Electronics and Computer Science

Faculty of Physical and Applied Sciences

University of Southampton

Matthew Consterdine

2017

Lect.Me: A Real-Time, Interactive Presentation & E-Learning Platform

Project Supervisor:	Rikki Prince
Second Examiner:	Martin Charlton

A project report submitted for the award of

MEng Computer Science

Abstract

Setting out to improve audience participation in lectures, this project aimed to help engage audiences by improving presentation flow, to maximise learning potential. Researched learning theory and different technologies to make the most of this opportunity. Decided to create a modern, responsive web-app designed for instantly sharing content, with interactive elements, to the entire audience. It was decided that the web-app needed to be able to share everything required for a lecture: text, pictures, video, pdfs, documents, quizzes, and most importantly for a web-app, websites using `<iframe>`s. This is while gathering statistics to help the presenter gauge participation and engagement. After the presentation, participants can revise from the material.

To achieve such synchronisation, Meteor (A JavaScript Node based framework) was used as it was designed for such tasks. Under the hood, WebSockets are used for minimal latency. After prototyping, development and testing, the web-app was tested and deployed for demonstration purposes using the domain name Lect.Me, a VPS, Docker, and Nginx. Feedback was evaluated, and the web-app was improved.

In this report, I discuss the steps taken during the various stages of this project, and offer my conclusions on the results.

Keywords

Interactive Learning, Education Systems, Sharing, Real-time Data, Software Development, HTML5, Computer Networks, Responsive Design, User Focus, Accessibility.

Contents

Acknowledgements

I would like to thank Rikki Prince for his help and guidance throughout this project. Without him, I would have been difficult to complete.

I would like to thank Adriana Wilde for meeting not once but twice following a troubling demonstration, Oliver Bills for eager words, Federica Paci for great feedback, Adam Warren for an insight into similar systems, and Ed Zaluska for helping me organise meetings. Additionally, thanks to ECS for lending me technology when I needed it most.

Lastly, thanks to my family and friends for making this an amazing year.

1. Introduction

1.1. The Problem

Lectures, especially those spanning multiple hours, often run the risk of disengaging their audience without adequate breaks in the presentation flow, damaging the learning potential. To combat this, lecturers may either give the audience a few minutes to stand up, or better yet, break up the presentation with useful examples or quizzes to both review and motivate [1]. However, this switch between slides and supplementary content frequently takes significant time away from the lecture, is unreliable, and often fails to live up to its true potential.

My project aims to fix this disconnect, and bring more interactive lectures to the halls.

1.2. The Solution

I aim to create a fully responsive, online web-app that will aid the learning experience both inside and outside of the lecture theatre. My web-app will combine lecture slides, quizzes, videos, links, and more in one place, helping the lecturer and students focus on the material, and provide added learning.

In the lecture theatre, lecturers will be able to use a standard clicker to advance within and between any resource, both on the projector, and on viewer's devices. I chose a web-app both so it can run anywhere, and to enable students to follow along on their devices. Following the lecture, the lecturer can share the presentation, allowing the learner to revise at home. Statistics will be gathered.

The learner will be able to join a room using either a link or QR Code (Quick Response Code), or by subscribing to the lecturer.

1.3. Project Scope

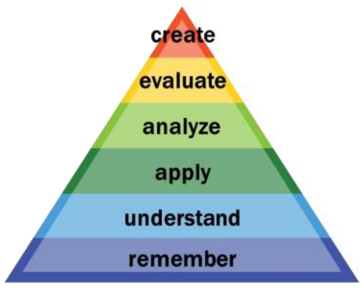
I have chosen to create a responsive web-app because that will allow it to run on any modern smartphone, tablet, laptop, or desktop without needing to create native apps for each device and OS.

I plan to support the ability to display slides, external resources, and quizzes both on a projector and on learner's devices at minimum; I aim to create different handlers for different external resource types. To display documents and presentations, I plan to use viewerjs.org as it supports the most common document formats. To synchronize devices, I shall use WebSockets as they are the lightweight real-time communication protocol supported by all modern web browsers. To create a responsive, fun design I am planning to make use of materializecss.com which takes inspiration from Google's material design standard [2].

My project has a wide scope, so even if I encounter setbacks I should be able to adjust and deliver a useful, polished web-app. If I get time, I will add additional features such as quizzes, different content, and anything else that while reading, I discover to be of use to my final design.

2. Background Research

2.1. Education Theory



Bloom's Taxonomy [27]

While researching education theory, I discovered Bloom's Revised Taxonomy [3] and how it can be used to understand and enhance learning. Anderson and Krathwohl build upon Bloom's original work in their 2001 book, splitting the cognitive process into six parts. Using the Taxonomy allows for the organization of learning objectives, and can provide clarity and optimally spread knowledge. I aspire to create an app that helps both creators and participants at each level of the pyramid. It is ineffective to focus on a single layer.

Mark Prensky writes about the divide between "Digital Natives, Digital Immigrants" in his 2001 article of the same name. In it, he describes the decline in the American education system specifically caused by a technological disconnect between the teachers and the students, and the need for media rich and interactive content to bridge the generational gap. My app aims to help solve this, giving presenters a new means to engage with their audience.

In his 1997 paper Eric Mazur states, "we can no longer afford to ignore the inefficiency of the traditional lecture method" [4] proposing instead a method focused self-study, and the student's ability to build understanding by applying it in his "ConcepTests [sic]". Thus, he improves both energy and enthusiasm; a higher and more active participation leads to greater understanding. I want to bring forward this focus on rich interactive media, and learning reinforcement into my final app design.

I went on to research E-Learning (Electronic Learning) and found R. Clark, and R. Mayer's 2016 book: E-learning and the science of instruction [5]. Inside they discuss how E-Learning can be used to enhance both asynchronous and synchronous learning. I want to enhance both. Unlike other books, their findings are based on a strong foundation of research and study. Finally, I am going to discuss M-Learning (Mobile Learning), which is a "new stage of progress" that aims to take the freedom granted by E-Learning out of the classroom or house [6]. While my plan is to develop a responsive web-app that will work anywhere, I am not planning to specifically tailor it towards M-Learning.

As I continued working on my third-year project, I continued my research to help answer important design and planning questions, ensuring that I made informed decisions.

2.2. Existing Solutions

During my research, I looked at Nearpod [7] and Tophat [8], two existing solutions in the same space. In both you can present learning material to an audience, which is one of the features I am looking to implement in this project.

There is certainly overlap, but I want to produce a web-app with a greater focus on user feedback, and data. To achieve this, the developed system will need to be quick and easy to use, trivial to join, and provide added value to participants. There is little point in developing a web-app that requires participation, but does not reward the user for doing so.

2.3. Technology

2.3.1. Deciding on a Web-App

Initially, I was unsure whether I wanted to make a native app or focus my efforts on a single web-app. I was tending towards web-app as that would entail creating a single app that runs nearly everywhere, however I decided I needed to do further demographic research.

The University of Glasgow has released a biennial Technology Survey of students since 2007 [9], inside it breaks down the technology ownership, and more importantly carrying habits of their students. From this data, it can be observed that 91.2% of students carry a smartphone at all times, with 87.1% either happy or neutral to use them in class to learn. On the other hand, only 20.4% of students carry a laptop at all times (69.3% carry if they know they will need one), although 95.1% are either happy or neutral to use them in class to learn.

I settled on creating a web-app, not only are students using a wide range of devices, they are also using a wide range of operating systems. This would require creating at least 4 different native apps (Android, iOS, Windows, & OSX all have significant market share [9]) and even then, it would exclude those running smaller operating systems including Linux (2.0% Glasgow market-share [9]), and Windows Phone (9.1% UK market-share [10]).

2.3.2. Deciding a Technology Stack

Knowing I was going to create a web-app, next I had to determine which software stack I would be using behind the scenes. While on the front-end I knew, I would need to use a mixture of HTML5, and CSS for the markup, with either JavaScript or a language that compiles to JavaScript for any scripting, the back-end was a mystery.

Looking back at what I already knew, I immediately discarded using raw sockets due to the sheer complexity of working that close to the hardware. I do not need, or have time, to implement HTTP and TLS myself. Java RMI (Remote Method Invocation) was also quickly culled due to its high coupling, poor design decisions, and general outdatedness of the platform itself; there are much better options [11].

From here I looked towards PHP, a language I had some previous experience with, and one designed for the web. Last semester, I was taking the Cyber Security module that had two courseworks focused on PHP web-app security. They were also part of the reason I eventually decided against using PHP. PHP is "a fractal of bad design" [12] full of unpredictable, inconsistent, verbose, flaky, and opaque features. Choosing PHP on a new code base is not a sensible idea, there is a reason Facebook decided to create their own language and runtime [13].

Several weeks into my project, my supervisor Rikki Prince casually suggested using Meteor, a real-time JavaScript framework focusing on 'write once, run everywhere', drawing power from the large Node.JS eco-system [14]. With Meteor, I can create a single responsive web-app that works on virtually any device, I can quickly make changes using unit testing to ensure nothing breaks, and if desired I can effortlessly build it into a native app. I already had significant font-end JavaScript experience, so I felt it was time to try back-end JavaScript.

3. Planning

To allocate time, I produced a Gantt chart (Made using Microsoft Project) including not only the important project deadlines, but also exam and coursework deadlines for each module I am taking every semester. I will be able to actively and accurately manage my time to ensure that I can produce this web-app to meet the deadline set and to the best of my abilities.

3.1. Risk Assessment

Risk	P.	S.	E.	Mitigations
Inexperience	5	3	15	While I have used JavaScript to develop small web-apps, I have never built something of this scale or used Meteor. To overcome this, I experimented with it, completing tutorials and going so far as to make a small prototype.
Scope too large	3	3	9	If the scope proves too large to handle, I will plan around this to reduce the unnecessary functionality to create a useful, minimal viable product. Using Agile methodology helps, as it helps ensure an always working web-app.
Performance Issues	2	2	4	MongoDB is designed for big data so I do not foresee any performance issues. Meteor on the other hand, might need profiling and testing to ensure optimum performance. I am not creating a native app.
Illness / Absence	1	5	5	If I were to fall ill to the point where I could not work on the project, I will be sure to alert my project supervisor so he is aware. Then I will scale back my work to the MUSTs, and if truly necessary, ask for accommodations.
Data Loss	1	5	5	I will be using GIT for version control and pushing to GitHub ensuring I have backups in different locations.
Poor Testing & Evaluation	2	3	6	For me to make any formal conclusion or evaluation I will need to test and evaluate my project. Investigating formal user testing will be a matter of priority.
Technology Change	0.5	5	2.5	I cannot foresee any change that completely obsoletes Meteor or MongoDB within the next few months. However, if lightning strikes, I shall consider migrating to a different yet similar JavaScript framework.

For brevity, **P**robability, **S**everity, and **E**xtent have been replaced with single letter acronyms.

3.2. Agile Methodology

When developing my web-app, I will be using Agile methodology to evolve my plans and designs as I develop the web-app [15]. I will use this evolutionary development and continuous improvement to focus on the few essential features at the heart of the project, then work outwards, adapting my plans when need be. This ensures that I always have a working web-app, and do not need to spend weeks working on it before I get any tangible results; I will not be left in the position where I have spent weeks working on a dead end.

3.3. Understanding Requirements

To understand, I first analysed the stakeholders, then determined their requirements.

3.3.1. Stakeholder Analysis

Stakeholder	Role	Description
Creator	Primary	Creates a presentation using the web-app by uploading material, ordering it and authoring quizzes.
Presenter	Primary	Presents the material, engaging with the audience
Participant	Primary	An active participant watches/listens/interacts with the presenter, actively using the web-app to make the most of it.
Observer	Secondary	An inactive participant only watches/listens with the presenter, not directly using the app or making the most of it.
Developer	Tertiary	Plans, designs and develops the web-app.

3.3.2. Requirements List

Requirements were determined following background reading, by studying existing solutions, and determining what would be needed for this project to be a success. I discussed the project with classmates, and my supervisor.

To rank requirements, I am using the MoSCoW method. It involves sorting requirements into Must have, Should have, Could have, and Won't have, this is where the acronym is from [16].

Sprint 1 Requirements	MoSCoW
Fully comply with the Data Protection Act of 2003 **[17]**	MUST
Ensure integrity of data before it reaches the database for security.	MUST
Accessible to all major devices and operating systems as a web-app.	MUST
Capacity to support at the very least a hundred connected users.	SHOULD
Able to add new content types easily due to design architecture	SHOULD
Quality web-app that works well and feels good to use	MUST
A reliable and stable web-app	MUST
Fast and responsive, should not take more than a second to react	MUST
Safe and secure with access controls and hashed passwords	MUST

Sprint 2 Requirements	MoSCoW
Fully comply with the European Union Cookie Law **[18]**	MUST
Ensure integrity of data before it reaches the database for security.	MUST
Provide admin tools to manage data and accounts without manual work.	COULD
Provide useful statistics for the creator/presenter to gather insight.	SHOULD
Add sufficient logging to help identify bugs and protect against misuse	SHOULD
A native app for each device and operating system.	WONT
Creators can create presentations accessible to deaf users.	SHOULD
Accessible to blind participants using screen readers (HTML ARIA **[19]**).	COULD
Help topics, possibly an onboarding tutorial to help people start using it.	COULD
Unit testing to prevent regressions	SHOULD
Collect and present real-time feedback	SHOULD
Allow participants to revise afterwards	SHOULD

3.4. Budget

As primarily a software project, I do not need to design and print circuit boards, buy a significant amount of hardware, or incur many costs. This is not to say that I will not definitely not need to use any of the budget.

