November 2014

E.8.1 Sprint retrospective

- Number of points per person was good

E.8.2 Client/supervisor meeting

We demonstrated the authoring tool UI which has been designed, but not made functional.

We showed the presentation planned for Wednesday. Mike suggested to mention the interactive features such as skipping back in the video.

E.9 24th November 2014

E.9.1 Meeting with Gary Wills

- Report is what Gary's going to mark
- UML in design section
- Why didn't we present it to users for evaluation?
 - The client (Mike) thinks that it is not viable.
- Write about testing and/or scenario based testing instead.
- Could evaluate based on "How well does this output suit your requirements?"
- Doing team skills audit will improve mark
- Don't include 'if we have time' things in goals
- Should have a consistent voice throughout the report
- We **should not** be working through Christmas
- Manage the client (Mike)
- "The key to good assessment is instant feedback."
- Go through report with fine-tooth spelling and grammar comb before submitting report.

E.9.2 Client/supervisor meeting

Mike said he was generally happy with our latest progress presentation.

Looking at the authoring tool:

- It would be good to make single choice question type a special case of multiple choice question.
- A “Load current time” button next to the time selector would be useful.

We should mention that (some) server implementations are example only and have no scalability guarantees.

E.10 1st December 2014

E.10.1 Client/supervisor meeting

Yunjia Li also attended.

We demonstrated the authoring tool.

Yunjia suggested something for the Further Work report section: making encoding the video for different browsers/platforms more user-friendly.

Chewett proposed a list of deliverables:

- Videogular Questions
 - Example proof-of-concept site (Videogular Questions Example)
- Videogular Cuepoints
 - As demonstrated is Videogular Questions Example
- Videogular Heat Maps
- Videogular Analytics
 - API specification for Videogular Analytics
 - Analytics back-end is an example only
- Authoring tool

Mobile browsing will be part of future work.

Mike approved that list of deliverables.

E.11 8th December 2014

E.11.1 Client/supervisor meeting

Yunjia Li also attended.

Sign off on deliverables

The deliverables had been sent to Mike last Thursday. He had found that the heat map did not appear in the analytics interface in Google Chrome. We clarified a misunderstanding about the vertical scales on results charts.

Yunjia asked about the status of the seek event in the analytics plug-in, which has not been implemented but should not be problematic.

Mike and Yunjia said that they were satisfied with the deliverables. Yunjia said they were ‘pretty cool’.

Mike said he’d be happy to comment on draft report stuff we send him.

E.11.2 After-meeting

We discussed the report structure, and decided to put all the testing in one section (with subsections for each component), to emphasize the range of testing methods we used.

Gantt Charts

An appendix detailing the Gantt charts that were initially created and the revised second draft based on how the project actually went.

“Even if you are on the right track, you will get run over if you just sit there.”

- Will Rogers

F.1 Initial Plan

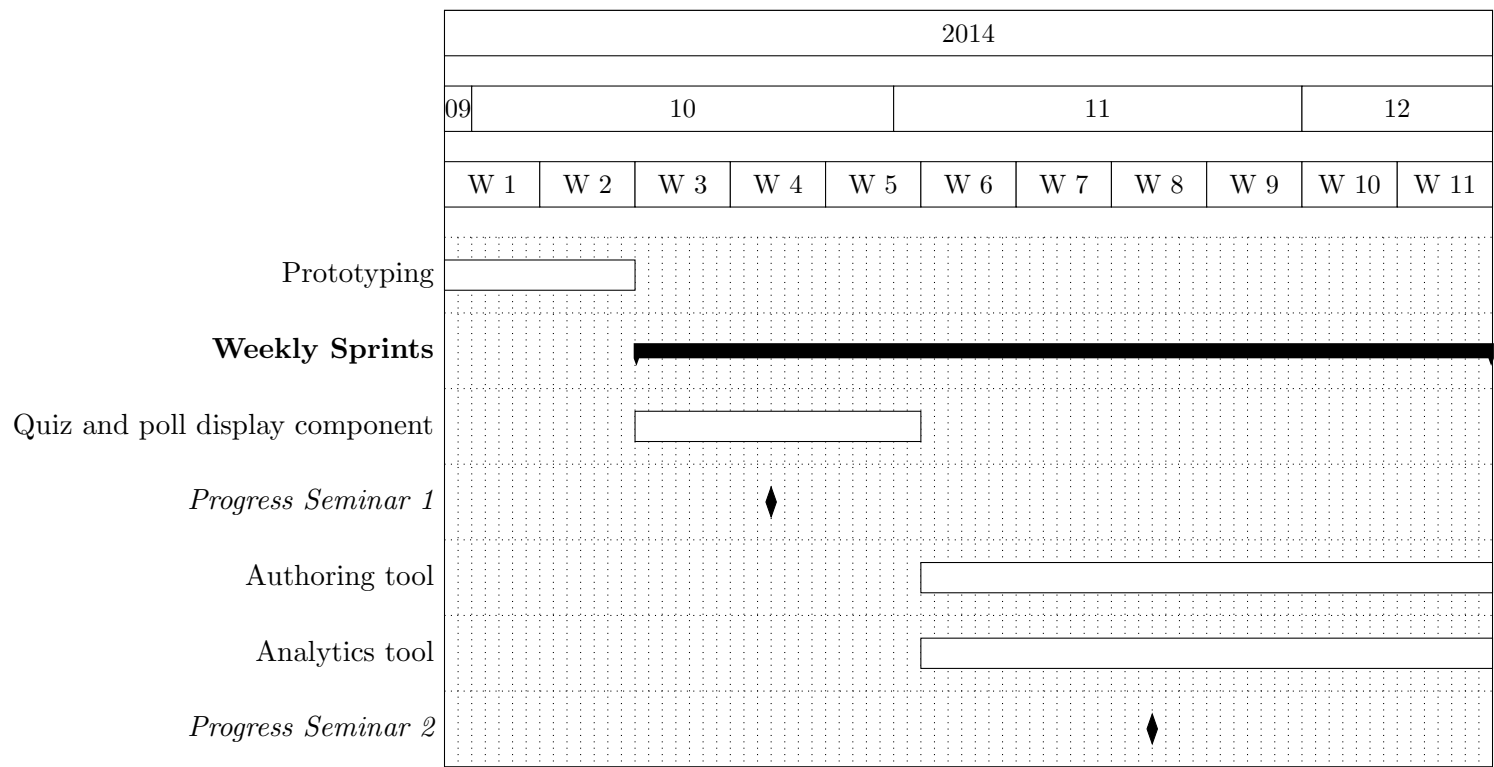


Figure F.1: Gantt Chart prior to the Christmas vacation.