- For hosting, I plan to use ECS's Virtual Machine Self Service [20].
- If I need a domain I shall either use GitHub's education pack to get a free .me domain [21], or spend roughly $10 (£8 although the price varies) [22] to purchase one.
- For TLS, I will use either the GitHub Education pack, Let's Encrypt [23] or Cloudflare [24].
- It is possible that I will need to purchase a presentation clicker, or wireless keyboard.
- If I decide to create an Android app, it will cost $25 (£20) to publish [25].

3.5. Ethics

While it was possible that during my evaluating, that I might have needed to complete an Ethics form for a study [26], I decided to interview a small number of lecturers instead, rendering that point moot.

4. Design

4.1. Use Cases

4.1.1. Editing

As	I want...	So that...	MoSCoW
Creator	Create and manage multiple presentations from a single account	Easily create a series of linked presentations and manage them all from a single account.	MUST
Creator	Upload material of many file types, inserting them	Use whatever material I already have without needing to unnecessarily convert everything.	MUST
Creator	Rearrange uploaded material to create a coherent presentation.	Aid understanding and remembering for anyone using the presentations [27].	MUST
Creator	Author simple quizzes, and questions for the participants	Let participants analyse and apply their knowledge, not simply recite [27].	SHOULD
Presenter	Use the site to present material	Enhance presentations and help participants.	MUST
Presenter	Host quizzes and get results	Engage with the audience	MUST
Presenter	Gather real-time feedback	Measure the lectures impact	SHOULD

4.1.2. Observing

As	I want...	So that...	MoSCoW
Participant	Join with a link	Quickly and easily join with a short unique code.	MUST
Participant	Join with a QR/NFC code	Quickly and easily join by scanning with your phone.	SHOULD
Any	Observe without an account	Only create an account if necessary. It will help adoption.	MUST

4.1.3 Accounts

As	I want...	So that...	MoSCoW
Creator	Create and manage multiple presentations from a single account	Easily create a series of linked presentations and manage them all from a single account.	MUST
Any	Observe without an account	Only create an account if necessary. It will help adoption.	MUST
Any	Create accounts with emails	Log in with the credentials that everyone has.	SHOULD
Any	Create accounts with social media	Quickly sign up, linking your account with social media.	COULD

4.1.4. Social Media

As	I want...	So that...	MoSCoW
Any	Create accounts with social media	Quickly sign up, linking your account with social media.	COULD
Creator	Post new presentations to social media	Automatically update social media followers of your progress.	COULD

4.2. Database Structure

I am going to be using MongoDB, a database engine that "[builds] on the best of relational with the innovations of NoSQL" [28]. The reason for doing so is that MongoDB is tightly integrated with Meteor, to the extent that each client has its own Minimongo client side cache of the database. Instead of tables, MongoDB makes use of collections of data, validated against a specific schema. This schema is what I am going to discuss.

For user accounts, I plan on using the Accounts meteor packages to quickly and easily integrate traditional email/password based accounts with modern social media ones [29]. This choice abstracts user accounts to an external package.

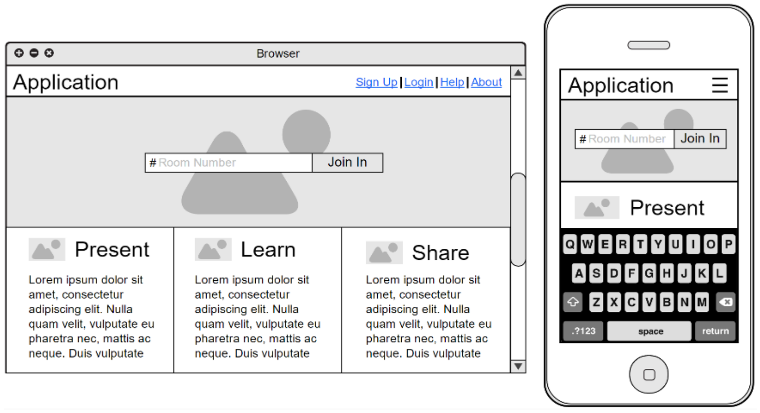
Presentations will be stored as MongoDB documents. The slides with be represented as a JSON array in the document, with the various pieces of metadata (author, title, time created stored as attributes). Access to this database will be managed and restricted by the server, to ensure data integrity and implement access controls.

Files will be uploaded and stored on disk, with an entry added to the presentation slides array if successful. To prevent collisions, I will take a hash of the file to use as the file name, possibly using SHA1. Metadata about the files will be primarily stored using the file system, however if this proves to be too limiting, I can add an additional MongoDB database and track it there instead. Access will be controlled by the server.

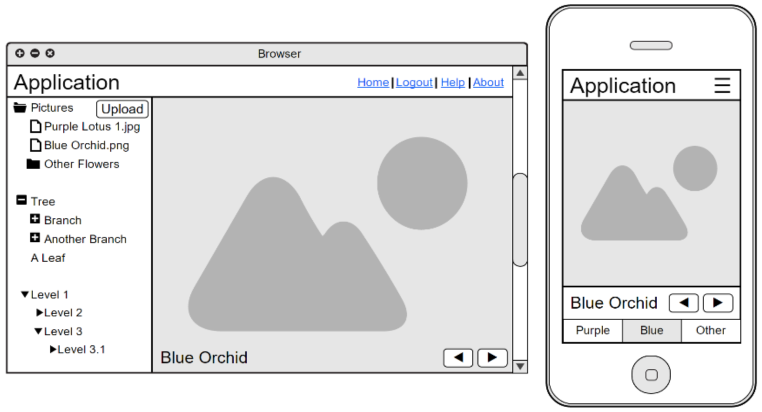
Interactive content will be represented as distinct MongoDB documents, and referred to in the Presentation database with a unique ID. Quizzes, for example will be stored as a title, and an array of questions; the results will be recorded.

While I do not anticipate any significant changes, the beauty of MongoDB allows a rather flexible database design while still validating to ensure data integrity. I can easily adapt as I continue to develop new and exciting features.

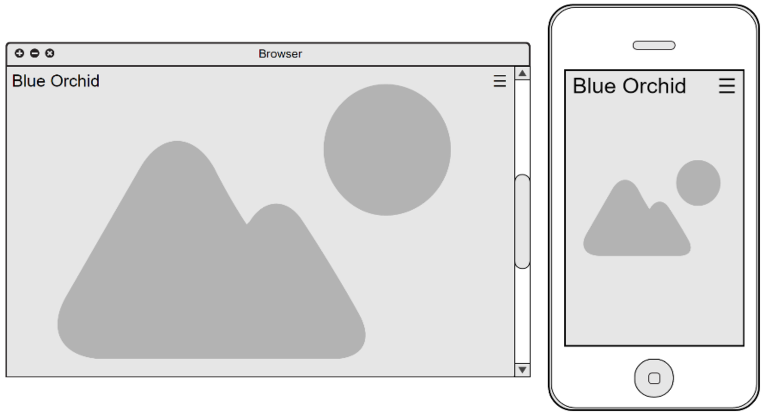
4.3. Mockups


Browser
Application
Sign Up | Login | Help | About
# Room Number
Join In
Present
Learn
Share
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate
Application
space
return

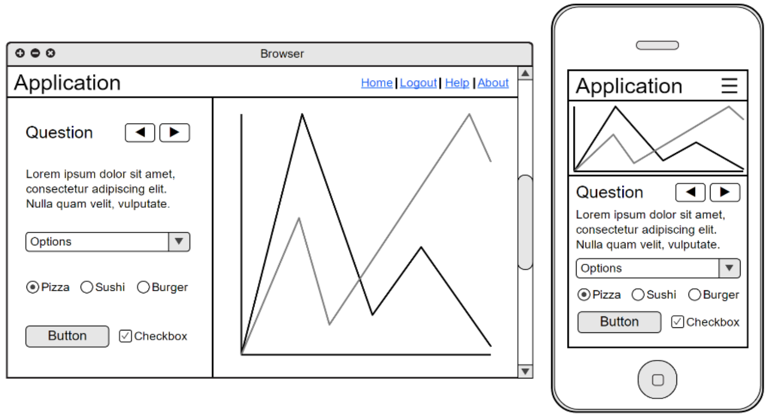
On the landing page, the layout adapts. To join a room, enter a number.



This page allows the creator and presenter to upload, edit and present.



I chose to keep it minimal, with a placeholder representing the slide.


Browser
Application
Home | Logout | Help | About
Question
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate.
Options
Pizza
Sushi
Burger
Button
Checkbox
Application
Question
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate.
Options
Pizza
Sushi
Burger
Button
Checkbox

Lastly, here is a quiz with real-time quiz progress. Layout is not final.

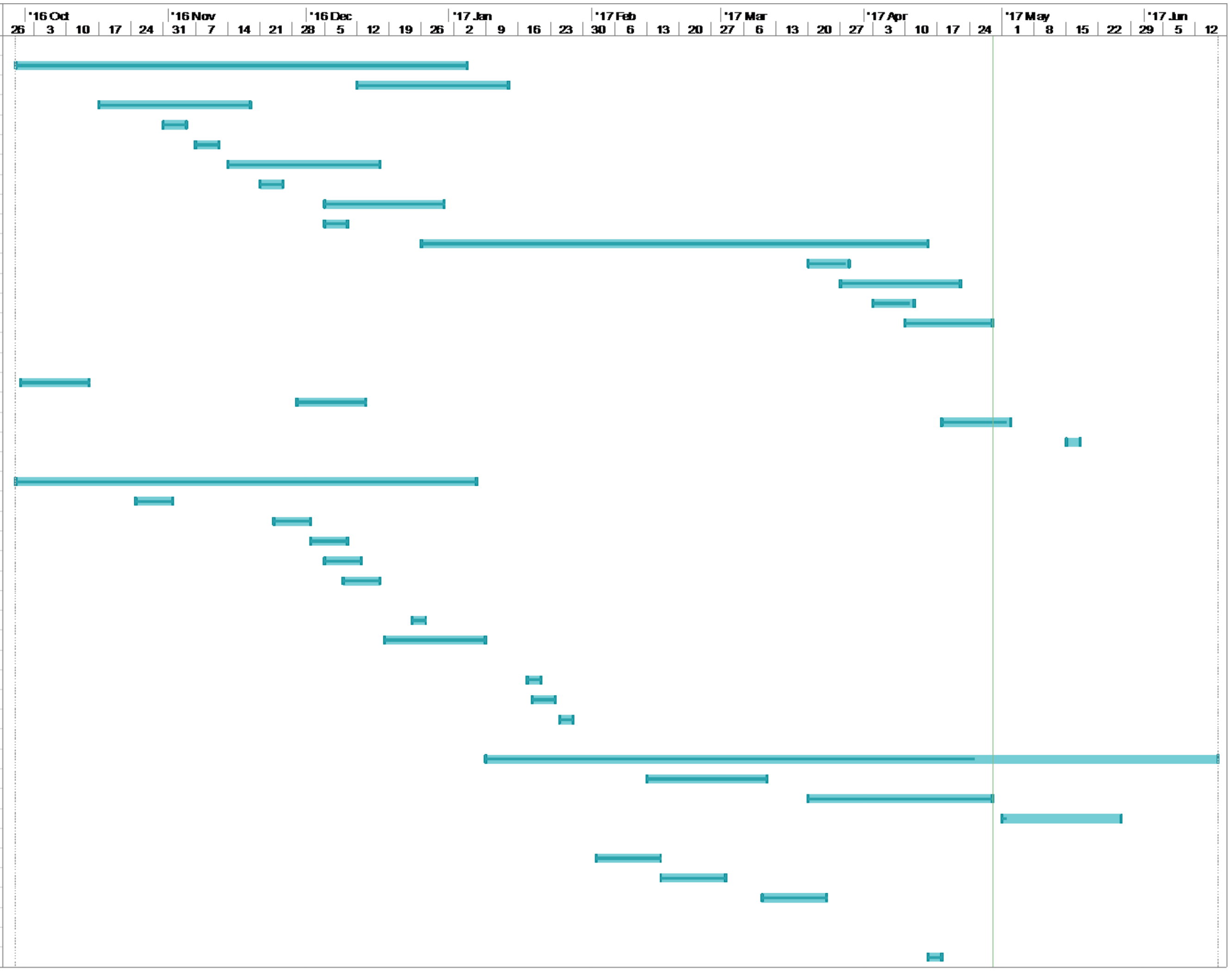
4.4 Prototyping

To prepare, I completed several of the official Meteor tutorials, experimenting as I went. When I fully begin, I will make a clean start accounting for the lessons learnt. For example, I will be removing the default packages `insecure` and `autopublish` [30] as they are purely for prototyping.