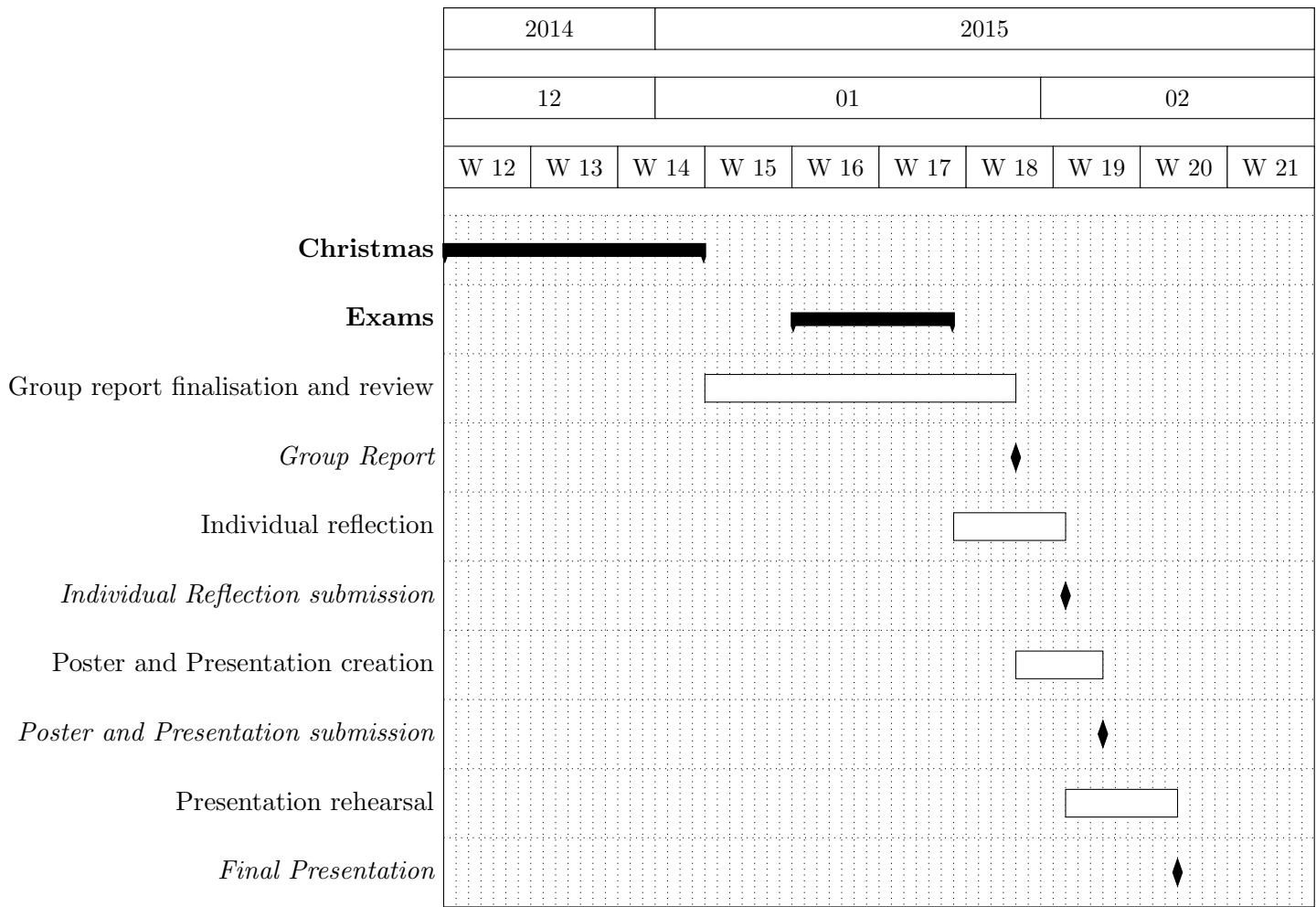


Figure F.2: Gantt Chart after the Christmas vacation

F.2 Actual

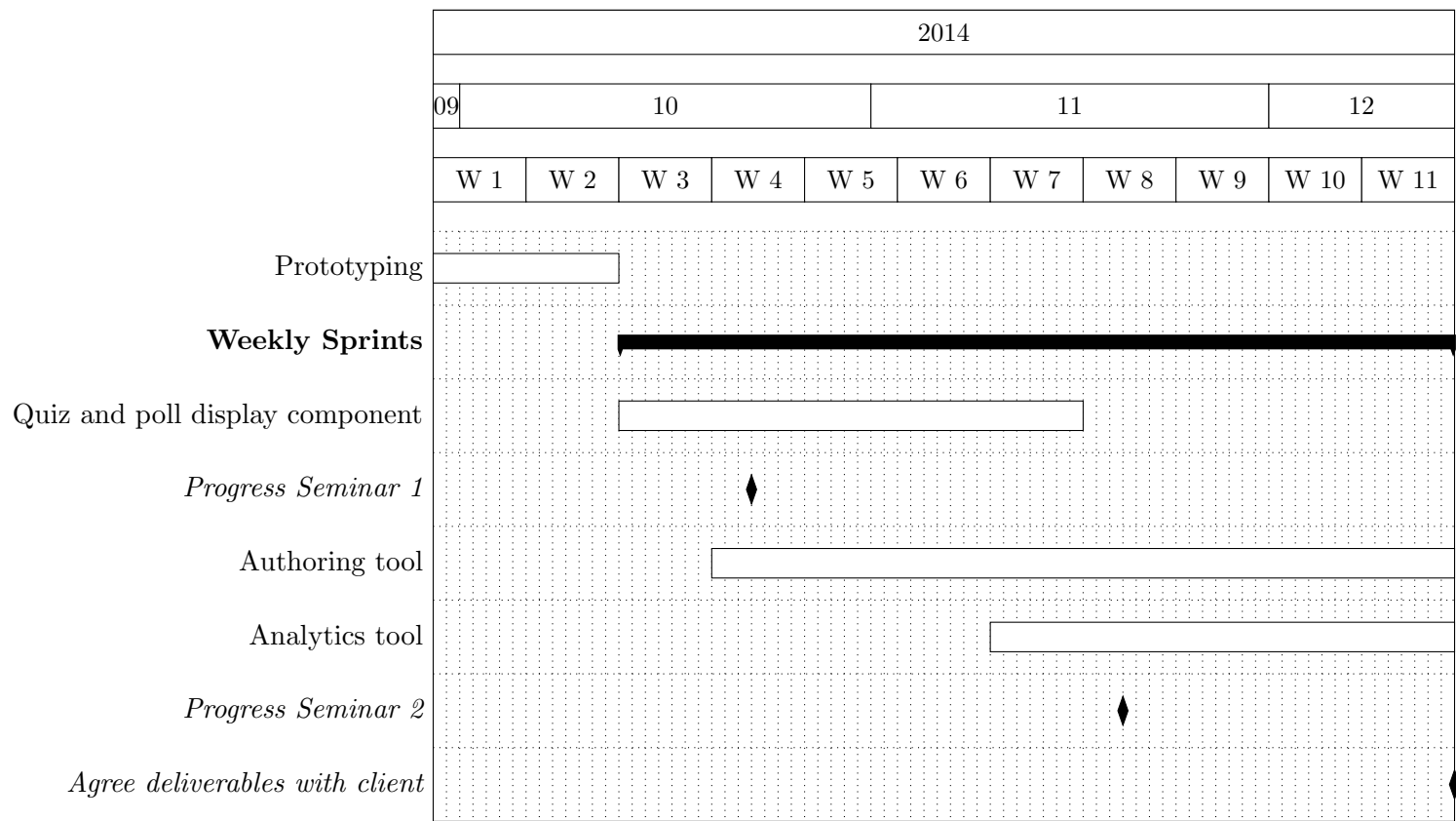


Figure F.3: Gantt Chart prior to the Christmas vacation.

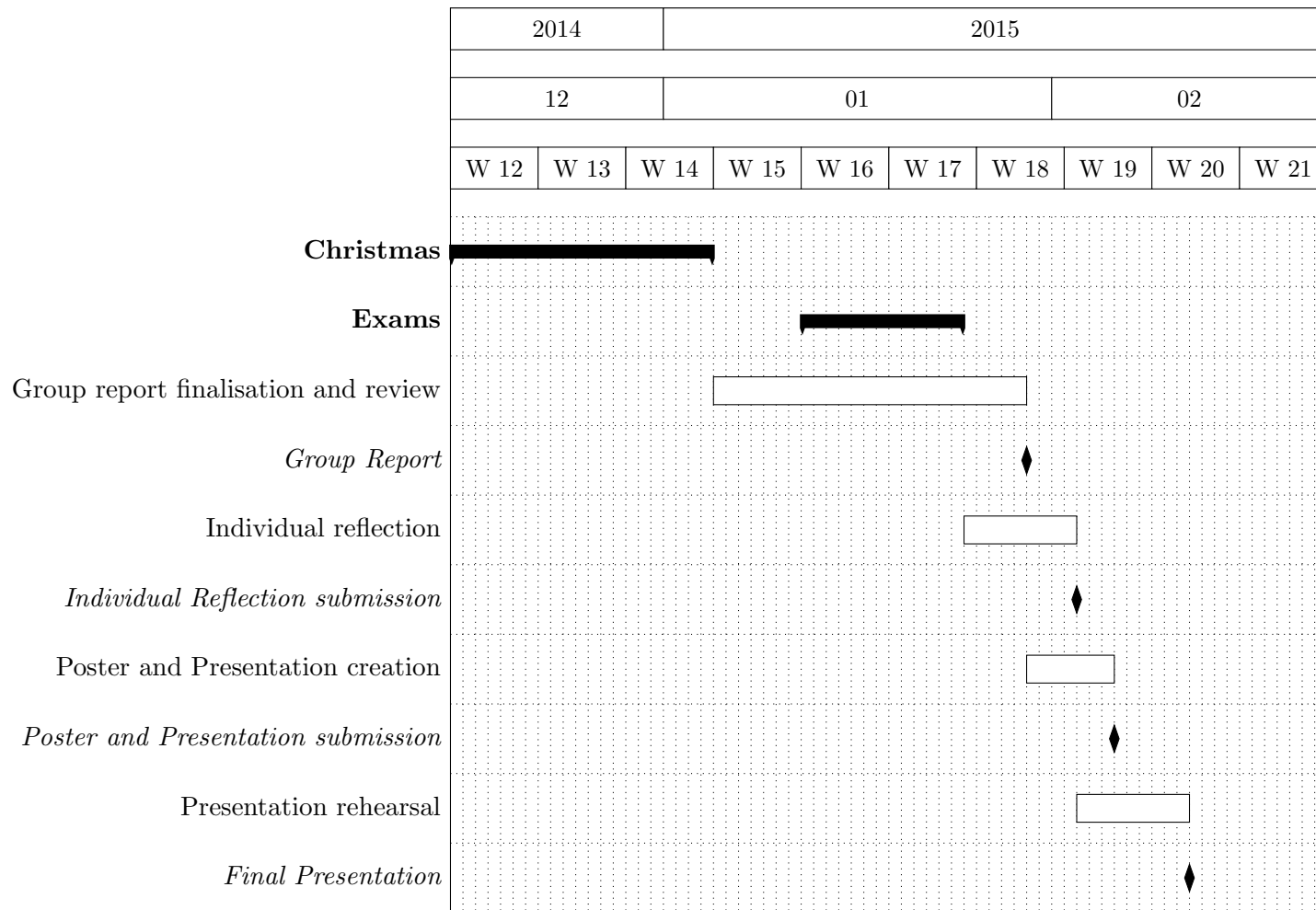


Figure F.4: Gantt Chart after the Christmas vacation

Report Authorship

Appendix

G

An appendix detailing the main contributors to each section of the report.

G.1 Introduction

All members worked on all parts of the report, this appendix details the primary authorship of each section.

G.2 Authorship

The primary contributors for each section have been ordered by the size of their contributions.

Introduction

Written by Maria, Samuel

Previous Work

Written by Maria, Harry

Overall Approach

Written by Maria, Chris B, Chris H

Project Management

Written by Samuel, Chris H

Video Player

Written by Harry, Maria

Videogular Questions

Written by Chris B, Maria

Authoring Tool

Written by Chris H, Samuel

Analytics

Written by Chris H, Chris B

Testing

Written by Chris B, Chris H, Harry, Maria

Future work

Written by Chris H, Maria, Samuel

Conclusions

Written by Maria, Chris H, Chris B, Samuel

Appendices

Written by all group members

Diagrams

Created by Samuel, Chris B, Chris H

Document Design

Designed by Samuel and Maria

Textual Editing

Edited by Harry and Maria

Management

Managed by Samuel, Maria

Authoring Tool Wireframes

Appendix

H

An appendix showing the initial authoring tool wireframes. These were used as examples of the interface was planned until the HTML views were created and iterated upon.

H.1 Introduction

Wireframes were created to allow for feedback from users and the client regarding potential designs for the Authoring tool. These initial wireframes were presented to Shameem for the first user evaluations to give a rough idea of what would be designed.

H.2 Authoring Tool Primary User Interface

When designing the user interface for the authoring tool there were initially had two main concepts. There were to be a large amount of settings to configure when using the tool and overloading the user was to be avoided ([Requirement N8](#)). Therefore it was planned to hide most of the options initially.

This led to deciding on two different options for the hiding of this interface.

H.2.1 First Option - Accordion

The first option considered was using an accordion view shown in [Figure H.1](#). Here the options for each part of the question sets are shown in accordions that appear as the question set is edited.

This meant that only the options for the current question you were working on were shown but that you could switch between different questions to review the information.

H.2.2 Second Option - Pop ups

A second option was considered where you would press an edit button and the options would pop up. Here you originally see the main screen shown in [Figure H.2](#) and then once you select an options button such as the “advanced option” button, you would see the popup shown in [Figure H.3](#).