5. Project Management

5.1. Project Schedule

ID	Task Name	Duration	Start	Finish
1	~~Third Year Project~~			
2	Research	70 days	Thu 16-9-29	Wed 17-1-4
3	Technology	25 days	Mon 16-12-1	Fri 17-1-13
4	Planning	25 days	Mon 16-10-1	Fri 16-11-18
5	Risk Assessment	5 days	Mon 16-10-3	Fri 16-11-4
6	Requirements Analysis	5 days	Mon 16-11-7	Fri 16-11-11
7	Design	25 days	Mon 16-11-1	Fri 16-12-16
8	Use Cases	5 days	Mon 16-11-2	Fri 16-11-25
9	Prototyping	20 days	Mon 16-12-5	Fri 16-12-30
10	Mockups	5 days	Mon 16-12-5	Fri 16-12-9
11	Development	80 days	Mon 16-12-2	Fri 17-4-14
12	Deployment	7 days	Mon 17-3-20	Tue 17-3-28
13	Testing	20 days	Mon 17-3-27	Fri 17-4-21
14	User Feedback	7 days	Mon 17-4-3	Tue 17-4-11
15	Evaluation	15 days	Mon 17-4-10	Fri 17-4-28
16				
17	~~Deliverables~~			
18	Project Brief	11 days	Fri 16-9-30	Fri 16-10-14
19	Progress Report	11 days	Tue 16-11-29	Tue 16-12-13
20	Final Report	11 days	Tue 17-4-18	Tue 17-5-2
21	Viva	3 days	Mon 17-5-15	Wed 17-5-17
22				
23	~~Semester 1~~	~~72 days~~	~~Thu 16-9-29~~	~~Fri 17-1-6~~
24	Security: RobTheShop	6 days	Tue 16-10-25	Tue 16-11-1
25	Security: RobPress	6 days	Thu 16-11-24	Thu 16-12-1
26	EM&L: Management CW	6 days	Fri 16-12-2	Fri 16-12-9
27	Machine Learning: CW	6 days	Mon 16-12-5	Mon 16-12-12
28	EM&L: Accounting CW	6 days	Fri 16-12-9	Fri 16-12-16
29				
30	Christmas	2 days	Sat 16-12-24	Mon 16-12-26
31	Revision	17 days	Sun 16-12-18	Sun 17-1-8
32				
33	Exam: EM&L	3 days	Wed 17-1-18	Fri 17-1-20
34	Exam: Machine Learning	3 days	Thu 17-1-19	Mon 17-1-23
35	Exam: Security	3 days	Wed 17-1-25	Fri 17-1-27
36				
37	~~Semester 2~~	~~115 days~~	~~Mon 17-1-9~~	~~Fri 17-6-16~~
38	Game Design: Sprint 1	4 wks	Mon 17-2-13	Fri 17-3-10
39	Game Design: Sprint 2	6 wks	Mon 17-3-20	Fri 17-4-28
40	Game Design: Sprint 3	4 wks	Mon 17-5-1	Fri 17-5-26
41				
42	Secure Systems: Lab 2	2 wks	Thu 17-2-2	Wed 17-2-15
43	Secure Systems: Lab 4	2 wks	Thu 17-2-16	Wed 17-3-1
44	Secure Systems: Lab 6	2 wks	Fri 17-3-10	Thu 17-3-23
45	~~Secure Systems: Lab 8~~	~~2 wks~~	~~Unknown~~	~~Unknown~~
46				
47	Easter	2 days	Sat 17-4-15	Mon 17-4-17



Above is the final Gantt chart showing this project's progress. As can be observed, progress slowed until towards the end wherein progress rapidly accelerated. I will explain this inconsistency in my conclusion.

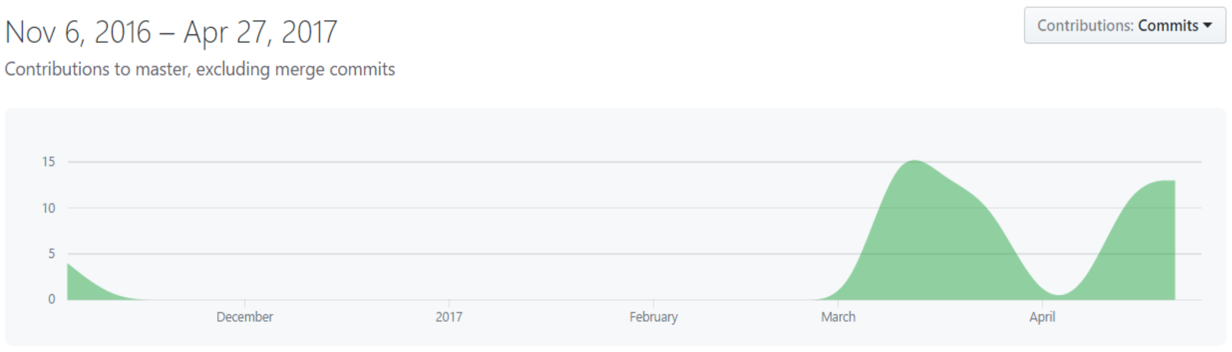
5.2. Version Control

Throughout this project, I used GIT to both keep a reliable remote history, and to help quantify the work done. In the table below, I have listed the repositories used, a brief description of them, and how many commits were made over the history of this project.

Repo	Description	Commits
Master	The main repository, containing the entire history of the implementation of this project. Every piece of the code is in here.	**83**
Docker	Branch of the main repo containing the Dockerfile, Ningx config, along with a bundle of support scripts to help build and reliably deploy the application to the web.	10
Soton	A large monolithic repo containing miscellaneous pieces of coursework from my entire time at university. Here I stored the progress report and the final report while I worked on them.	Unknown

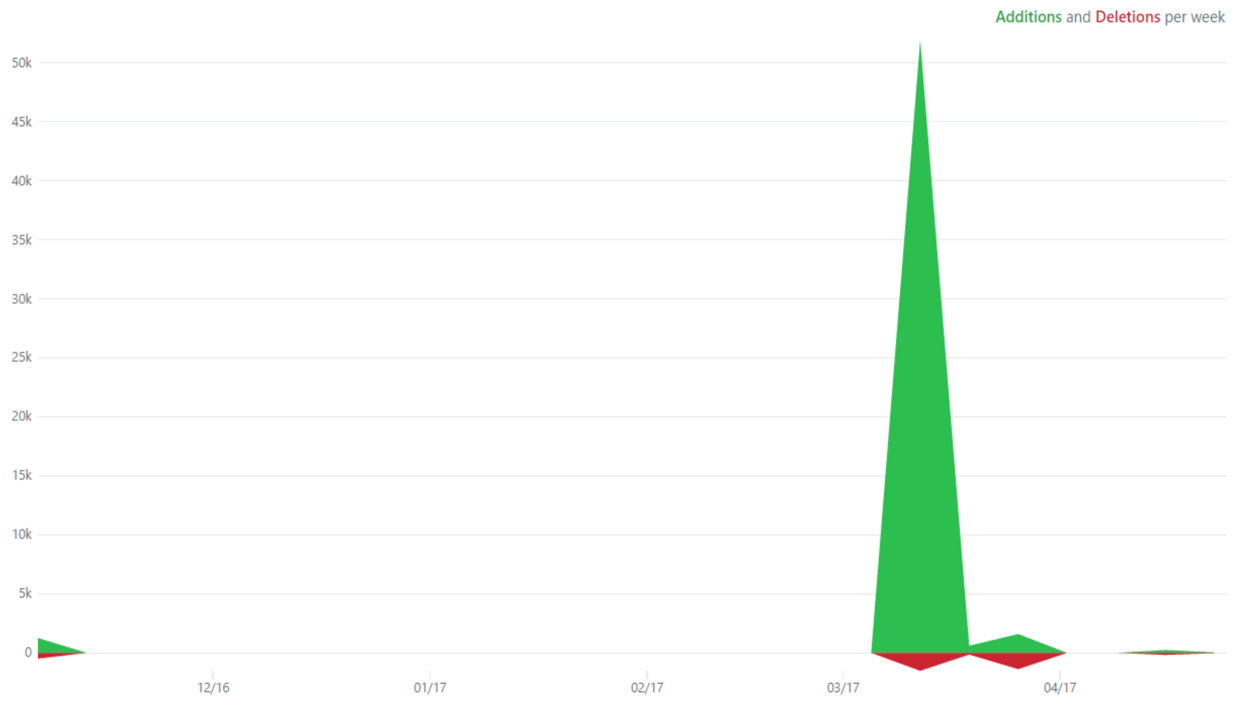
Resulting from this split, the archive I will be submitting will contain both Master and Docker in separate folders, with a note explaining the build process.

5.2.1. Commits



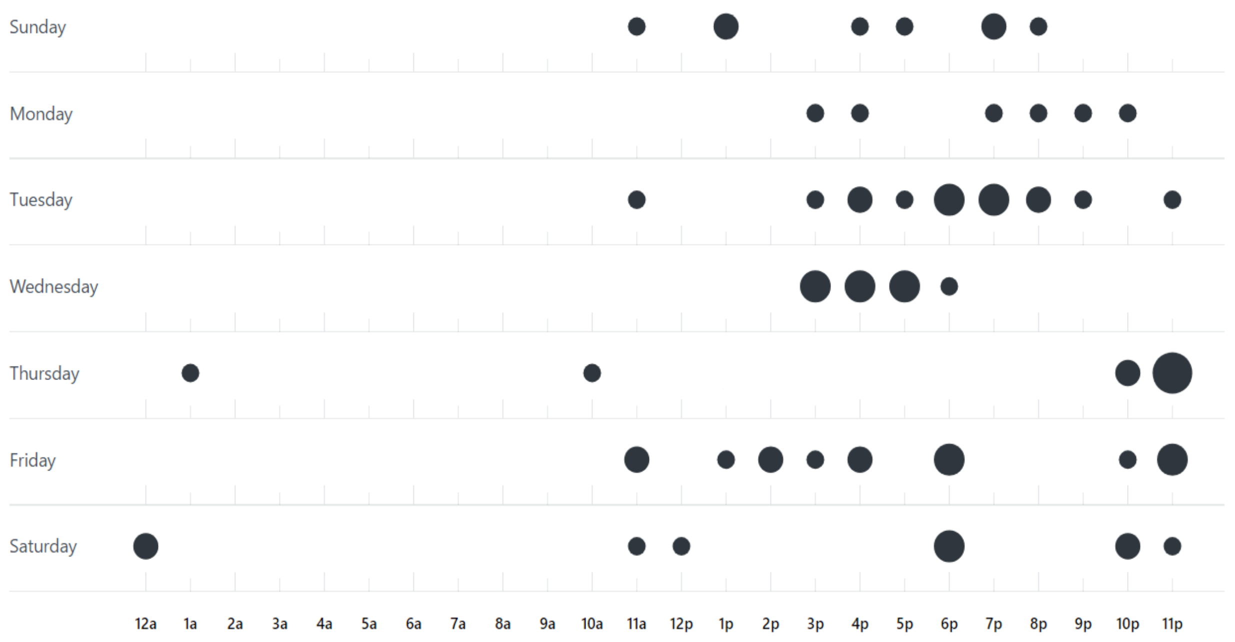
This graph shows the number of commits per week. There is a sizable gap in the middle where I was a little too focused on other coursework and activities. While this was expected to some extent, in the future I should work harder to stay on track. Fortunately, I managed to put in the work and finish my project.

5.2.2. Code Frequency



The graph above shows the number of additions compared with the number of deletions per week. It is mostly symmetrical with a small bias towards additions. I suspect the large bump was a result of me importing ViewerJS and other files into the public directory.

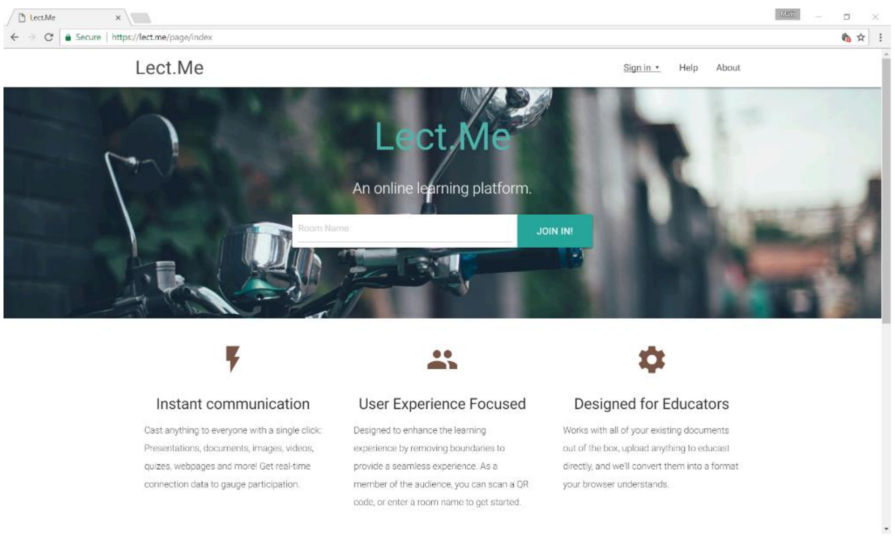
5.2.2. Punch Card



Above is a punch card of commits divided into time of day. Virtually all the commits fall into a 10am to 12am slot, with a single exception. Oddly enough, it seems I did not commit much on Thursdays. I suspect it is due to Thursday usually being coursework submission day.

6. Implementation

6.1. Screenshots



Instant communication

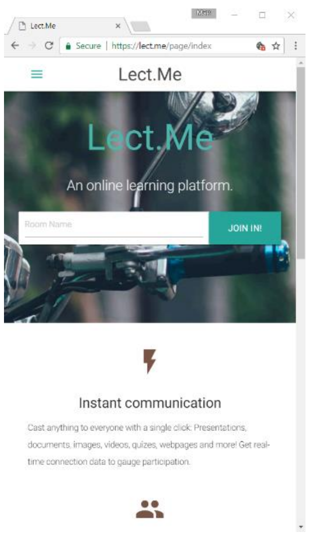
Cast anything to everyone with a single click: Presentations, documents, images, videos, quizes, webpages and more! Get real-time connection data to gauge participation.

User Experience Focused

Designed to enhance the learning experience by removing boundaries to provide a seamless experience. As a member of the audience, you can scan a QR code, or enter a room name to get started.

Designed for Educators

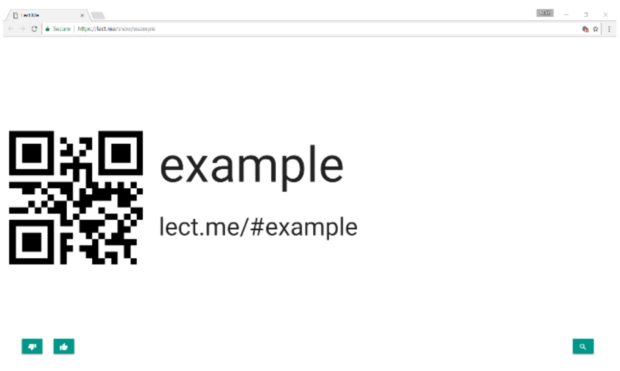
Works with all of your existing documents out of the box, upload anything to educast directly, and we'll convert them into a format your browser understands.

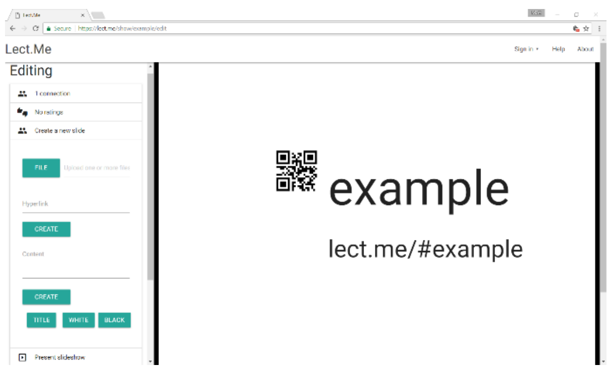


Instant communication

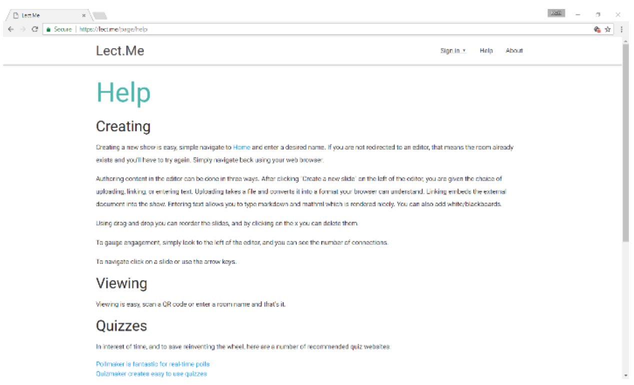
Cast anything to everyone with a single click: Presentations, documents, images, videos, quizes, webpages and more! Get real-time connection data to gauge participation.

The landing page, showing how the responsive design adapts to different screen sizes. Please note that the theme was adapted from the Parallax template from Materialize CSS [31].





The presentation and editor pages. I tried to keep somewhat close to the mock-ups. That design itself was based upon existing presentation software; I wanted to create a familiar design.



Help

Creating

Creating a new show is easy, simple navigate to Home and enter a desired name. If you are not redirected to an editor, that means the room already exists and you'll have to try again. Simply navigate back using your web browser.

Authoring content in the editor can be done in three ways. After clicking 'Create a new slide' on the left of the editor, you are given the choice of uploading, linking, or entering text. Uploading takes a file and converts it into a format your browser can understand. Linking embeds the external document into the show. Entering text allows you to type markdown and mathml which is rendered nicely. You can also add white/blackboards.

Using drag-and-drop you can reorder the slides, and by clicking on the x you can delete them.

To gauge engagement, simply look to the left of the editor, and you can see the number of connections.

To navigate click on a slide or use the arrow keys.

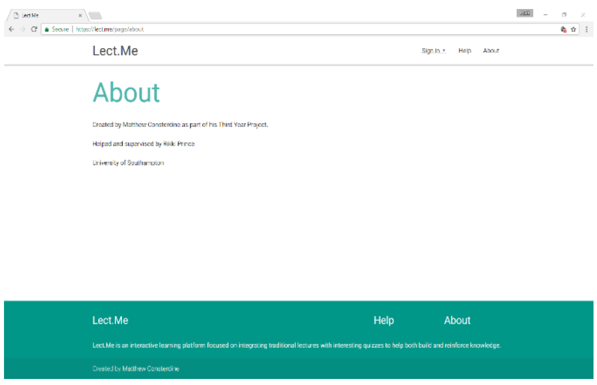
Viewing

Viewing is easy, scan a QR code or enter a room name and that's it.

Quizzes

In interest of time, and to save reinventing the wheel, here are a number of recommended quiz websites

Pollmaker is fantastic for real-time polls
Quizmaker creates easy to use quizzes



About

Created by Matthew Consterdine as part of his Third Year Project.

Helped and supervised by Rikk Prince

University of Southampton

Lect.Me Help About

Lect.Me is an interactive learning platform focused on integrating traditional lectures with interesting quizzes to help both build and reinforce knowledge.

Created by Matthew Consterdine

Lastly, I added a help and about page to give guidance to new and experienced users. This is accessible from two links in the header, and in the footer.

6.2. Templates

By default, Meteor does not come with a built-in layout engine. It follows the JavaScript and NPM philosophy of lots of small micro-packages where each package aims to do one thing well. This is not without downsides (A change to left-pad broke thousands of dependent applications last year [32]), but it does mean, "NPM is the largest package ecosystem ever" [33].

I chose to use BlazeLayout because it is designed for Meteor and allows attaching JavaScript code to each template in various different ways. I attach: an `onCreated` method to subscribe to the necessary databases; various helpers to call inside the templates to retrieve data; and miscellaneous event handlers to give the page interactivity.

This approach helps to organize the codebase by separating each template and its required scripts into different files from each other instead of one big monolith.

6.2.1. Templating

Meteor uses Spacebars for the default templating engine. It supports helper functions and importantly, `{{> Template.dynamic}}`. This crucial template is used throughout the web-app for slide rendering. To use `Template.dynamic`, supply the desired template name and data as named arguments. The alternative would be to use a long if, else chain inside its own template. `Template.dynamic` makes this process easy.

6.2.2. User Interface

The user interface is built from a hierarchy of templates, each building upon the last. This allows for a modular design with minimal dependencies. Each page uses the same automatically prepended HTML head for simplicity. As an example, the index page is built as follows:

```
<template name="layout_default">
  {{> header}}
  {{> Template.dynamic template=content}}
  <div id="footer-padding"></div>
  {{> footer}}
</template>
```

`header` and `footer` are two templates that are always included to keep the site consistent. `content` is a variable containing "page_index" which is included using `Template.dynamic`.

6.3. Data Types

From the start, I realized the need for creators to be able to both store and display many different datatypes. In the database, I store these datatypes as a JSON pair. Therefore, to represent an uploaded image, I use the following snippet:

```
{type: "picture", data: "http://lect.me/uploads/example.jpg"}
```

When viewing slides, the client gets given this pair to interpret using Meteor templates. First, the JSON pair is passed to the resource template, which matches each datatype with the correct template using the Spacebars `Template.dynamic` operator [34]:

```
<template name="resource">
  {{> Template.dynamic template=(concat "resource_" type) data=data}}
</template>
```

`resource_picture` is the simplest template. Here, it simply places data directly into the page.

```
<template name="resource_picture">
  <img src="{{data}}"/>
</template>
```

6.3.1. List of Data Types

Data Type	Description
Audio	Synchronized HTML5 audio player. Typically supports WAV, MP3, and AAC.
Blackboard	Simple blank black screen. Allows for writing on a whiteboard.
PDF	Uses ViewerJS for greater compatibility with more browsers.
Picture	Auto-resizing image that completely fills the screen, preserving aspect ratio.
Text	Type Markdown and MathJax with syntax highlighting support.
Title	The show's name, link to join and large scannable QR code.
Video	Synchronized HTML5 video player. Typically supports MP4 and WebM.
Webpage	Embed virtually any webpage inside the presentation.
Whiteboard	Simple blank white screen.
Undefined	Default template, displayed if given an invalid type. A blank screen.

6.3.2. Webpage limitations and X-Frame-Option

One limitation I noticed through the development was that certain webpages refused to load inside the `<iframe>`. On further inspection, I discovered that this was due to the HTML header: `X-Frame-Option`. Designed to protect websites against clickjacking attacks [35], this is by design impossible to overcome in a modern web-browser [36].

There are two work arounds for this problem, the first is to proxy all connections and embed that inside the iframe. While on the surface this appears to work for every website, it results in added server load, requires rewriting all external resources to use the proxy, itself does not work on all webpages, and is legally questionable. I chose not to take this approach.

The second is to design and create native apps as instead of using an `<iframe>` to embed other webpages, and just embed a webpage directly. This is however far beyond the scope of my third-year project, as even distributing it as an electron app would not cut it. To work, it would require significant re-engineering.

In the end, I added a note in Help explaining why certain websites could fail to load.

6.3.3. Quizzes

Originally, I planned to include Quizzes as a native part of the web-app, but later shelved that idea to focus on what was truly important in the web-app. The rationale behind the decision was while I could create my own polls and quizzes, there are so many fantastic ones out there that user's may already be familiar with; I could help creators to use them.

I decided to create a help page detailing not only how users can use the web-app, but links to useful polling and quizzing resources for them to include in their presentations.

6.4. Content Creation

There are four methods to insert a new slide into the show:

- Directly uploading a file to the web-app. The file is stored as its SHA256 hash to prevent different files colliding and overwriting (Note, I could not use SHA1 due Shattered, a recent collision attack [37]). The format is then identified, and matched it against a whitelist. If missing it do not proceed. Finally, a JSON pair is constructed and added to the database.
- Adding a slide from a website. It parses the URL to check if it is valid, and to extract a file extension. Again, it is compared against a blacklist to help prevent a malicious author linking to executable files. If the file is safe, it uses the whitelist to identify if it can be embedded directly, rewritten it to an embeddable URL, or should just use the webpage data type. Finally, it adds the JSON pair to the database.
- Entering Markdown, MathJax, or Code, simply adds it to the database as a JSON pair. The conversion from plaintext to nicely formatted text is done client side using marked [38], MathJax [39], and highlight.js [40].
- Clicking the Title, Whiteboard, or Blackboard buttons adds a JSON pair with no data.

Once created, the slides are quickly pushed to each connected client using WebSockets.

6.4.1. Black and White Listing

This is achieved using the same object that maps extension to format. When it checks the whitelist, it looks up the file extension and if it exists and does not return invalid, then it is whitelisted. To check the blacklist, it repeats the process except it only disallows when it returns invalid. If it does not exist then it is allowed.

6.4.1. URL Rewriting

Done using an object that maps regex with capturing groups to replacement `sprintf` URLs. Each time a new URL is entered, it is checked against the object, and if anything matches, it is replaced. The reasoning behind this rewriting is that some websites such as YouTube have special embed pages that display a nice, simplified layout. Sometimes these embed pages are the only ones that work inside `<iframe>`s.

6.5. User Statistics

To help the presenter gauge the response to their lecture, I added two useful statistics.

First, the ability to see how many clients are connected to the room at once is displayed in the left sidebar. This is done using heartbeats, so every minute each client sends a tiny packet to the server to confirm they are still alive and inside a specific room.

Second, is the ability for participants to rate the lecture, this is also displayed in the left sidebar. Each client starts each show with a rating of zero, when they rate up that rating is changed to plus one, and when they rate down, that rating is changed to negative one. Participants can change their rating throughout the lecture. Scoring is done by summing non-zero ratings, diving by their count, multiplying by 50, then adding 50 to get a score between 0% and 100%.

To give context, the number of ratings is displayed to the right of the overall rating.

6.6. Routing and Application Logic

By default, Meteor is a single page application with no knowledge of URLs. This means that in order to create any sort of modern website, I needed to pick a router. I picked FlowRouter as it works nicely with my templating engine BlazeLayout, and it nicely separates route definitions from the implementation of each page.

For example, with pages I can define a page group, and two child routes as follows:

```
var page = FlowRouter.group({prefix: '/page'});

page.route('/', {action: (p, q) => FlowRouter.redirect('/page/index')});
page.route('/:pageID', {action: (p, q) => {
  let file = 'page_' + Sanitizer.file(p.pageID);
  if(Template[file]) {
    BlazeLayout.render('layout_default', {content: file, ...});
  } else {
    FlowRouter.notFound.action(p, q);
  }
}});
```

The first route is a simple redirect; the second sanitizes the page name, checks if it exists, and then renders either it, or a 404 page. The above snippet has been edited for brevity.