The wireframe illustrates the Quiz Authoring Tool interface, which is organized into several functional areas:

- Header:** The title "Quiz Authoring Tool" is centered at the top. Below it, a "Video source:" label is followed by a text input field containing the URL "http://www.youtube.co.uk/".
- Video Player:** A large rectangular area on the left contains a video player. It features a play button in the center, a progress bar at the bottom with a red playhead, and standard video controls (volume, full screen, etc.) on the right.
- Video Default Options:** A panel below the video player contains settings for questions and polls.
 - Question:** Includes checkboxes for "Question can be skipped always", "Question can be skipped if answered correctly" (checked), "If answered incorrectly return to" (with a time input), "Show correct answer once submitted", and "Notify user if they try to skip past a question" (checked).
 - Poll:** Includes checkboxes for "Poll can always be skipped", "Poll can be skipped if answered previously" (checked), and "Notify user if they try to skip past a poll" (checked). It also has a "Show responses:" section with radio buttons for "After each poll" (selected) and "After question set".
- Export Quiz:** A large button at the bottom left of the options panel.
- Question Set Management:** On the right side, there are buttons for "Add Question Set" and "Change Video". Below these is a "Question Set Start Time:" label with a time input field. Further down are buttons for "Add New Poll", "Add New Question", and "Advanced Options".
- Question List:** A list of question sets is shown, each with a title and "Move up" / "Move down" buttons.
 - Question 1:** Includes a "Question:" label, a text input field, a "Question Type:" dropdown menu (set to "Question type"), and an "Answer panel (dependant on Question Type selected)".
 - Question 2:** A simple title entry.
 - Poll 1:** A simple title entry.
 - Question 3:** A simple title entry.

Figure H.1: Wireframe showing the accordion type design of showing/hiding options

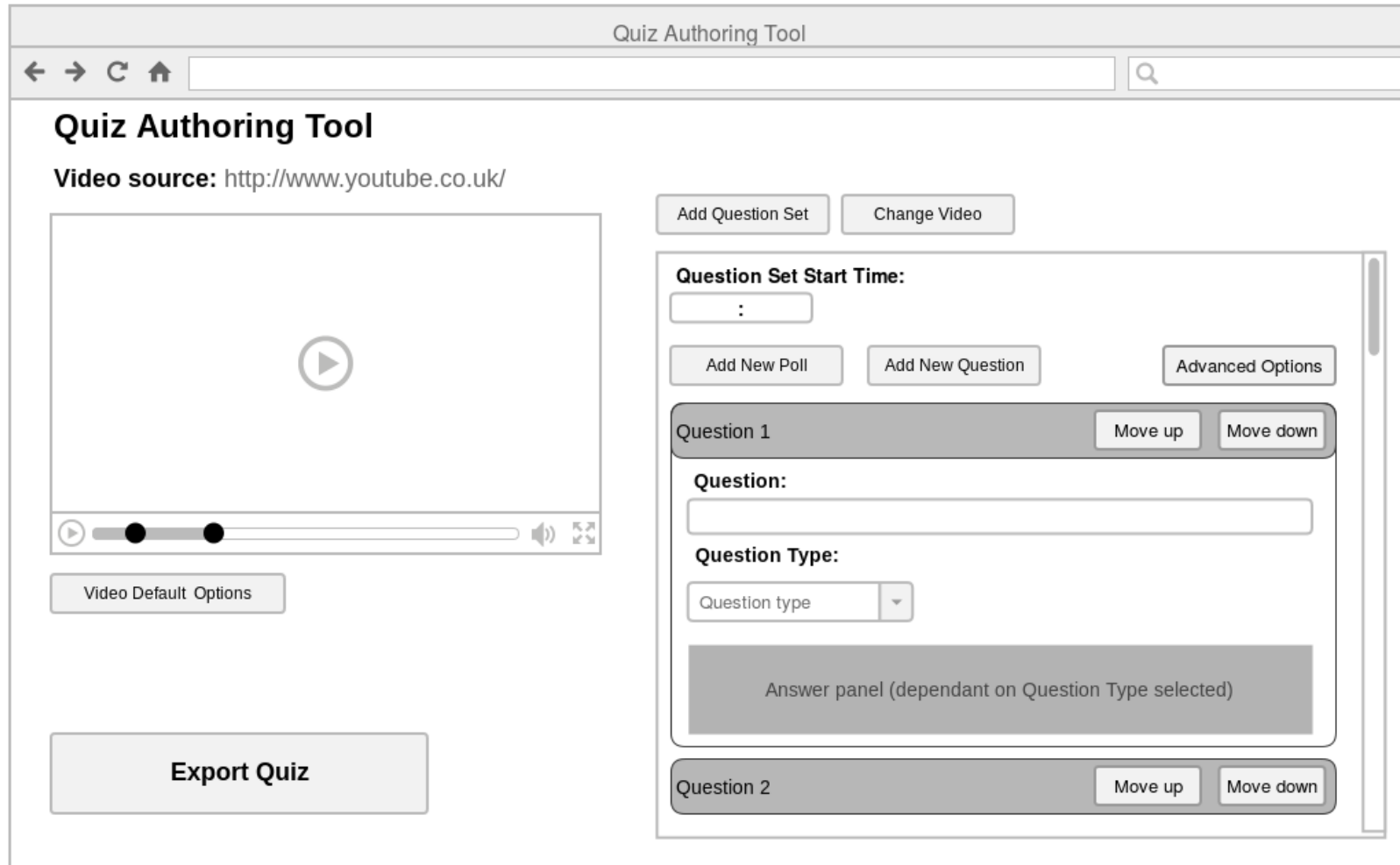


Figure H.2: Wireframe showing the pop up type design of showing/hiding options before opening a pop up.

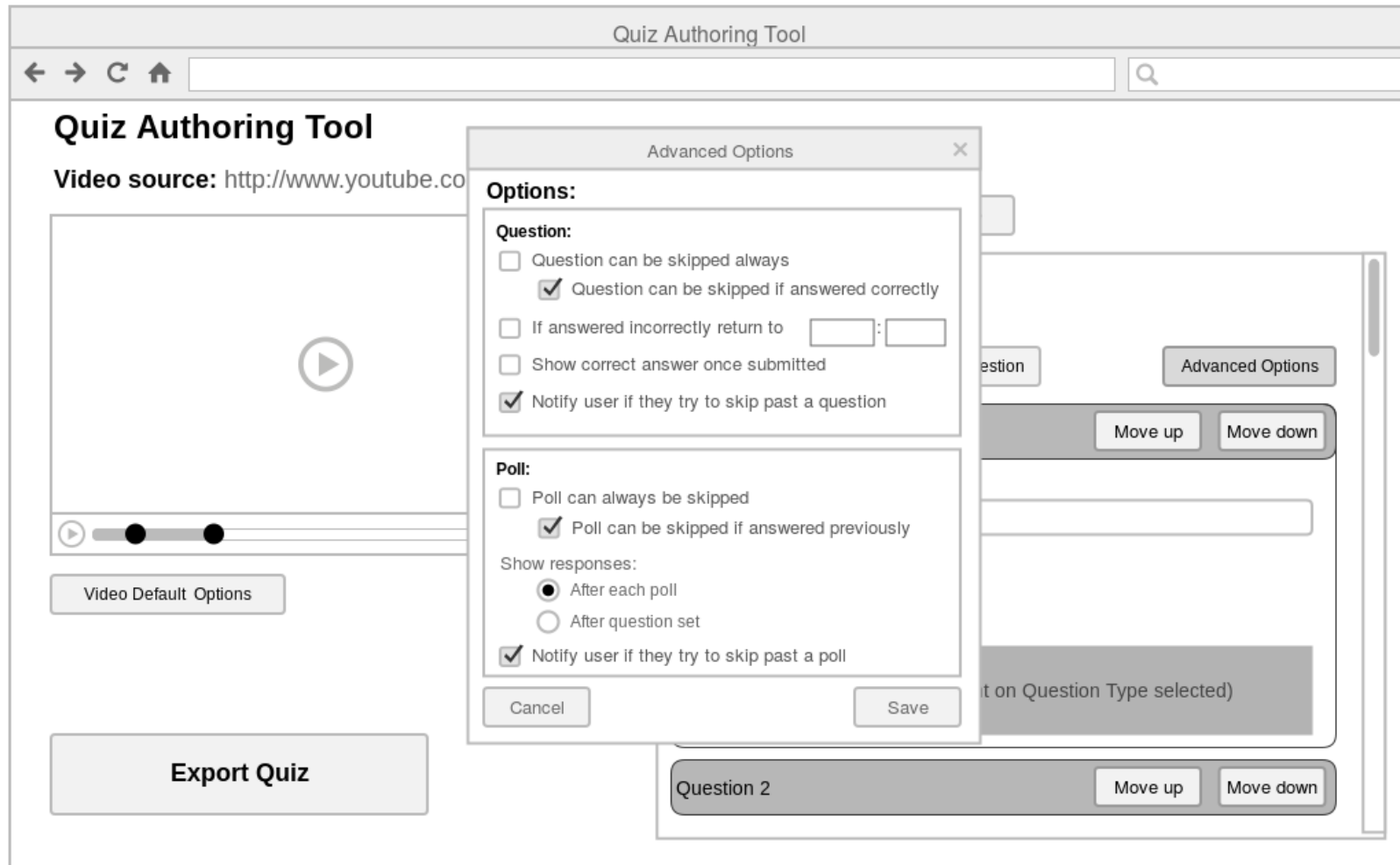


Figure H.3: Wireframe showing the accordion type design of showing/hiding options after opening a pop up.