6.6.1. Sitemap

- Root redirects to `/page/index`
- Pages are defined as: `/page/[about|help|index]`
- Shows are defined as: `/show/<roomID>/[edit|revise]`

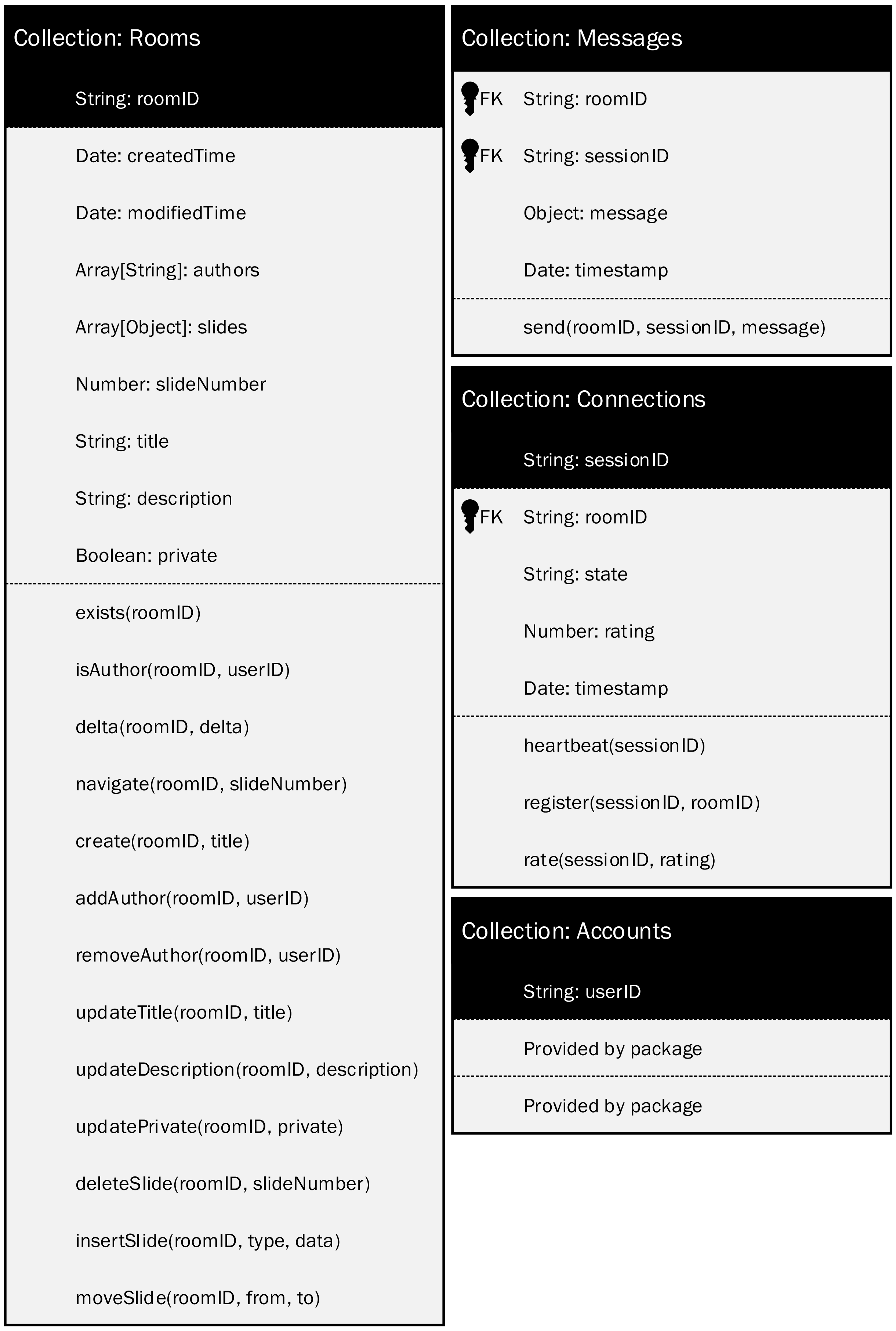
6.7. Database Design

As mentioned previously, I am using MongoDB as my primary data-store. MongoDB is a document, or NoSQL database. For this web-app, I settled on using four distinct collections:

Collection	Description
Rooms	The primary collection in the web-app, which stores each individual room. MongoDB is a document database, so Rooms stores each slide, the current slide position, and various pieces of metadata such as author.
Messages	Used to synchronize events between presenter and their audience. Therefore, whenever the presenter plays a video, or pauses music, the events are re-played on each participant with lag compensation. Now it can synchronously play a video to hundreds at once, assuming accurate system clocks.
Connections	Tracks individual connections, their activity and their current room. This is achieved by sending heartbeats from client to server every minute or so to check the client is active. For now, it only tracks the number of connected clients, but it could be easily adapted to track other data points.
Accounts	Automatically created and managed by Meteors fantastic Accounts package. Secure by default, and used for tying different rooms to different authors. This could also be extended to allow OAuth (Google, Facebook, etc.) logins.

As with any sufficiently complex application the database is not the only place information is stored. Pages such as index, help, and about are defined in their own templates for ease of development and maximum responsiveness. Files are stored on disk, and served using Nginx.

6.7.1. Database Diagram



Collection: Rooms
String: roomID
Date: createdTime
Date: modifiedTime
Array[String]: authors
Array[Object]: slides
Number: slideNumber
String: title
String: description
Boolean: private
exists(roomID)
isAuthor(roomID, userID)
delta(roomID, delta)
navigate(roomID, slideNumber)
create(roomID, title)
addAuthor(roomID, userID)
removeAuthor(roomID, userID)
updateTitle(roomID, title)
updateDescription(roomID, description)
updatePrivate(roomID, private)
deleteSlide(roomID, slideNumber)
insertSlide(roomID, type, data)
moveSlide(roomID, from, to)

Collection: Messages	
FK	String: roomID
FK	String: sessionID
	Object: message
	Date: timestamp
	send(roomID, sessionID, message)

Collection: Connections	
	String: sessionID
FK	String: roomID
	String: state
	Number: rating
	Date: timestamp
	heartbeat(sessionID)
	register(sessionID, roomID)
	rate(sessionID, rating)

Collection: Accounts
String: userID
Provided by package
Provided by package

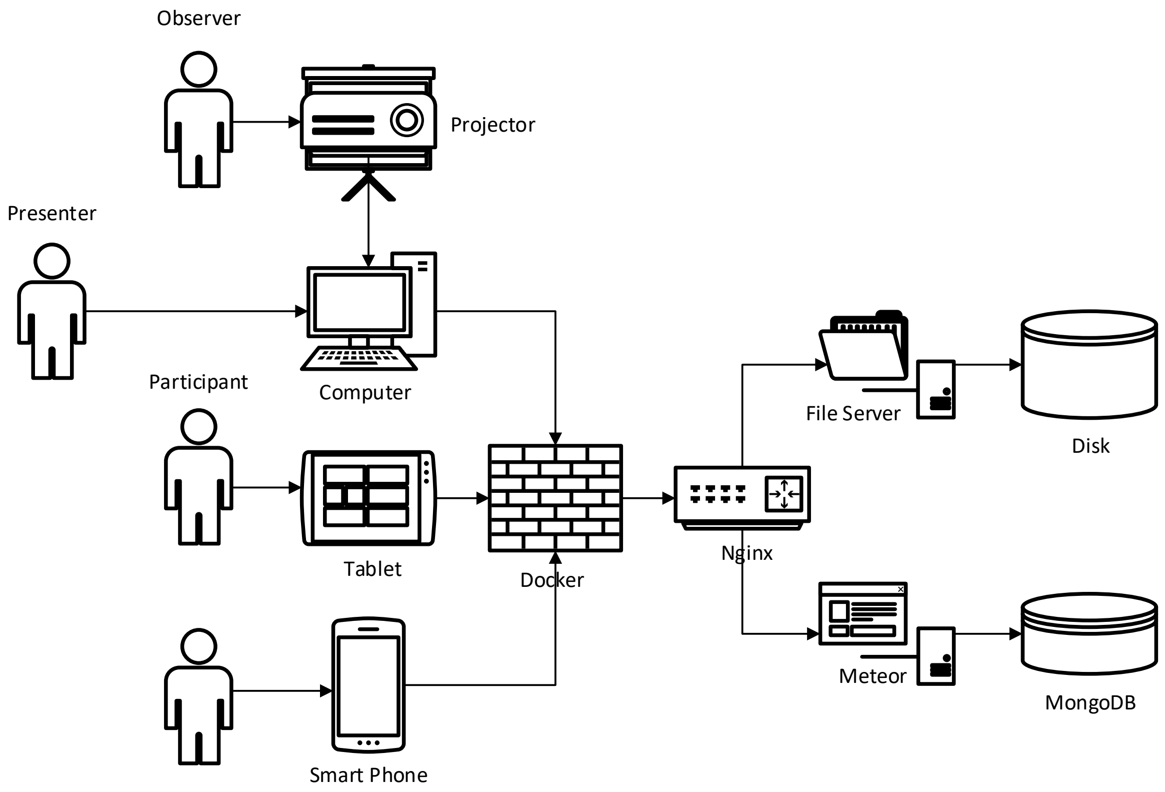
6.8. Server Architecture

At an early stage, I decided that creators needed to be able to upload files. This would be an easy task on a traditional web server, simply upload them into a folder, and serve it on the same domain. However, as Meteor automatically restarts your application when the files change in development, this was not an option.

I decided that I needed to run a second webserver simultaneously to Meteor. After looking around, I decided to deploy Ningx [41] as a file host and proxy to keep files on the same domain. Configuration was easy, but the setup had become complex and difficult to reproduce. This was an issue.

In production, Meteor does not automatically restart the application. However, I did not learn this until after deploying to Docker. I am still happy with my decision as it offloads work from the Meteor application server, to the Nginx file server and proxy. This design lets each server work to their strengths, to achieve greater performance. In this case, the added complexity is a good thing.

6.8.1. Systems Diagram



Above is a diagram showing the completed system. While from the surface it seems complex, it is built using Docker meaning reproducibility and complete autonomy of the setup process.

I have included an array of users interacting with many different devices, to highlight the broad compatibility of the system, as it was built for the web.

6.8.2. Deployment

After much research, I settled on using Docker. Using the latest Debian base image, I cloned the project folder, installed the necessary software, and configured it to run on image start-up. This fully automatic setup allows me to easily, reliably deploy both testing environments and update production with the latest version.

As a helping hand, I produced a set of scripts to automate and minimize user error:

Script	Description
build.sh	GIT clone the latest version, install NPM scripts, then build meteor for linux64.
copy.sh	Copy the tar.gz build to the server using SCP.
install.sh	Building the Dockerfile.
run.sh	Run the built Docker VM only exporting port 80.
startup.sh	Sets the server time. Starts Nginx, MongoDB, set $ENV, and launch the bundle.

To deploy for both demonstration purposes and my Viva I used the Docker image to deploy it to a DigitalOcean droplet, due to time concerns. Unlike the universities VM self-service, DigitalOcean's system allowed me to instantly create a new virtual machine. The second reason I chose DigitalOcean is that GitHub's education pack provides a free $50 credit [21].

Once I had hosting, the next thing I needed was a domain name. An IP address, while functional, is hard to type or remember. I settled on the domain name: Lect.Me as it was short (for lecture), snappy, and easy to remember. It also was part of the GitHub education pack [21].

It was time for DNS and I chose Cloudflare due to features, familiarity and ease of use.

Lastly, for TLS I used the GitHub Education pack to get a free certificate, after submitting my CSR, I validated the domain using DNS, concatenated the *.crt and *.ca-bundle files, and added the certificate to Nginx. I made a backup of the private key to be safe. While this setup is unlikely to protect against a nation state, it will protect against packet sniffing attacks such as Firesheep [42], and Man-in-the-Middle attacks such as Ettercap [43].

6.8.3. Nginx Configuration

Setting up Nginx as a file host and proxy for a WebSocket based application was rather difficult.

After much trial and error, I found that these blocks would need to be present for WebSockets to work. The second block sets up the proxy, and first block is used to proxy the WebSockets. Without both, Meteor falls back on a slower communication method.