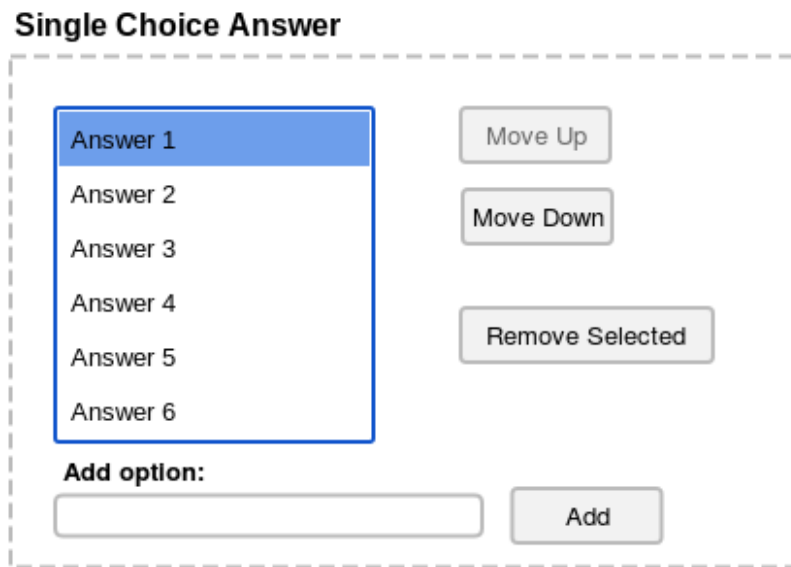


Figure H.4: A wireframe showing the possible interface when creating a single choice poll type question

H.3 Question Creation Wireframes

These first wireframes ([Figure H.4](#), [Figure H.5](#), [Figure H.6](#), [Figure H.7](#), [Figure H.8](#), [Figure H.9](#)) show how the questions were originally planned to look. This demonstrates the differences between the poll and quiz type options, where the quiz type questions allow the choice of a “correct” answer. These were used to generate the first iteration of HTML from the mockups, which were iterated over. This was because it was easier to have more realistic models to present to users and the client.

H.4 Accessibility Tooltips

To aid accessibility the appearance tooltips on some elements was planned. This was illustrated in the example tooltip wireframe [Figure H.10](#).

Single Choice Answer

Answer 1

Answer 2

Answer 3

Answer 4

Answer 5

Answer 6

Move Up

Move Down

Remove Selected

Set Selected as Correct Answer

Add option:

Correct Answer: Answer 2

Figure H.5: A wireframe showing the possible interface when creating a single choice quiz type question

Multiple Choice Answer

Answer 1

Answer 2

Answer 3

Answer 4

Answer 5

Answer 6

Move Up

Move Down

Remove Selected

Add option:

Minimum no of answers selectable:

Maximum no of answers selectable:

Figure H.6: A wireframe showing the possible interface when creating a multiple choice poll type question

Multiple Choice Answer

Answer 1
Answer 2
Answer 3
Answer 4
Answer 5
Answer 6

Move Up
Move Down
Remove Selected
Set Selected as Correct Answer

Add option:

Add

Correct Answers:

Answer 1
Answer 2
Answer 3

Remove Selected

Minimum no of answers selectable:

Maximum no of answers selectable:

Figure H.7: A wireframe showing the possible interface when creating a multiple choice quiz type question

Scale/Rating Answer

☒
☐

☐
☐
☐
☐
☐
☐

Figure H.8: A wireframe showing the possible interface when creating a range or star poll type question

Scale/Rating Answer

☒
☐

☐
☐
☐
☐
☐
☐

Figure H.9: A wireframe showing the possible interface when creating a range or star quiz type question

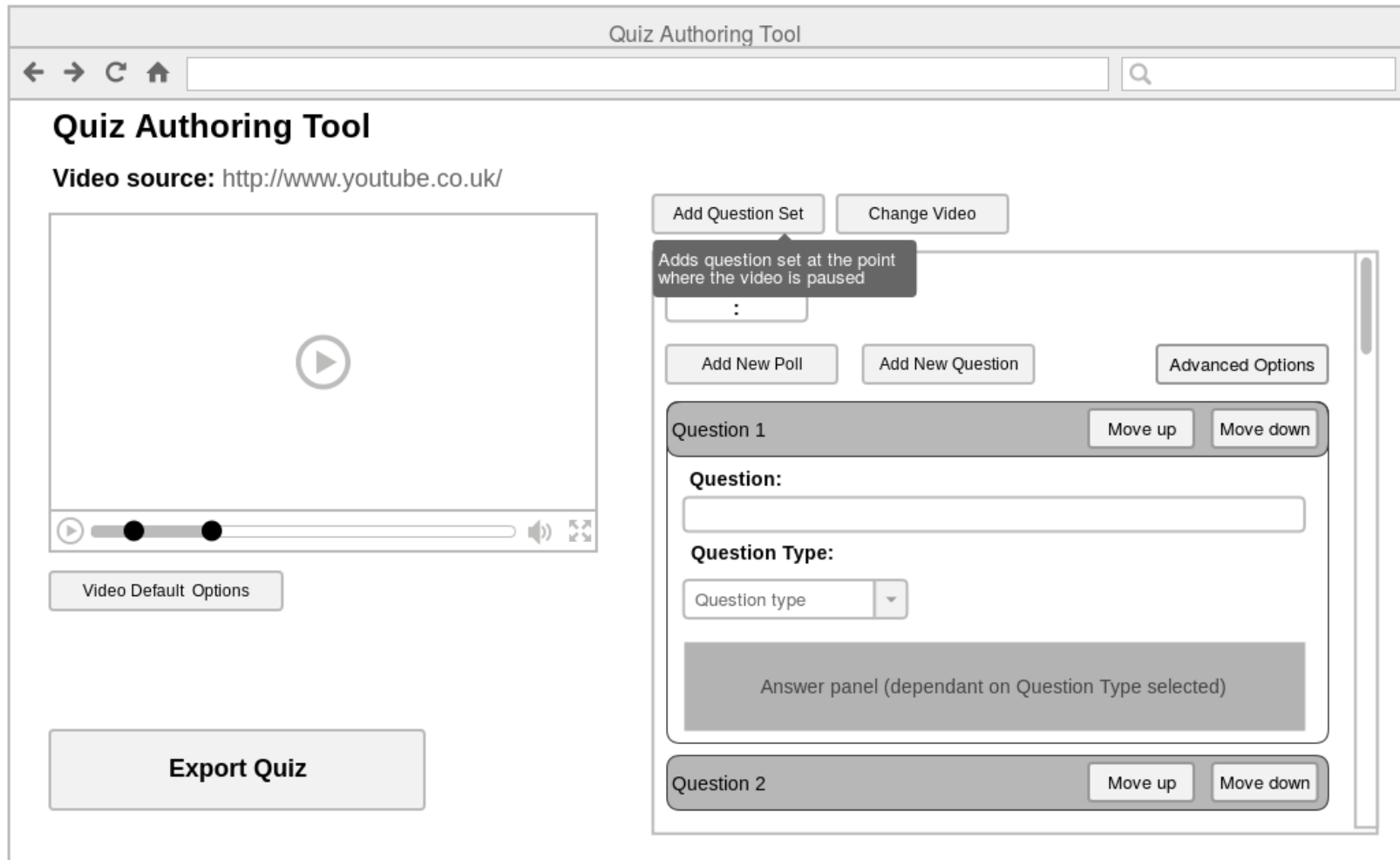


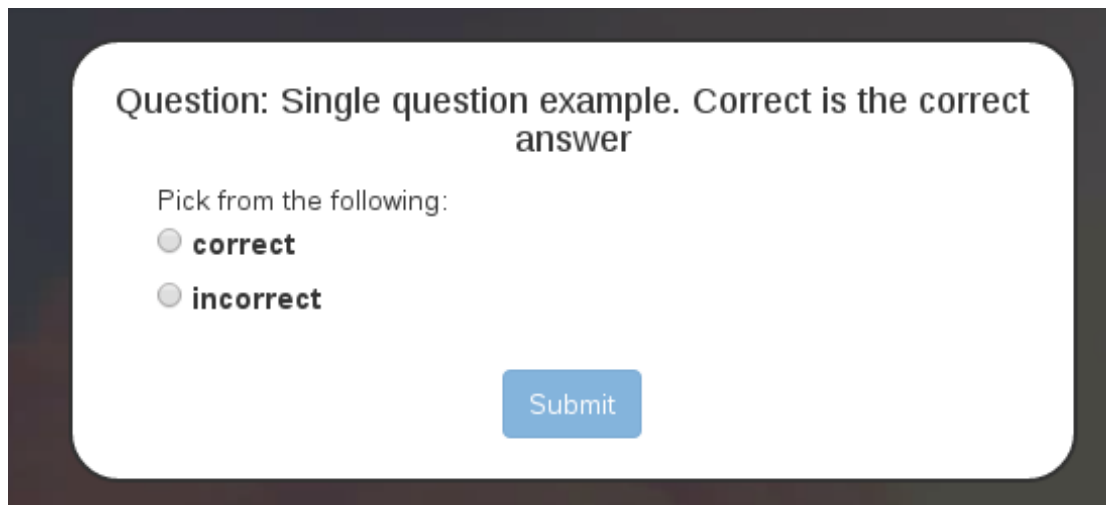
Figure H.10: A wireframe showing an example of how tooltips could be implemented

Question Mockups

Appendix

I

An appendix detailing the question mockups that were used during the user study.



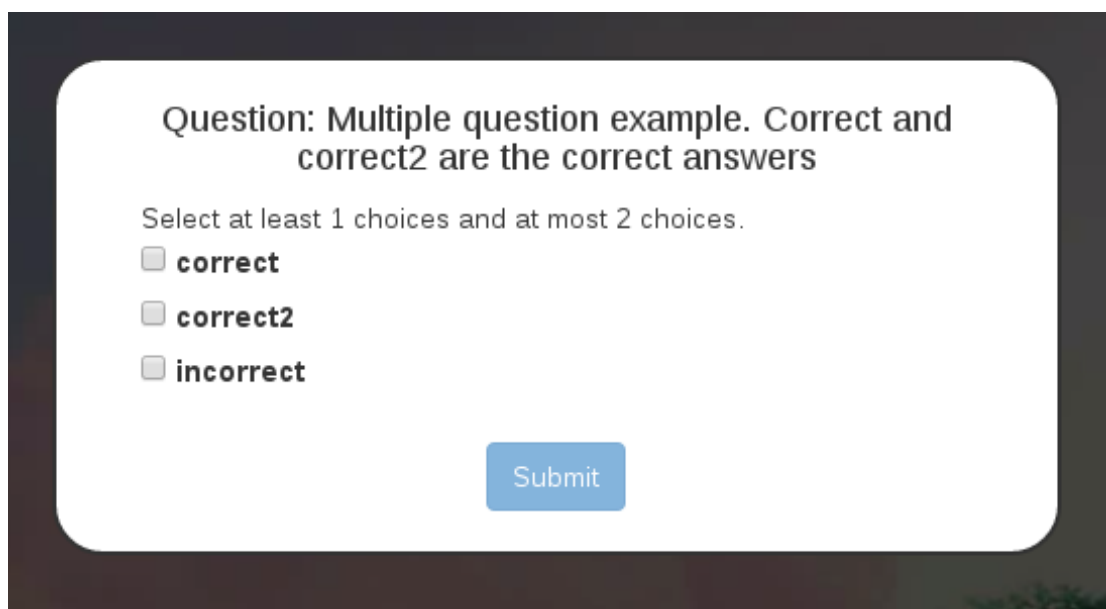
Question: Single question example. Correct is the correct answer

Pick from the following:

- ☐ **correct**
- ☐ **incorrect**

Submit

Figure I.1: An example of a single answer question



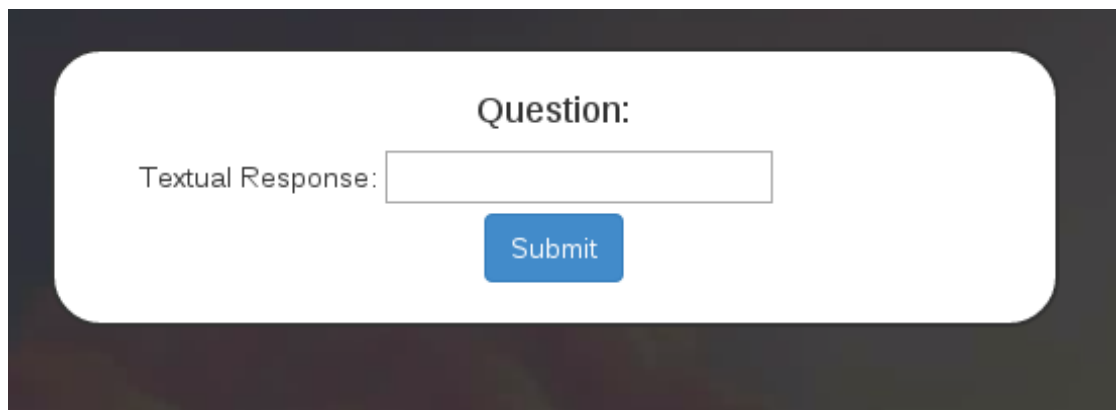
Question: Multiple question example. Correct and correct2 are the correct answers

Select at least 1 choices and at most 2 choices.

- ☐ **correct**
- ☐ **correct2**
- ☐ **incorrect**

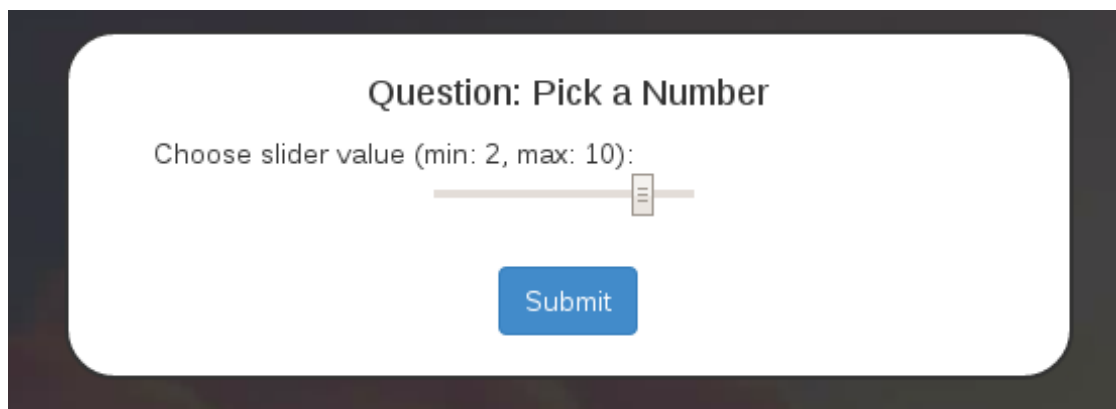
Submit

Figure I.2: An example of a multiple answer question



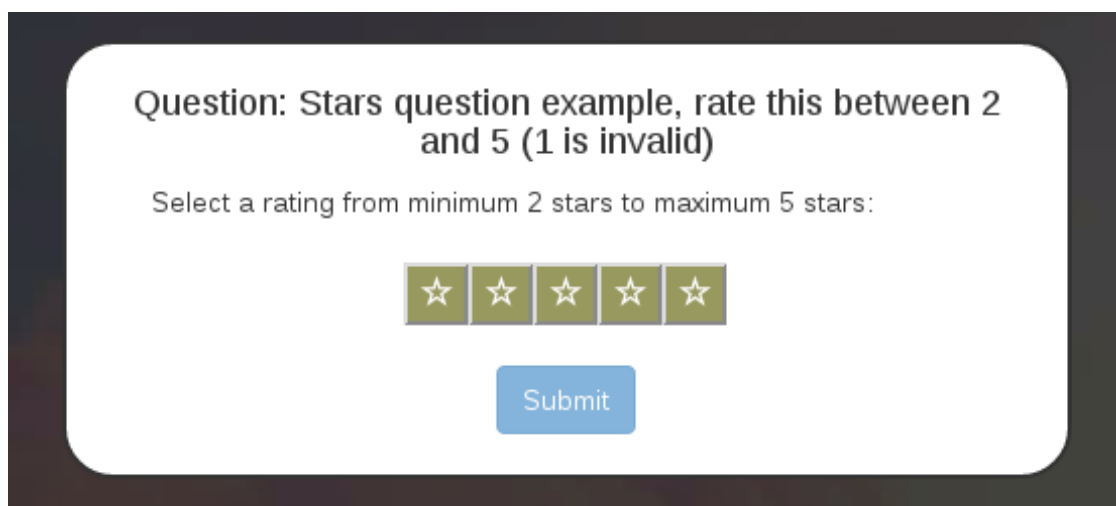
A mockup of a text type question interface. It features a dark gray background with a white rounded rectangle in the center. Inside the rectangle, the word "Question:" is centered at the top. Below it, the text "Textual Response:" is followed by a white rectangular input field. Underneath the input field is a blue button with the word "Submit" in white text.

Figure I.3: An example of a text type question



A mockup of a range type question interface. It features a dark gray background with a white rounded rectangle in the center. Inside the rectangle, the text "Question: Pick a Number" is centered at the top. Below it, the text "Choose slider value (min: 2, max: 10):" is followed by a horizontal slider bar with a small square handle. Underneath the slider is a blue button with the word "Submit" in white text.

Figure I.4: An example of a range type question



A mockup of a stars type question interface. It features a dark gray background with a white rounded rectangle in the center. Inside the rectangle, the text "Question: Stars question example, rate this between 2 and 5 (1 is invalid)" is centered at the top. Below it, the text "Select a rating from minimum 2 stars to maximum 5 stars:" is followed by five green star icons arranged horizontally. Underneath the stars is a blue button with the word "Submit" in white text.

Figure I.5: An example of a stars type question

User Study Results

Appendix

J

An appendix detailing results from the user study performed by Shameem Bajar.

- 8 participants:

Southampton, Bristol and Solent university students
6 undergraduate
2 postgraduate
- Mean age of 20
- Most students had experience of using e-learning technologies and possessed a visual and/or auditory learning style.

Activity	Duration
1. GDP Software Demo and Trial	15 minutes
2. Focus group Discussion	30 minutes
3. Individual Questionnaires	15 minutes

Figure J.1: Page 1 of user study presentation

- Over 50% use faculty intranet to access lecture slides
- 100% would watch video-recorded lectures if they were readily available on the intranet

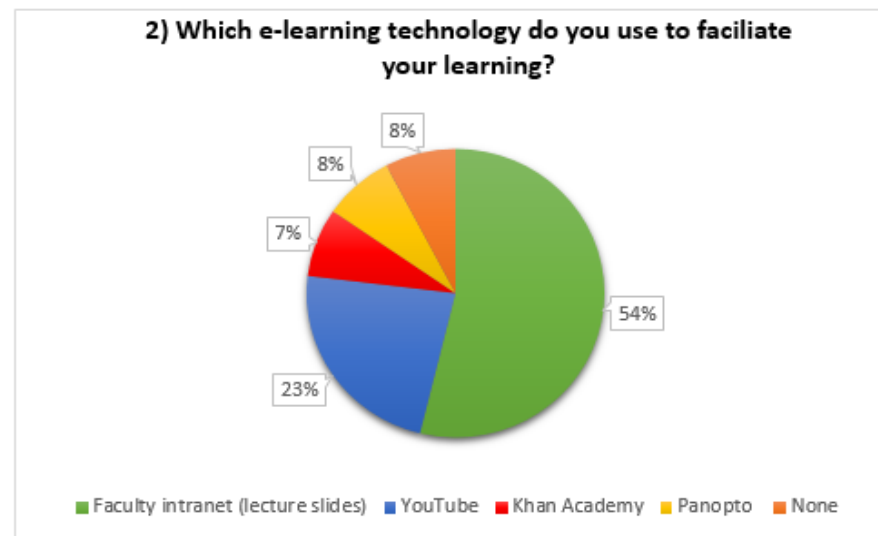
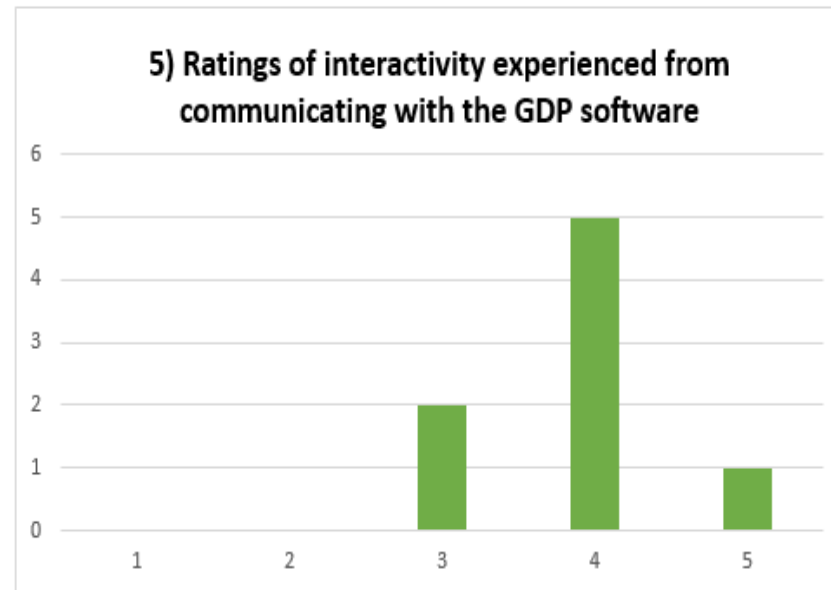


Figure J.2: Page 2 of user study presentation



- The conceptual model fulfils UX goals
- interactive, enjoyable, motivating, engaging, cognitively stimulating and rewarding
- 7 would use regularly

Figure J.3: Page 3 of user study presentation

“how to enhance the level of interactivity of the software, to improve student learning”

No.	Functional Requirements
FR1	The polls and quizzes should allow 2 attempts , then an annotation link should appear on the video allowing students to replay relevant sections of the video lecture.
FR2	Grading system which informs students of their level after the video quiz has been completed, e.g. 1:1, 2:1, 2:2 etc.
FR3	After completing a video quiz, recommended videos should be displayed on the mobile or computer screen, to encourage further learning.
FR4	A forum/comments section which allows students to contribute and discuss material, to encourage collaborative learning and peer feedback.
FR5	Time-frames for subtopics should be highlighted in the video, e.g. YouTube annotation messages, to allow students to skip to relevant sections to learn and assess their understanding.
FR6	Side bar featuring the status of embedded questions , e.g. questions answered, current question and score achieved so far.