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

location / {
    proxy_pass http://127.0.0.1:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_set_header X-Forwarded-For $remote_addr;
}
```

6.8.4. System Requirements

Through testing, I determined that the machine or VPS the web-app is running on needs a decent CPU; at least 1GB of RAM for running Nginx, Meteor, and MongoDB; along with enough storage space for uploads. Ideally, disk space usage should be monitored to determine if and when it needed to be cleaned up, or expanded.

As the main web-app runs inside Docker it should in theory run on any supported OS, however this is somewhat untested. If run using Docker Toolbox instead of Docker Native, it runs inside a VirtualBox VM with its own network interface. Therefore, to allow external connections, the host needs to be set to act as a proxy.

On Windows, for example:

```
netsh interface portproxy add v4tov4
  listenaddress=127.0.0.1 listenport=80
  connectaddress=129.168.99.11 connectport=80
```

6.9. Security

By default, Meteor is setup for prototyping, meaning rather insecure. However, this is an easy fix as the insecurities are isolated into two packages; the aptly named `insecure` package that allows unrestricted database writes from the client, and the `autopublish` package which lets the client read everything. Fortunately, both are easy to remove with `meteor remove insecure autopublish`, but in future, `meteor create -bare` creates a new project without them being pre-installed [44].

When data is published from the server to the client, Meteor can use the find method to restrict access to public or authored shows, the current room, or active connections. For example:

```
if(Meteor.isServer) {
  // Runs only on the server
  Meteor.publish('rooms', function() {
    return Rooms.find({$or: [{private: false},{authors: this.userId}]});
  });
}
```

As previously mentioned, I am using the built-in Accounts package, which uses industry best practices of salted `bcrypt` by default. It is easily extensible, but its base configuration is good enough for now.

By using forced TLS, the application is protected against packet sniffing attacks such as Firesheep [42], and Man-in-the-Middle attacks such as Ettercap [43].

Lastly, I should mention that it would be disingenuous to state that the web-app itself is truly secure and ready for wide-scale deployment. While I have deployed it for demonstration purposes, it is not yet intended for serious usage. This is because it was developed more as a proof of concept than a secure product, thus has yet to undergo proper penetration testing.

7. Testing

During my project, while I did not use formal unit testing, I did however test it in various ways. As I was developing more of a proof of concept web-app, I decided that unit testing would be more effort than it was worth, when compared with further development and getting user feedback. Future work could include adding both automatic unit and integration testing to help deliver quality software, although I will talk more about this in a later chapter.

One way I tested the web-app was as I developed, I was checking each feature worked as intended – there is little point adding a feature if it has not been verified to ensure that it is working and has been implemented correctly! This process of trial and error continued throughout development. This was made possible by Meteor auto-reloading on file change, which allowed quick testing without recompiling the web-app from scratch.

7.1. Integration Testing

Without complicated scripting, it is hard to perform integration testing of a complex interactive system. However, unlike unit testing, it is significantly easier to perform manually. So, in order to work around that, I developed the following checklist that I manually ran through after deploying a major change:

1. The main index page loads up as expected. This tests that Cloudflare, DigitalOcean, Docker, Nginx, Meteor, and MongoDB are all working correctly.
    a. If the site fails to load entirely with no response, this means that there is an issue with Cloudflare, as it is designed to give a response with its Always Online technology.
    b. If the site displays a host offline error, typically Error 521/522, that means that there is an issue with DigitalOcean, Docker, or Nginx. The first can be verified by logging into the DigitalOcean dashboard and confirming the VPS is online. The second can be verified by connecting to the VPS via SSH and running `docker ps` to list running Docker containers. If the container is running, then it has been deduced to error with Nginx.
    c. If the site displays a gateway error, typically Error 502, it means that Nginx is running as intended, but cannot connect to Meteor. To resolve connect to the VPS, attach to the Docker container with `docker exec -t -i $image_name /bin/bash`, and check the logs.
2. Create a new user account, which persists between browser sessions. This tests Local Storage, and that MongoDB is working and allowing both database reads and writes.
    a. If it fails, ensure MongoDB is running correctly behind the scenes by attaching to the Docker container, and using the mongo client.
3. Each different method of content creation works, and the changes are quickly synchronised to the client. This tests file uploading, slide insertion, the event queue, and connection pool.
    a. If file uploading fails it is probably a permissions error, check the uploads folder exists and can be written to. If not, check the logs.
    b. If creation fails, check the logs as it is likely an issue with the JavaScript code that is executed before or after slide creation. Alternatively, you are not logged in.
    c. If the synchronisation is slow, try recreating the issue on localhost, to determine if it is the software or the network causing the fault. If it is the network, look into altering the Nginx config, VPS location, or Cloudflare caching behaviour.

Once run, these three tests allowed me to be confident that the web-app was working as intended, and if not the steps noted would put me back on the right path.

7.2. User Testing

Once the web-app was ready to demonstrate, I performed user testing. This was done after I demonstrated the web-app, and before I asked for feedback. I chose not to do blind user testing, where I just throw the user in and see how they behave to save time, as a short ‘guided tour’ of the web-app would answer should answer most of the obvious questions. After further development, it would be advisable to both increase the breadth of the help topics available, and to have blind user testing to help identify potential UX improvements.

Results of the testing will be discussed in the next chapter, but overall I can say I am happy with the approach taken, as it was successful in identifying both strengths and weaknesses. While the participants liked the simplicity of the user interface, they found the participation options somewhat lacking.

The users interviewed consisted of teaching staff at the University of Southampton. Due to them all being consenting adults, and the short, safe nature of the testing, I decided that I did not need to seek Ethics approval. If had decided that I needed to ask children, or ask for them to trial the software in actual lectures, my conclusion would obviously have been different.

8. Evaluation

To evaluate, I decided I needed to acquire feedback from potential users in addition to formally evaluating the requirements and risk analysis. This further external feedback is crucial in understanding what parts of the system are a success, and what parts needs further work.

8.1. Acquiring Feedback

I drafted a questionnaire and sent emails to many ECS lecturers asking if we could meet and discuss the project; I got several responses. When I met up with them, I started by giving a short demonstration of the web-app, highlighting what I thought they would find the most interesting. I then asked them a series of questions and transcribed their responses. I have reproduced the full transcripts in the Appendices.

I designed the questions to gather feedback both on the current state of the web-app, alongside improvements that would help a potential creator use it in the future. They are as follows:

1. What do you like about the system?
2. Could you see yourself using this system, or one like it?
3. What would you change, given the chance?
4. Are there any other statistics you would like gathered?
5. Do you have any suggestions to increase participation and interaction?
6. Any other thoughts?

Before starting, I knew I was unlikely that I would have a large sample size, so I decided against using statistical analysis. As this was not a finished product, and still rough around the edges, this is not too big a loss. Ideally, it would have been nice to have a sample size at least twice as big but that is work for the future.

8.2. Questionnaire Results

The questionnaires proved extremely useful. Getting a fresh perspective helped me see it from a fresh angle, and helped to highlight both how it would be used, and how it could be improved. In total, I emailed many academics, and ended up interviewing four of them.

Now I will summarise the points made:

- Overall, the lecturers liked the simplicity and usability of the user interface. This was a focus of mine, as I wanted it to be quick to understand, especially for viewers. I based the interface of a template from the Materialize CSS framework, with some tweaking.
- Accessibility was very important, and as I stuck to standard HTML5 it works correctly with browser zooming and extensions. One way I could have improved this, would be adding aria tags for screen readers.
- Most lecturers saw being able to upload any file format as a key feature. I am happy with the support I added, however I do wish I had done more to support certain plain text formats. So for example, you could just upload source code directly without copying and pasting it into the text content box.

- The synchronisation feature was universally appealing. Although, one commented that it would have been a good idea to allow viewers to also view different slides at their own pace. I agree, and decided to implement it.
- While most saw it as a good framework for starting from, they could not really see them using it until more interactive features were added to the web-app. This is understandable so I acted on some of feedback to improve the site.
- Scalability was questioned and as it was untested I had no real answer for that. The most I can say it is stable with over a week's uptime, and is able to support at least five clients with no distinguishable slowdown. My guess is that it could support at least 100 clients due to the design of the web-app. When navigating slides, Meteor only sends a small JSON packet to each client, with the heavy lifting done by Nginx. Caching could be used to reduce load.
- It would have been nice to allow live editing, one lecturer commented. In its current state, the only way to edit is to delete the slide, and re-upload. This would have been possible with some UI work.
- Adding games as a feature, similar to the Jackbox Party Pack, would be a nice feature to add to help expand the usefulness of this web-app beyond lectures.
- Asking students for feedback as well as lecturers would have been a good idea. I could have written a questionnaire and performed statistical analysis on the results.
- While the QR code was liked, one worry was that participants would feel the need to walk up the projected image. One solution brought up was handouts.
- Students need to be able to revise from lecture material. This feature was added.
- The domain name Lect.Me was a hit, easy to remember, with a short link to connect.

After listening to feedback, I implemented several suggestions such as removing the navigation buttons to help improve focus, to tidying up the user interface, adding in a revision feature, and allowing users to thumbs up/thumbs down to provide real-time feedback. More changes would have been made given more time, and this has been described in Potential Future Work.

8.3. Evaluating Risks

The risk assessment was useful as it helped me plan around my inexperience taking part in a long term, personal project. This was done by slightly reducing the scope moving quizzes from an integrated part of the web-app, to dependence on external resources. In my opinion, this was a good choice.

When deployed I did notice slightly increased latency, however this is to be expected when moving a project from LAN to the internet. Besides that, everything performed as expected, even if it required 1GB of RAM to run. While work could be done to reduce this figure, RAM is cheap, and additional development would be better suited on adding more features.

Fortunately, I did not suffer any significant illness, my data was kept safe with version control, technology did not radically change, and I managed to test and evaluate my project. In my project, I was fortunate enough to not really suffer from any unforeseen problems, my risk assessment proved a success.

8.4. Evaluating Requirements

The following list corresponds to the table in section 3.4.1. For brevity, if an item is missing, it was an optional requirement, and was not implemented. Some requirements are combined.

8.4.1. Data Protection Act of 2003

The only data stored is account name, hashed account password, and slide contents. The above are used both fairly and lawfully for their one specific purpose. It is entirely user generated, and is as accurate as possible. Proper firewalling is used in the form of Docker, with security measures to prevent database dumping. The VPS chosen resides inside the United Kingdom and will not be transferred to a different region. While slides can be deleted, rooms and users cannot be automatically, so I have provided contact information for manual removal. I am not a lawyer.

8.4.2. European Union Cookie Law

I embed a script from Cookie Consent by Insites [45], which notifies that cookies are used. If the user does not want to be tracked by cookies, they can leave the site and watch the projector instead. Alternatively, they can block cookies which allows them to use the site, but prevents them from logging in and creating content.

8.4.3. Data integrity

Data integrity was achieved by using both server and client side validation. This is necessary as it is impossible to trust the user is not an attacker trying to break your system.

8.4.4. Useful Statistics and Logging

The web-app provides the presenter with a live count of connected users, and their current rating. Output and error logs are saved to disk for debugging and for identifying misuse.

8.4.5. Accessibility

I have developed a native web-app that will work on any device with a modern web browser. While it is accessible to the deaf as there is little to no audio, and those with poor sight can magnify a screen right in front of them, screen readers have not been tested or designed for.

8.2.6. Concurrent Users

Tested to support many concurrent users with over a week's uptime, hundreds of users is untested due to difficulty in testing it. It would be good next step to find this information out.

8.2.7. Help Topics

I have created a Help and About page to make the web-app easier to use. If that is not sufficient, I have added contact details to let users bring up any other issues they are having.

8.2.8. Content Types

The database and templating system handle the data types generically; it is easy to extend.

8.2.9. Quality of Life

The web-app works well and feels good to use, it is reliable and stable with so far with over a week's uptime. It is fast and responsive on LAN with marginally latency on the web. It uses the Accounts package to securely handle passwords and access controls.

8.2.10. Unit Testing

Unit testing was not used due to complexity and time constraints. Manual integration testing and user testing were used instead to similar effect. A checklist is given in the Testing chapter.

8.2.11. Revision

Students can revise from a lecture by opening it up, and clicking the magnifying glass button.

8.5. Budget

The budget was not used, with free alternatives to the listed paid services used instead. GitHub's education pack [21] was used to acquire the domain and hosting. The reasoning behind this was speed and ease of use.

8.6. Summary

In summary, the project has been evaluated using feedback from lecturers, understood its risks, met its requirements, and remained well within budget. Evaluating helped me to understand the strengths and weaknesses of my web-app, and where to go next. After further development, I would like to evaluate it by demonstrating it to, then by asking students to complete a short, quick questionnaire; this would allow statistical analysis.

9. Potential Future Work

As with any big project, there is typically work left that could be done. I have listed several examples below to show how this web-app could be extended in the future.

9.1. A Focus on Security

Right now, the web-app was not designed with security in mind, and has not been penetration tested. I would not recommend putting it immediately in production, and instead perform a security sweep and refactoring to improve the security and reliability of the system. This is not to say that the system itself is unreliable, as I have deployed it for over a week without any such problems.

9.2. Collaborative Editing Tools

Currently, short of sharing credentials, there is no way to work collaboratively. A focus on tooling to both allow creators to manage authors, and editing would go a long way to improving the overall usefulness of the web-app. It would also benefit single creators.

9.3. Adding Integrated Quizzes

One of the features cut to focus on the core product was integrated quizzes. Therefore, a good place to expand would be adding these to the web-app instead of depending on third party sites. This reduction of external dependencies would help improve flow and interaction. A creator would be able to manage not only the presentations, but also quizzes, all in one site.

9.4. Gathering Additional Statistics

Currently, the only statistics gathered are number of active connections and current user feedback; these are discarded after use. One piece of future work would be to graph user engagement throughout the lecture, and replay it with the different slides. There are so many different statistics that could be measured to help improve lectures.

Once this was in place, a dashboard could be created to bring everything together.

9.5. Developing Native Applications

While I do not regret focusing on a web-app, native applications tend to have the upper hand in some respects. They take longer to develop, but integrate properly with the target platform and can even be listed in app stores. As they are displayed as an app on a home screen, once installed, users are more likely to continue using them than a random web-app.

One issue that I encountered in this project was that some websites failed to load correctly inside the `<iframe>`. This is due to the `X-Frame-Options` header restricting the browser from loading them. I suspect that if the header was disobeyed, most of the sites would use JavaScript to detect if they are running in an `<iframe>` and halt execution. By switching to a native application, instead of embedding the website in an `<iframe>`, it would be embedding it inside a web view. This would allow embedding any site without having to worry about `X-Frame-Options`.

One possible way to start developing native applications, without a complete rewrite, would be to leverage electrify [46] to help bridge the gap.

10. Conclusion

In conclusion, I believe my project was largely a success. I designed and developed, with user feedback, a prototype web-app for improving lectures by helping build audience participation. There is still more work to be done before it can compete with other existing commercial systems, but the core functionality is implemented, and there is a clear path forward as noted in Potential Future Work.

It is a fully responsive web-app that works well on any device with a modern web browser. It supports and combines a large variety of different content types including text, pictures, pdfs and documents. Authoring tools are there for the creator, with the presenter able to seamlessly switch between anything and everything, gathering real-time statistics in the process. The user interface is simple and easy to use. After a presentation is over, participants can revise from the slides by clicking on the magnifying glass.

While quizzes are not a native part of the web-app, I am satisfied with embedding them in for this prototype. Although, adding quizzes directly as a native part of the web-app would help with consistency, interactivity, and appeal. Further statistics should be gathered.

10.1. Personal Reflection

Overall, I am happy with how the project has turned out. Every project has its ups and downs but in the end, I managed to deliver a good piece of software, and write this report about it. Along the way, I learned a great deal about both working on lengthy projects, and the JavaScript ecosystem in general. Before working on this project, my knowledge of JavaScript was limited to small, fun web-apps without any server component. Since then, I have developed and deployed not just this project, but multiple NPM modules [47].

In future projects, I need to focus harder on research, and organisation. Research, be it reviewing literature, or interviewing stakeholders is of crucial importance when it comes to designing, developing, and evaluating any product. Lacking time management did contribute towards unnecessary stress. If I had structured my time better, and kept to it, I could have implemented at least one of the features listed in potential future work. However, while the situation was not ideal, I did manage to get the development, feedback, and report all delivered on time.

References

[1] S. Thiagarajan, Thiagi's Interactive Lectures: Power Up Your Training with Interactive Games, American Society for Training & Development, 2005.

[2] Google, “Introduction - Material Design Guidelines,” [Online]. Available: https://material.google.com/. [Accessed 14 Oct 2016].

[3] t. a. a. :. a. r. o. B. t. o. e. o. A taxonomy for learning, A taxonomy for learning, teaching, and assessing, New York: Longman, 2001.

[4] E. Mazur, “Peer Instruction: Getting Students to Think in Class,” in *The American Institute of Physics*, New York, 1997.

[5] R. C. C. a. R. E. Mayer, E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning, John WIley & Sons, 2016.

[6] E. G. a. A. S. Tsvetozar Georgiev, “M-Learning - a New Stage of E-Learning,” in *International Conference on Computer Systems and Technologies*, CompSysTech, 2004.

[7] Nearpod, “Nearpod - Create, Engage, Assess through Mobile Devices,” [Online]. Available: https://nearpod.com/. [Accessed 1 5 2017].

[8] Top Hat, “Welcome to the new classroom,” [Online]. Available: https://tophat.com/. [Accessed 1 5 2017].

[9] Sarah Honeychurch, “Digital Student Survey 2015,” Learning and Teaching Centre, Glasgow, 2016.

[10] Statista, “Smartphone market share of leading mobile operating systems in the United Kingdom in 2015,” [Online]. Available: https://www.statista.com/statistics/271325/. [Accessed 9 Dec 2016].

[11] A. Recarey, “The significance of Java RMI please?,” [Online]. Available: http://stackoverflow.com/a/14327231. [Accessed 9 Dec 2016].

[12] Eevee, 9 Apr 2012. [Online]. Available: https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/.

[13] Facebook, “Introducing Hack - A Programming Language for HHVM,” 20 Mar 2014. [Online]. Available: http://hhvm.com/blog/4223/introducing-hack-a-programming-language-for-hhvm.

[14] G. Manricks, Instant Meteor JavaScript Framework Starter, Packt Publishing, 2013.

[15] D. M. L. a. P. C. Cohen, “Agile software development.,” *DACS SOAR Report,* no. 11, 23 Oct 2003.

[16] Coley Consulting, “MoSCoW Prioritisation,” [Online]. Available: http://coleyconsulting.co.uk/moscow.htm. [Accessed 1 5 2017].

[17] J. Woulds, A Practical Guide to the Data, London: The Constitution Unit, 2004.

[18] Information Commissioner's Office, “Guidance on the rules on use of cookies and similar technologies,” Information Commissioner's Office, 2012.

[19] T. P. G. a. W. Steve Faulkner, “ARIA in HTML,” 21 Nov 2016. [Online]. Available: https://www.w3.org/TR/html-aria/.

[20] Electronics and Computer Science at the University of Southampton, “VM Self Service,” [Online]. Available: http://selfservice.ecs.soton.ac.uk/. [Accessed 8 Dec 2016].

[21] Github, Inc., “Student Developer Pack: The best developer tools, free for students,” [Online]. Available: https://education.github.com/pack. [Accessed 8 Dec 2016].

[22] A. Ford, “What is the average cost of a domain?,” 20 Apr 2011. [Online]. Available: https://www.quora.com/What-is-the-average-cost-of-a-domain/answer/Alex-Ford-1.

[23] Internet Security Research Group, “Let’s Encrypt,” [Online]. Available: https://letsencrypt.org/. [Accessed 8 Dec 2016].

[24] M. Prince, “Introducing Universal SSL,” 29 Sep 2014. [Online]. Available: https://blog.cloudflare.com/introducing-universal-ssl/.

[25] Google, “Get Started with Publishing,” [Online]. Available: https://developer.android.com/distribute/googleplay/start.html. [Accessed 8 Dec 2016].

[26] University of Southampton, “ERGO : Ethics and Research Governance Online,” [Online]. Available: https://ergo.soton.ac.uk/. [Accessed 12 Dec 2016].

[27] L. S. LW Anderson, Bloom’s Taxonomy, Old Dominion University, 1994.

[28] MongoDB, “mongoDB | For giant ideas,” [Online]. Available: https://www.mongodb.com/. [Accessed 10 Dec 2016].

[29] Meteor, “Users and Accounts | Meteor Guide,” [Online]. Available: https://guide.meteor.com/accounts.html. [Accessed 12 Dec 2016].

[30] S. Greif, “Meteor & Security: Setting the Record Straight,” 18 Jun 2016. [Online]. Available: https://discovermeteor.com/blog/meteor-and-security/.

[31] MaterializeCSS, “Getting Started - Materialize,” [Online]. Available: http://materializecss.com/getting-started.html. [Accessed 27 4 2016].

[32] C. Williams, “How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript,” The Register, 23 3 2016. [Online]. Available: https://theregister.co.uk/2016/03/23/npm_left_pad_chaos/.

[33] D. Shaw, “npm is Massive,” Node Source, 1 7 2015. [Online]. Available: https://nodesource.com/blog/npm-is-massive/.

[34] Meteorcapture, “Understanding Spacebars,” [Online]. Available: http://meteorcapture.com/spacebars/#template-dynamic. [Accessed 26 4 2017].

[35] OWASP, “Clickjacking Defense Cheat Sheet,” [Online]. Available: https://owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet. [Accessed 17 8 2016].

[36] Can I use?, “Can I use X-Frame-Options HTTP header?,” [Online]. Available: http://caniuse.com/x-frame-options. [Accessed 25 4 2017].

[37] E. B. P. K. A. A. a. Y. M. Marc Stevens, “The first collision for full SHA-1,” [Online]. Available: https://shattered.io/static/shattered.pdf. [Accessed 26 4 2017].

[38] chjj, “marked - A full-featured markdown parse and compiler, written in Javascript. Built for speed.,” 30 6 2016. [Online]. Available: https://npmjs.com/package/marked.

[39] MathJax, “MathJax,” [Online]. Available: https://mathjax.org/. [Accessed 26 4 2017].

[40] highlight.js, [Online]. Available: https://highlightjs.org/. [Accessed 26 4 2017].

[41] Nginx, “NGINX | High Performance Load Balancer, Web Server, and Reverse Proxy,” [Online]. Available: https://www.nginx.com/. [Accessed 1 5 2017].

[42] codebutler, “Firesheep,” [Online]. Available: https://codebutler.github.io/firesheep/. [Accessed 25 4 2017].

[43] Ettercap, “WELCOME TO THE ETTERCAP PROJECT,” 18 5 2016. [Online]. Available: https://ettercap.github.io/ettercap/.

[44] Meteor Issue Tracker, “Meteor create --bare should not add the insecure and autopublish packages #8145,” 2016 Dec 12. [Online]. Available: https://github.com/meteor/meteor/issues/8145.

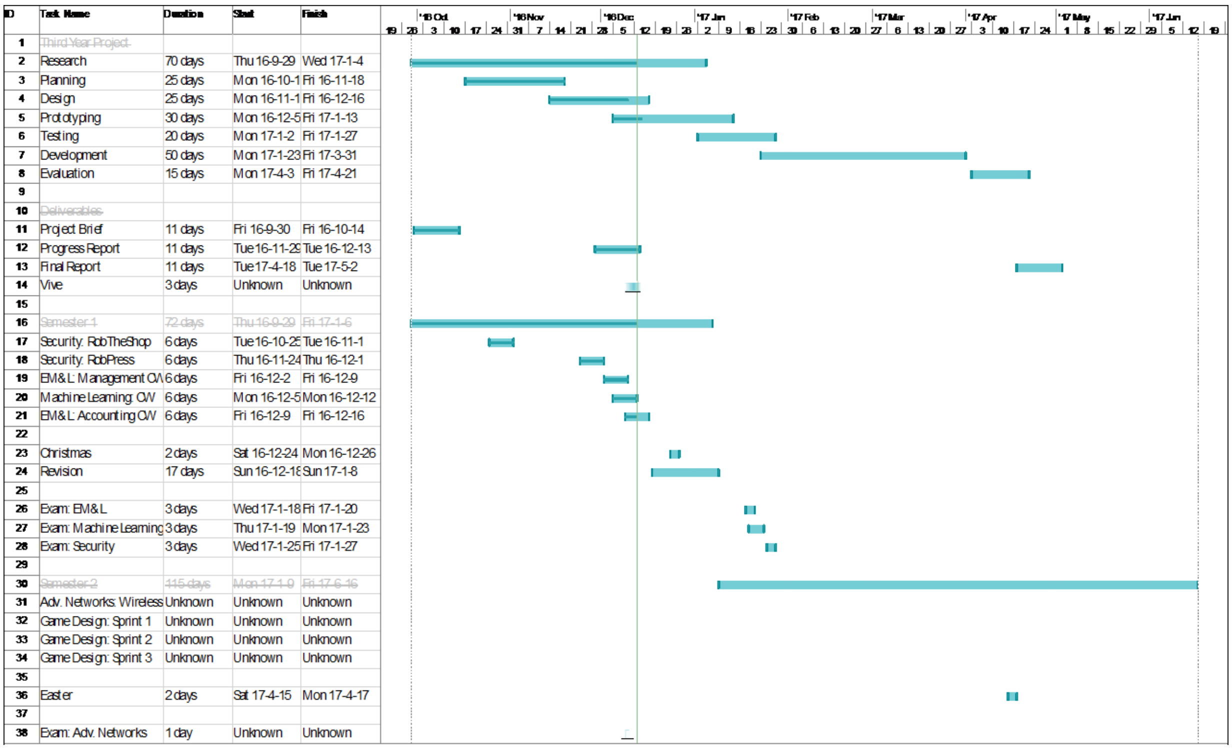
[45] Insites, “Cookie Consent - The most popular solution to the EU cookie law,” [Online]. Available: https://cookieconsent.insites.com/. [Accessed 27 4 2017].

[46] arboleya, “arboleya:electrify | Atmosphere,” [Online]. Available: https://atmospherejs.com/arboleya/electrify. [Accessed 27 4 2017].

[47] M. Consterdine, “@mattconsto,” npm, [Online]. Available: https://npmjs.com/~mattconsto. [Accessed 27 4 2017].

Appendices

Appendix A: Gantt Chart from Progress Report



ID	Task Name	Duration	Start	Finish
1	~~Third Year Project~~			
2	Research	70 days	Thu 16-9-29	Wed 17-1-4
3	Planning	25 days	Mon 16-10-1	Fri 16-11-18
4	Design	25 days	Mon 16-11-1	Fri 16-12-16
5	Prototyping	30 days	Mon 16-12-5	Fri 17-1-13
6	Testing	20 days	Mon 17-1-2	Fri 17-1-27
7	Development	50 days	Mon 17-1-23	Fri 17-3-31
8	Evaluation	15 days	Mon 17-4-3	Fri 17-4-21
9				
10	~~Deliverables~~			
11	Project Brief	11 days	Fri 16-9-30	Fri 16-10-14
12	Progress Report	11 days	Tue 16-11-29	Tue 16-12-13
13	Final Report	11 days	Tue 17-4-18	Tue 17-5-2
14	Vive	3 days	Unknown	Unknown
15				
16	~~Semester 1~~	~~72 days~~	~~Thu 16-9-29~~	~~Fri 17-1-6~~
17	Security: RobTheShop	6 days	Tue 16-10-25	Tue 16-11-1
18	Security: RobPress	6 days	Thu 16-11-24	Thu 16-12-1
19	EM&L: Management CW	6 days	Fri 16-12-2	Fri 16-12-9
20	Machine Learning: CW	6 days	Mon 16-12-5	Mon 16-12-12
21	EM&L: Accounting CW	6 days	Fri 16-12-9	Fri 16-12-16
22				
23	Christmas	2 days	Sat 16-12-24	Mon 16-12-26
24	Revision	17 days	Sun 16-12-18	Sun 17-1-8
25				
26	Exam: EM&L	3 days	Wed 17-1-18	Fri 17-1-20
27	Exam: Machine Learning	3 days	Thu 17-1-19	Mon 17-1-23
28	Exam: Security	3 days	Wed 17-1-25	Fri 17-1-27
29				
30	~~Semester 2~~	~~115 days~~	~~Mon 17-1-9~~	~~Fri 17-6-16~~
31	Adv. Networks: Wireless	Unknown	Unknown	Unknown
32	Game Design: Sprint 1	Unknown	Unknown	Unknown
33	Game Design: Sprint 2	Unknown	Unknown	Unknown
34	Game Design: Sprint 3	Unknown	Unknown	Unknown
35				
36	Easter	2 days	Sat 17-4-15	Mon 17-4-17
37				
38	Exam: Adv. Networks	1 day	Unknown	Unknown

One of the changes you may notice is that at the very start of second semester, I changed modules from Advanced Networks, to Secure Systems. This is shown on the new Gantt chart.

Appendix B: Help Content

Creating

Creating a new show is easy, simple navigate to Home and enter a desired name. If you are not redirected to an editor, this means the room already exists and you will have to try again. Simply navigate back using your web browser.

Authoring content in the editor can be done in three ways. After clicking "Create a new slide" on the left of the editor, you are given the choice of uploading, linking, or entering text. Uploading takes a file and converts it into a format your browser can understand. Linking embeds the external document into the show. Entering text allows you to type Markdown and MathML which is rendered nicely. You can also add white/blackboards.

Using drag-and-drop you can reorder the slides, and by clicking on the × you can delete them.

To gauge engagement, simply look to the left of the editor, and you can see the number of connections.

To navigate click on a slide or use the arrow keys.

Viewing

Viewing is easy, scan a QR code or enter a room name and that is it.

Quizzes

In interest of time, and to save reinventing the wheel, here are a number of recommended quiz websites:

- Pollmaker is fantastic for real-time polls
- Quizmaker creates easy to use quizzes

If you have any suggestions to be added, do not hesitate to ask.

Web-pages

Some web pages may fail to load and display a blank screen; this is because of the X-Frame-Option header which is preventing the browser from loading the sites. This is designed to stop clickjacking. Unfortunately, there is no easy way to get around this. For some sites such as YouTube, URLs are automatically rewritten to the embed page, allowing them to work.

Contact

Feel free to talk to me in person, or send an email to mc21g14@soton.ac.uk

Appendix C: Adriana Wilde Questionnaire

1. What do you like about the system?

I like that it's simple and it seems to be useable in terms of, it um, having the main functionality that you would require in the lecture. Supports many different file formats, keeps everything together by advancing everyone with you. I like that feature.

2. Could you see yourself using this system, or one like it?

We do sometimes use poll anywhere, would be keen to see how it integrates. If the polls work well, yes... it's a good thing that it integrates everything in one site. If integration works well why not.

3. What would you change, given the chance?

I would like to see it fully working. I would like live editing, to correct myself. To be able to persist, so people can revise from those notes.

4. Are there any other statistics you would like gathered?

Poll data, to be able to keep that data for later on. So, you can reflect on the learning activities. What students did with what you presented. More bidirectional. Can students post questions, reflect attendance, what else can you do with that.

5. Do you have any suggestions to increase participation and interaction?

How scalable is it, would it be able to handle hundreds of students at a time? Does it feel really like real time. Check on the scalability.

6. Anything other thoughts?

No, well done, looks good. Recommend showing to Adam Warren, a learning technologist. He's constantly evaluating these response systems. University had clickers, moving away from them. Looking to evaluate technologies. Clickers are gone, too complex. Interop problems. Do they work with everything? Why I'm pointing you towards Adam.

7. Follow Up:

I would say that, that seeing it working, it's something you could use straight away. Before I had doubts it was ready. Now you can just use it.

Appendix D: Adam Warren Questionnaire

Make a note of the QR code being too small. Some people might come to the front. A handout is a good idea although a link helps. Can you go through it at its own pace, or is it linked? PDFs can be zoomed. Not everyone has laptops, about half.

1. What do you like about the system?

I like that it works, that it, um, is fully functional which is great. The domain names a hit. The, you've certainly been able to crack the showing of a range of media types. So, in science and engineering, latex and sort is nice. I like being able to include LaTeX. People want an easy way to convert it so it can be copied and pasted. Nice clean interface, easy to use. One of the many things to adoption. If it looks difficult, getting people over the first hump is really tricky.

2. Could you see yourself using this system, or one like it?

The reason not, is that there are plenty of much more sophisticated systems currently available, which not only do presentation, do student interaction. The interaction that it enables is the key thing. You can link through to links such as quizlet. There are all in one solutions out there. You are definitely heading in the right direction. You need to mindful this is a third-year project, against developed systems with teams of developers and venture capitals. Much more mature, well developed systems

3. What would you change, given the chance?

There is an awful lot that could be added. But at the moment, in terms of feedback, how difficult would it be. One example would be a thumbs up or down system. The tutor can see on the slide can see how many people understand. That's where it changes from PowerPoint, to something where they start being able to interact.

4. Are there any features you feel are missing?

There are lots of interactive in class systems, that just allow polling to messing systems, moderation of the messages, lots of different quiz types. There are such a lot of systems. Nearpod and TopHat are turning into more of a substitute for the learning environment. In NearPod you can send people a question out that given an axis, ask you to draw an equation. As a tutorial you can mark in real time, show a super position. There is a lot that can be done.

5. Are there any other statistics you would like gathered?

It's hard to know what it could be, because it's a presentation device. So, I can image people would be interested in what devices are connecting, more of a technical question. Could be done via google analytics. Would be useful.

6. Do you have any suggestions to increase participation and interaction?

It's the one about thumbs up and thumbs down for now. One thing I noticed that there is people start from a simple system that does one job really well, suffers from feature creep. Before you know you have a complex user interface, and more choices than people need. A good example is poll everywhere which is great, but it does so much these days it's a bit daunting. Do one thing well.

7. Anything other thoughts?

Is it all just regular HTML5? Can people use a screen reader? Can people zoom in on their devices? It would be possible to talk about accessibility. What does it do, what could be done to improve it? But if it's basically HTML5. Good to see it working. Hide the navigation buttons from students. Need to suppress those.

Appendix E: Oliver Bills Questionnaire

1. What do you like about the system?

I think it's a good idea, and it would be useful for following along in class, and people who like doing interactive kind of lecturing systems as opposed to siting there and talking. It's a good foundation, but to truly be useful, I need the ability, I need the ability to input PowerPoint slide and walking through that. As well as being able to distribute exercises and files like that. Would be handy if I could put code and give them a link.

2. Could you see yourself using t
3.
4. his system, or one like it?

I could, given the points mentioned above.

5. What would you change, given the chance?

Um, again, I've kind of mentioned that. I would like to see more of the built-in interactivity. Polls and quizzes. Not needing to use a third-party service, built into the application. Would like to put up an exam question, they answer and we work on it together. The interaction side would go towards the strengths of the application. How could you best utilize the two-way link, pushing and pulling.

6. Are there any other statistics you would like gathered?

I would like to see whether or not people are idle or active. Do people use the application, or do they just use it at the start of the lecture, and let the phone turn off. How many people look at it or end up using it.

7. Do you have any suggestions to increase participation and interaction?

Add a last active thing, to measure participation

8. Anything other thoughts?

Combine it with Jackbox, so we can run games through it. Would like game based modules so we could set them up inside the user interface. Something I've tried to setup my self, but never had time for it. The user interface looks really good. Really impressed. It's something I'd expect. Minor issue with the buttons going off the screen, looks professional, more so than other third year projects I get too see.

Appendix F: Federica Paci Questionnaire

1. What do you like about the system?

I like that it's synchronized across different devices. If I understood, this is one of the features of your tool.

2. Could you see yourself using this system, or one like it?

If it adds integrated functionality. Not just creation of webpages or presentation, yes. As I see it right now, it's the same thing as google docs, or any other cloud service.

3. What would you change, given the chance?

Would integrate additional functionality. In my context, it is fine to have different content. For lecturing content, it needs social features. Needs to enable students to comment. If you share slides or coursework, you want a discussion page.

4. Are there any other statistics you would like gathered?

No, no.

5. Do you have any suggestions to increase participation and interaction?

Add in quizzes.

6. Anything other thoughts?

Nothing comes to my mind right now. It would be a good idea to have a questionnaire about the user. Would have been a good idea to do some form of statistical analysis.

Appendix G: Initial Project Brief

The Problem

Lectures, especially those spanning multiple hours, often run the risk of disengaging their audience without adequate breaks in the presentation flow, damaging the learning potential. To fix this, often lecturers either give the audience a few minutes to stand up, or better yet, break up the presentation with useful examples or quizzes to both review and motivate [1]. The problem is that often the switch between slides and resource or quiz takes many minutes away from the lecture, is unreliable, and often fails to live up to its true potential.

My project aims to fix this disconnect, and bring more interactive lectures to the halls.

My Goals

I aim to create a fully responsive, online web application that will aid the learning experience both in and out of the lecture theatre. My application will combine lecture slides, quizzes, videos, links, and more in one place, helping the lecturer and students focus on the material, and provide added learning.

In the lecture theatre, the lecturer will be able to use a standard clicker to advance inside and between any resource, both on the projector, and on viewer's devices. I chose a web application both so it can run anywhere, and so students who wish to can follow along on their devices. After the lecture if desired, the lecturer can share the presentation, allowing the learner to revise at home. Statistics may be gathered.

The learner will be able to either join a room using either a link or QR code, or by subscribing to the lecturer.

Project Scope

I have chosen to create a responsive web based application because that will allow it to run on any modern smartphone, tablet, laptop, or desktop without having to create native applications for each device and OS.

I plan to support at minimum the ability to display slides, external resources, and quizzes both on a projector and on learners' devices; I aim to create different handlers for different external resource types. To display documents and presentations, I plan to use viewerjs.org as it supports the most common document formats. To synchronize devices, I shall use WebSockets as they are the lightweight real-time communication protocol supported by all modern web browsers. To create a responsive, fun design I plan to make use of materializecss.com which takes inspiration from Google's material design standard [2].

My project has a wide scope, so even if I encounter setbacks I should be able to adjust and deliver a useful, polished application. If I get time, I will add additional features such as more quizzes, different content, and anything else that while reading, I discover to be of use to my final design.

References

1. S. Thiagarajan, Thiagi's Interactive Lectures: Power Up Your Training with Interactive Games. Alexandria, Va.: American Society for Training & Development, 2005.
2. "Introduction", Material design guidelines, 2016. [Online]. Available: https://material.google.com/. [Accessed: 14-Oct-2016].

Appendix H: Contents of Design Archive

project/