Figure J.4: Page 4 of user study presentation

“how to enhance the level of interactivity of the software, to improve student learning”

No.	Type	Non-Functional Requirements
NFR1	Usability	The system should have an intuitive & user-friendly interface ; user functions should be simple to perform.
NFR2	Usability	The design (colours, fonts, images) should be engaging and applicable to the nature of the Synote system and the services it provides.
NFR3	Usability	Visual feedback in the form of a green tick to highlight a correct answer and a red cross to highlight an incorrect answer should be implemented to increase motivation for learning.
NFR4	Performance	Videos should load efficiently, within 5 seconds .
NFR5	Compatibility	The system should be accessible on all main browsers and operating system platforms .
NFR6	Security	The system should provide a unique user space for each user to access securely, via a username and password.

Figure J.5: Page 5 of user study presentation

These results were presenting during a client meeting by Shameem, during this some of the requirements that were apparent after the user study were reviewed.

On reviewing requirements with the client FR1 and FR2 was already implemented but needed some more expansive user code. FR3 and FR4 was considered out of scope by the client. FR5, FR6, and NFR6 were considered a Synote addition that would fit better in Synote. NFR1, NFR2, NFR4, and NFR5 were already requirements given by the client so no changes were needed. NFR3 was considered to be implemented but the client recommended prioritising on the core features before working on this.

Appendix

K

Code Fragments

An appendix with large code fragments not suitable for inclusion in the main body of the text.

*“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”
- Bill Gates*

K.1 Videogular Examples Code Fragments

These code fragments have been taken from the Videogular Questions Examples repository [subsection D.3.3](#).

K.1.1 Videogular Questions Example Question Definition File

[Listing K.1](#) is an example of the Question [Definition File](#) used by the [Videogular Questions](#) repository.

```
1 importScripts("../questions-worker.js");
2
3 loadAnnotations({
4   "first-annotation": {
5     // the time that this annotation will show up
6     // (in seconds from the start of the video)
7     time: 1,
8     items: [
9       {
10        id: "first-question",
11        type: "single",
12        question: "What is the moon made of?",
13        options: [
14          {
15            name: "cheese"
16          },
17          {
18            name: "cheeese"
19          },
20          {
21            name: "cheeeeeeeese"
22          }
23        ],
24        correctAnswer: "cheese"
25      },
26      {
27        id: "check-question",
28        type: "single",
29        question: "Answer incorrect, do you want to review the video",
30        options: [
31          {
32            name: "Yes"
33          },
34          {
35            name: "No"
36          }
37        ],
38        action: function(questions, video) {
39          if (questions.get("check-question").response === "Yes") {
```



```

40         video.setTime(0);
41     }
42 },
43     condition: function(questions) {
44         return questions.get("first-question").isNotCorrect();
45     }
46 }
47 ]
48 }
49 });

```

Listing K.1: Code for loading an annotation

K.2 Authoring Tool Code Fragments

These code fragments have been taken from the authoring tool repository [subsection D.2.5](#).

K.2.1 Authoring tool rewatch template

[Listing K.2](#) is an example of the template used when the user is asked if they wish to review the video.

```

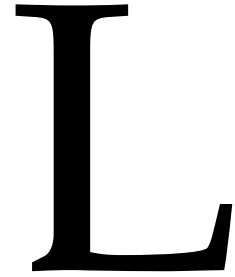
1  {
2      id: "incorrect-QUESTION_ID",
3      type: "single",
4      question: "Answer incorrect, do you want to review the video",
5      options: [
6          {
7              name: "Yes"
8          },
9          {
10             name: "No"
11         }
12     ],
13     action: function(questions, video) {
14         var question = questions.get("QUESTION_ID");
15         if (question.response !== question.correctAnswer) {
16             video.setTime(TIME);
17         }
18     },
19     condition: function(questions) {
20         return questions.get("QUESTION_ID").isNotCorrect();
21     }
22 }

```

Listing K.2: Base template for a question asking if the viewer wishes to review the video section they answered incorrectly

Deployment of Webservers

Appendix



An appendix detailing how each of the tools can be deployed to a website.

*“Intuitive design is how we give the user new superpowers.”
- Jared Spool, Web Site Usability: A Designer’s Guide*

L.1 Introduction

This is divided into the different types of Web server applications that were used. Different Web technologies have been used based on their strengths on what was needed to be produced.

Each repository has a section in the `README.md` file detailing how to set up and install the tool. The individual guides have been reproduced below.

L.2 Deployment of All Projects

To deploy many of these projects they require you to run `npm install` which will also run `bower install`. The specific requirements for each application is defined in the `README.md` file in each repository.

L.3 Deployment of Flask Applications

The only Flask application is the results server therefore this guide is customised to deploy the results server.

To deploy the results server using Apache first you need to clone the repository.

Then the Apache `httpd.conf` file needs to be configured by adding the following entry somewhere in the file:

```
1 <VirtualHost <hostname>:80>
2     ServerName <hostname>
3     WSGIDaemonProcess results_server user=<user> group=<group> threads=5
4     WSGIScriptAlias / <location>/results_server.wsgi
5     ErrorLog logs/results_server-error_log
6     CustomLog logs/results_server-access_log common
7
8     <Directory <location>>
9         WSGIProcessGroup results_server
10        WSGIApplicationGroup %{GLOBAL}
11        Order deny,allow
12        Allow from all
13    </Directory>
14 </VirtualHost>
```

Listing L.1: Apache configuration

Parameters needing changes:

- `<hostname>` is the hostname of the server. e.g. `website.domain.net`
- `<location>` is the location of the source code on the server. e.g. `/var/www/results_server/`
- `<user>` is the user you want the script to run under, by default Apache
- `<group>` is the group you want the script to run under, by default Apache
- The `ErrorLog` and `CustomLog` parameters can be changed to any location

L.4 Deployment of Node.js Servers

The only Node.js application is the example analytics backend therefore this guide is customised to deploy this.

Apache cannot directly run Node.js webservices therefore it needs to be run by Node itself.

This can be run by using `npm start` which will run the Node server. This needs to be running on the server all the time you want the server to be accessible. A web search will return details of how to turn a Node.js webserver into a service however it can just be run from the command line using `screen` or a similar program.

You can configure Apache to proxy any connections to the Node server. This is the suggested method for integrating an Apache server with Node.

To do this `mod_proxy` needs to be installed, web searches will find guides to install this for your chosen operating system.

Then the Apache `httpd.conf` file needs to be configured by adding the following entry somewhere in the file:

```
1 <VirtualHost <hostname>:80>
2     ServerName <hostname>
3     ErrorLog logs/analytics-error_log
4     CustomLog logs/analytics-access_log common
5
6     ProxyRequests off
7
8     <Proxy *>
9         Order deny,allow
10        Allow from all
11    </Proxy>
12
```

```
13 <Location />
14     ProxyPass <analytics address>
15     ProxyPassReverse <analytics address>
16 </Location>
17 </VirtualHost>
```

Listing L.2: Apache configuration

Parameters needing changes:

- <hostname> is the hostname of the server. e.g. hostname.domain.com
- <analytics address> is the URL for the analytics server. This is displayed when npm start is run. by default this is 'http://localhost:5001/'
- The `ErrorLog` and `CustomLog` parameters can be changed to any location

L.5 Deployment of AngularJs Files

The authoring tool and [Videogular](#) Questions example repositories are both written in AngularJS.

AngularJS files are JavaScript and therefore can be run on a normal Apache webserver.

To install these repositories you must clone the repository to a location served by Apache and then install the dependencies by running `npm install`

Once this has been performed the application will be able to be used.

Accessibility Standards

Appendix

M

An appendix detailing the levels of accessibility standards compliance in the tools produced.

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”

- Tim Berners-Lee, Creator of World Wide Web

M.1 Introduction

Both of the example user interfaces were tested against the [Web Content Accessibility Guidelines \(WCAG\) 2.0 Standards](#)¹ to check for accessibility.

M.2 Videogular Questions Example

Table M.1: Conformance to WCAG 2.0 Guidelines for Videogular Questions Example

Standard	Pass	Comment
1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.	✗	There is no text alternative for the video but as the idea is to integrate this into Synote giving this feature would cause duplication
1.2 Time-based Media: Provide alternatives for time-based media.	✗	There are no captions for the video content. This is down to the video uploaded by the user.
1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)	✓	All buttons that use symbols have aria-labels
1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)	✓	Sensible read order determined by WAVE toolbar
1.3.3 Sensory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size, visual location, orientation, or sound. (Level A)	✗	Video content relies on sound. As this is for integration into Synote the transcript will be provided there.

Continues on the next page...

¹<http://www.w3.org/TR/WCAG20/> (Accessed: 15 Jan 15)

Standard	Pass	Comment
1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)	✗	The cuepoints are only conveyed by a different colour on the scrub bar. Future work could be to add information conveyed in other ways.
1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)	✓	Video does not play until play button is clicked. The volume can also be adjusted separately of system volume
1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA) <ul style="list-style-type: none"> • Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1; • Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement. • Logotypes: Text that is part of a logo or brand name has no minimum contrast requirement. 	✓	All text has an acceptable contrast ratio
1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)	✓	All text can be resized to 200% without loss of content or functionality

Continues on the next page...

Standard	Pass	Comment
<p>1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA)</p> <ul style="list-style-type: none"> • Customizable: The image of text can be visually customized to the user's requirements; • Essential: A particular presentation of text is essential to the information being conveyed. <p>Note: Logotypes (text that is part of a logo or brand name) are considered essential.</p>	✓	There are no images of text
<p>2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)</p>	✓	All content is operable using only a keyboard
<p>2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)</p>	✓	There are no keyboard traps
<p>2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)</p>	✓	No specific timings are needed for the keyboard strokes (you can pause then move along the scrub bar rather than having to pause at the correct point).
<p>2.2 Enough Time: Provide users enough time to read and use content.</p>	✓	Level AAA as no user interactions are time sensitive
<p>2.3 Seizures: Do not design content in a way that is known to cause seizures.</p>	✓	None of the content flashes.

Continues on the next page...

Standard	Pass	Comment
2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)	✓	There is a skip to content link for screen readers
2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A)	✓	All pages are titled descriptively
2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)	✓	The tab order matches the read order
2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)	✓	Links are obviously to other videos
2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)	✓	Every page can be found from all pages
2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)	✓	Headings and labels are descriptive
2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)	✓	Keyboard focus is visible (dotted lines)
2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA)	✓	The title shows which of the pages the user is on and there is only one level of navigation
2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA)	✓	Links are all to other videos

Continues on the next page...

Standard	Pass	Comment
2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA) <ul style="list-style-type: none"> Note 1: "Heading" is used in its general sense and includes titles and other ways to add a heading to different types of content. Note 2: This success criterion covers sections within writing, not user interface components. User Interface components are covered under Success Criterion 4.1.2. 	-	No section headings are needed as content is all in one section
3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)		
3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)	✓	The lang attribute is set to en
3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)	✓	No settings to change
3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)	✓	Same navigation on each page
3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)	✓	All symbols and references are consistent
3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)	✓	User needs to click on a link to change the context

Continues on the next page...

Standard	Pass	Comment
3.3 Input Assistance: Help users avoid and correct mistakes.	✓	Validation on answers but no help in actually answering the question. Can skip back to certain content if that setting is set
4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)	✓	HTML validates apart from some of the angular attributes which will not HTML validate
4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)	✓	Aria roles and marks have been defined

M.3 Videogular Analytics Example

Table M.2: Conformance to WCAG 2.0 Guidelines for Videogular Analytics Example

Standard	Pass	Comment
1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.	✗	There is no text alternative for the video.
1.2 Time-based Media: Provide alternatives for time-based media.	✗	There are no captions for the video content. This is down to the video uploaded by the user.

Continues on the next page...

Standard	Pass	Comment
1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)	✗	The graphs are not explained in text. However this was a proof-of-concept to show that graphs could be used. The text could be added by a developer
1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)	✓	Correct reading sequence is determined by the WAVE toolbar
1.3.3 Sensory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size, visual location, orientation, or sound. (Level A)	✗	To understand the graphs you need to be able to see them. Future work could be on the development of accessible graphs
1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)	✓	The heat map data is conveyed by a different colour on the scrub bar but also in a table
1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)	✓	Video is not playing by default and the volume can be adjusted independently of overall system volume

Continues on the next page...

Standard	Pass	Comment
<p>1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)</p> <ul style="list-style-type: none"> • Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1; • Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement. • Logotypes: Text that is part of a logo or brand name has no minimum contrast requirement. 	✓	All colour contrast is of an acceptable ratio
<p>1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)</p>	✓	The page is still functional when zoomed
<p>1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA)</p> <ul style="list-style-type: none"> • Customizable: The image of text can be visually customized to the user's requirements; • Essential: A particular presentation of text is essential to the information being conveyed. <p>Note: Logotypes (text that is part of a logo or brand name) are considered essential.</p>	✓	All text can be resized without assistive technology up to 200% without loss of content or functionality

Continues on the next page...

Standard	Pass	Comment
2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)	✓	All content is completely functional for a keyboard-only user
2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)	✓	No keyboard traps exist
2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)	✓	No specific timings are required
2.2 Enough Time: Provide users enough time to read and use content.	✓	Level AAA as no user interactions are time sensitive
2.3 Seizures: Do not design content in a way that is known to cause seizures.	✓	No flashing content
2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)	✓	Skip to content link is available for screen readers
2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A)	✓	All tabs have descriptive titles
2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)	✓	Tab order matches read order

Continues on the next page...

Standard	Pass	Comment
2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)	✓	Links are only to switch tabs which are named
2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)	✓	All information is in one page
2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)	✓	All headings and labels are descriptive
2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)	✓	The keyboard focus is visible
2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA)	✓	All information is on one web page
2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA)	✓	Links are only for switching tabs and roles are defined for the tabs
2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA) <ul style="list-style-type: none"> • Note 1: "Heading" is used in its general sense and includes titles and other ways to add a heading to different types of content. • Note 2: This success criterion covers sections within writing, not user interface components. User Interface components are covered under Success Criterion 4.1.2. 	-	No sections headings as sectioning is not required
3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)	✓	The lang attribute is set to en

Continues on the next page...

Standard	Pass	Comment
3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)	✓	A change of focus does not change the context
3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)	✓	User is advised of behaviour in the text by the component
3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)	✓	This is a one page site
3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)	✓	All names and references used are consistent
3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)	✓	A change of context can only come from a user changing the tab
3.3 Input Assistance: Help users avoid and correct mistakes.	-	There is no wrong input
4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)	✓	HTML validates apart from some of the angular attributes which will not HTML validate
4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)	✓	Aria roles and markers are set

M.4 Authoring Tool

Table M.3: Conformance to WCAG 2.0 Guidelines for the Authoring Tool

Standard	Pass	Comment
1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.	✗	There is no text alternative for the video. As the video is uploaded by the user this should not be an issue.
1.2 Time-based Media: Provide alternatives for time-based media.	✗	There are no captions for the video content. This is down to the video uploaded by the user.
1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)	✓	Sensible linear read order identified by the WAVE toolbar ²
1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)	✓	Sensible linear read order identified by the WAVE toolbar
1.3.3 Sensory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size, visual location, orientation, or sound. (Level A)	✓	All instructions are in a textual format
1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)	✓	Colour has no meaning in the page

Continues on the next page...

²<https://wave.webaim.org/toolbar/> (Accessed: 15 Jan 15)

Standard	Pass	Comment
1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)	✓	Video content is paused by default and has a play/pause button
1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA) <ul style="list-style-type: none"> • Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1; • Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement. • Logotypes: Text that is part of a logo or brand name has no minimum contrast requirement. 	✗	All of the editable colours are of an acceptable contrast ratio but the select highlight colour is not alterable. This element would need to be switched for a completely different element
1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)	✓	Text reflows well. Some buttons are outside their boundaries
1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA) <ul style="list-style-type: none"> • Customizable: The image of text can be visually customized to the user's requirements; • Essential: A particular presentation of text is essential to the information being conveyed. <p>Note: Logotypes (text that is part of a logo or brand name) are considered essential.</p>	✓	All text is customisable

Continues on the next page...

Standard	Pass	Comment
2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)	✗	All components are accessible from the keyboard in Firefox. However on Chrome there is no keyboard accessibility for using the multiple select box
2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)	✓	There are no tab loops
2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)	✓	None of the keyboard combinations require specific timing
2.2 Enough Time: Provide users enough time to read and use content.	✓	Level AAA as no user interactions are time sensitive
2.3 Seizures: Do not design content in a way that is known to cause seizures.	✓	There is no flickering content
2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)	✓	There is only one webpage
2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A)	✓	The page is titled
2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)	✓	The tab order is the same as the programatically determined read order

Continues on the next page...

Standard	Pass	Comment
2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)	✓	There are no links on the page
2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)	✓	There site is all on one page
2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)	✓	The page has a logical heading and the sections are also headed
2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)	✓	Focus is made obvious e.g. buttons have a thick dotted line around them when in focus
2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA)	✓	This is a one page application
2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA)	✓	No links in the page
2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA) <ul style="list-style-type: none"> • Note 1: "Heading" is used in its general sense and includes titles and other ways to add a heading to different types of content. • Note 2: This success criterion covers sections within writing, not user interface components. User Interface components are covered under Success Criterion 4.1.2. 	✓	Each section (accordion) panel has a heading

Continues on the next page...

Standard	Pass	Comment
3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)	✓	The HTML <code>lang</code> attribute is set to <code>en</code>
3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)	✓	No change of focus when components receive focus
3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)	✓	Changing settings does not change the context
3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)	✓	This is a one page application
3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)	✓	This is a one page application
3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)	✓	No changes of context
3.3.1 Error Identification: If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text. (Level A)	✗	No error messages
3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A)	✓	Instructions or labels are provided
3.3.3 Error Suggestion: If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content. (Level AA)	✗	No suggestions for input errors detected

Continues on the next page...

Standard	Pass	Comment
<p>3.3.4 Error Prevention (Legal, Financial, Data): For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true: (Level AA)</p> <ul style="list-style-type: none"> • Reversible: Submissions are reversible. • Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them. • Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission. 	-	N/A
<p>4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)</p>	✓	HTML validates correctly
<p>4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)</p>	✓	ARIA roles and states are set where appropriate

Firefox Bug Report

Appendix

N

An appendix detailing a bug report submitted for the Firefox browser.

N.1 Introduction

This is the contents of the bug filed in the Mozilla bug tracker (number 1106045) [8], reporting the occurrence of “focus loops” in certain HTML structures. It was later discovered that these structures were invalid HTML, and so the bug will likely go unfixed.

N.2 What did you do?

I was testing a Web application for keyboard accessibility. In the application there were some HTML structures like the following:

```
1 <a href="#">
2   <div>
3     The link text
4     <button type="button">Button 1</button>
5     <button type="button">Button 2</button>
6   </div>
7 </a>
```

(I have created a JSFiddle illustrating the problem at <http://jsfiddle.net/jjxuL1tz/5/>.)

I attempted to tab through this structure to focus elements further on in the HTML.

N.3 What happened?

I became stuck in a focus cycle. That is, pressing Tab when Button 2 was focussed returned focus to the parent <a> tag, so the focus went as follows:

```
<a> -> Button 1 -> Button 2 -> <a> -> Button 1 -> ...
```

When <a> was focussed, I could press Shift+Tab to get out of the cycle and focus the previous element in the document, but I could not focus the element after the cycle without using the mouse.

N.4 What should have happened?

With Button 2 focussed, I should have been able to press Tab to focus the next element on the page.

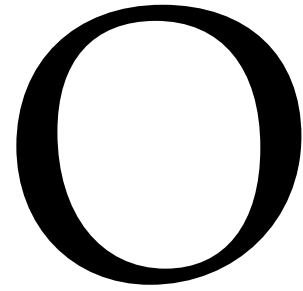
N.5 Workaround

The problem does not occur if the `<button>`s are replaced with `<a>`s; indeed, adding one `<a>` tag as a sibling of the `<button>`s fixes the problem, even if it has its `tabindex` attribute set to -1.

An example of this can be found in the JSFiddle (<http://jsfiddle.net/jjxuL1tz/5/>).

Deliverables Report

Appendix



An appendix detailing the deliverable report approved by the client on 2014-12-08.

School of Electronics And Computer Science

ELEC6200 MEng Group Design Project, Group 12

Christopher Baines
 Samuel Bennett
 Harry Cutts
 Christopher Hewett
 Maria Lynch

Contents

1	Customer Deliverables	
2	Videogular Cuepoints	
3	Videogular Heatmap	
4	Videogular Analytics	
4.1	Feature Walkthrough	
4.2	Analytics emitted events	
5	Authoring tool	
5.1	Features	
5.2	Full Feature Walkthrough	
6	Videogular Questions	
6.1	Features	
6.2	Walkthrough	

1 Customer Deliverables

It has been agreed during the weekly customer meeting on 1st December 2014 that the following deliverables shall be produced as part of the final project:

- Videogular Questions: a plugin for adding polls and questions to videos
 - Source code, under the MIT licence
 - An example proof-of-concept website
- Videogular Cuepoints: a plugin which displays informational marks on the scrub bar of a video
 - Source code, under the MIT licence
 - Demonstration of usage in Videogular Questions example site

- Videogular Heatmap: a plugin which displays heat map information on the scrub bar of a video
 - Source code, under the MIT licence
- Videogular Analytics: a plugin for reporting of events within the Videogular player
 - Source code, under the MIT licence
 - An API specification for Videogular Analytics
 - An example site showing basic usage of the analytics data collected from the plugin
- Authoring tool: a Web application to produce the JavaScript file to be used with Videogular Questions
 - Source code, under the MIT licence

Links to <http://kanga-cb15g11.ecs.soton.ac.uk> will require access to the ECS network. Accessing this website from a lab machine or directly networked computer will work, otherwise the ECS VPN will be required.

2 Videogular Cuepoints

The plugin source code is available to download from our repository located at

<https://github.com/soton-ecs-2014-gdp-12/videogular-cuepoints>

Usage instructions can be found in the README file in that repository, and an example can be found in the Videogular Questions example.

3 Videogular Heatmap

The plugin source code is available to download from our repository located at

<https://github.com/soton-ecs-2014-gdp-12/videogular-heatmap>

Usage instructions can be found in the README file in that repository.

4 Videogular Analytics

The plugin source code is available to download from our repository located at

<https://github.com/soton-ecs-2014-gdp-12/videogular-analytics>

The example analytics back end has been set up as a demo at

<http://kanga-cb15g11.ecs.soton.ac.uk/gdp/analytics>

4.1 Feature Walkthrough

The example that we have written for the Videogular Analytics plugin has a storage and replay facility. You can store a list of events received with a JSON file and replay them later. This allows you to post-process events as needed.

To demonstrate the features of the analytics back end, we have loaded some sample data into the example website.

However, since this is only an example of how you can use the analytics, not a deliverable product, it is not fully featured. Further processing methods could be implemented in the future, but all current features work.

The following demonstration features have been implemented:

- Events Log page
 - The top of the page lists all events received, including the event type, the time received, and the data held by the events.
 - The bottom of the page shows the sections of the video that each user has watched. The Unique User Identifier (UUID) is used to show different users, but does not identify a user in any way.
- Per user statistics
 - The watched video segments are processed to show how much of the video each user has viewed.
 - The data is also processed to show which how many times each segment has been watched. There is a known bug with Videogular reporting odd times, causing this statistic to display incorrectly. We have been investigating this, but the data demonstrates a processing possibility. This is the numerical form of the data displayed using the heatmap.
- Results page
 - This shows the results of each question, and the correct answer where possible.
- Results Correlation
 - The “% watched by correct answers” and “Time watched by correct answers” graphs show two ways you could represent the data in a different way.
 - Both of these examples use random data but the majority of the code for producing the graphs has been written. We have abstracted the scatter graph creation code so that arbitrary data sets can be given to the scatter plots.

4.2 Analytics emitted events

The Videogular Analytics plugin emits a number of events which can be sent to a Web server for collection. The table below describes these events.

The plugin is designed so that events can easily be added by sending events to the ‘analytics’ channel. These are automatically collected and prepared for sending to the given Web server.

Event name	Emitted when	Expected Payload
show_question	a question is shown to the user	All of the associated question data
end_question	the annotation being shown has finished	None
show_results	there are results that will be shown	The results data being shown
submitted_question	the user submits a question	The chosen question response
skipped_question	the user skips a question	None
continue_question	a results page is closed by pressing the continue button	None
play	the video starts to play	The time the video plays from
pause	the video is paused	The time the video pauses at
stop	the video is stopped	The time the video was stopped at

5 Authoring tool

The plugin source code is available to download from our repository located at

<https://github.com/soton-ecs-2014-gdp-12/authoring-tool>

The authoring tool has been set up as a demo at

<http://kanga-cb15g11.ecs.soton.ac.uk/gdp/authoring-tool>

5.1 Features

The authoring tool is being delivered as a working prototype and allows you to create a JavaScript file for the Videogular Questions plugin.

Videogular Questions provides a preview of the questions being created. There will be minor display changes when going from the authoring tool to a working example as the user may style the questions freely.

Pressing **Export** will provide you with a JavaScript file which can be used. Pressing **Preview** will reset the video back to the start, load the questions into the preview, and begin playing.

All 5 question types are implemented and, as suggested, the Single choice question (with radio buttons) is merged with the Multiple choice question. If you create a Multiple choice question with one correct answer and allow the user to select a minimum and maximum of 1 answer a Single choice question will be generated.

For demonstration purposes, the authoring tool has been set up with the Caesar Cipher example. Once Videogular's YouTube plugin has been fixed for the current version this will be easier to modify, however this was agreed not to be a priority for our project.

It is possible to generate a number of JavaScript files which produce invalid or odd questions, such as a rating question with a minimum of 0 and a maximum of 0. To more easily allow further improvements to the authoring tool, it has been decided that we shall not limit what the user can create.

In the future, the tool could be made 'safe' so that no user can create an invalid question. However, leaving the user to create a quiz precisely how they wish, not limiting the large number of options, allows them much more power.

5.2 Full Feature Walkthrough

We have designed a number of tests for the authoring tool. Below is the process we have been going through to ensure that the functionality works as expected. After each step the preview button is pressed to confirm that the JavaScript file is correct, and the JavaScript file is validated by checking it manually.

- A question set is added at a specific time.
- A single question is added, then tested to ensure it appears.
- Each question type is tested along with common options.
- Skipping is tested for one question.
- Recording responses and viewing them is tested for one question.
- The interface is reloaded, then a number of question sets and random questions are created and tested, ensuring that they appear at the correct times.

6 Videogular Questions

The plugin source code is available to download from our repository located at <https://github.com/soton-ecs-2014-gdp-12/videogular-questions>

An example site can be found at

<http://kanga-cb15g11.ecs.soton.ac.uk/gdp/examples/#/simple-example>

6.1 Features

- Single Question provides radio buttons to select at most one answer.
- Question Multiple provides checkboxes to select one or more results. The min/max values should be shown and should be disabled if these requirements are not met.

- Questions Star should allow you to click on the stars and select one rating. The Min/Max values should be shown and should be disabled if these requirements are not met.
- Questions Text should allow you to enter a textual value before submitting.
- Question Range should present a slider which allows you to choose a range of values. The min/max values should be shown on the slider.

6.2 Walkthrough

Each tab should be tested to see the example works.

The tabs Single Question, Question Multiple, Question Stars, Question Text, Question Range all should show one demonstration of the specific question type.

The simple example should show a very basic example, this will be similar to Single Question.

The poll simple example should show an example poll.

Caesar Example is used to demonstrate a number of features being used. This has Videogular Cuepoints enabled so you will be able to see where the questions will be shown. The second question at the end of the example will have a poll so you will be able to see the results of all those who have answered the question.

Appendix

P

Project Brief

An appendix detailing the project brief submitted 2014-10-09.

School of Electronics And Computer Science

ELEC6200 MEng Group Design Project

Project Specification And Plan

Title: Interactive Web video quizzes and polls

Supervisor: Mike Wald

Team Members:

Christopher Baines

Samuel Bennett

Harry Cutts

Christopher Hewett

Maria Lynch

Customer: Mike Wald

Project Specification: A library will be implemented which allows quizzes and polls to be overlaid on Web videos, playable in the main browsers (Firefox, IE and Chrome) on different platforms (PC, Mac and Android). These videos could be any of the main video formats (MP4, WebM, OGG). Questions will be specified in JSON files interpreted by the library. Work will be done to create an authoring tool for the system. Accessibility considerations for the overlaid videos and authoring tool will be investigated. Metrics of user behaviour will be recorded and methods of displaying these to the author explored.

Work plan: Gantt charts can be found in figures 1 and 2.

SCRUM, an Agile methodology, will be followed. This means that the work done in each Sprint will be subject to change until the end of the Sprint Planning Meeting on the Monday of each week.

The aim for the end of Sprint 3 will be to have the quiz and poll overlay complete. During the following six Sprints the authoring and analytics tools will be created in parallel.

Please note that each sprint contains some time in which report writing will be completed.

Initial Gantt charts are not shown, can be found in [section F.1](#).