```
├── README.md
├── client
│   ├── errors
│   │   └── 404.html
│   ├── layouts
│   │   ├── default.html
│   │   ├── footer.html
│   │   ├── header.html
│   │   └── simple.html
│   ├── main.html
│   ├── main.js
│   ├── main.less
│   ├── pages
│   │   ├── about.html
│   │   ├── help.html
│   │   ├── index.html
│   │   ├── present.html
│   │   ├── revise.html
│   │   ├── slides.html
│   │   └── users.html
│   ├── resources
│   │   ├── audio.html
│   │   ├── blackboard.html
│   │   ├── pdf.html
│   │   ├── picture.html
│   │   ├── resource.html
│   │   ├── text.html
│   │   ├── title.html
│   │   ├── undefined.html
│   │   ├── video.html
│   │   ├── webpage.html
│   │   └── whiteboard.html
│   ├── scripts
│   │   ├── events.js
│   │   ├── helpers.js
│   │   ├── iframe.js
│   │   ├── present.js
│   │   ├── replacements.js
│   │   ├── revise.js
│   │   └── slides.js
│   └── styles
│       ├── cards.import.less
│       └── overrides.import.less
├── imports
│   ├── api
│   │   ├── connections.js
│   │   ├── messages.js
│   │   └── rooms.js
│   ├── blacklist.js
│   └── startup
│       └── accounts-config.js
├── lib
│   ├── routes.js
│   ├── sanitizer.js
│   └── settings.js
├── package.json
├── public
│   ├── ViewerJS
│   │   ├── compatibility.js
│   │   ├── example.local.css
│   │   ├── images
│   │   │   ├── kogmbh.png
│   │   │   ├── nlnet.png
│   │   │   ├── texture.png
│   │   │   ├── toolbarButton-download.png
│   │   │   ├── toolbarButton-fullscreen.png
│   │   │   ├── toolbarButton-menuArrows.png
│   │   │   ├── toolbarButton-pageDown.png
│   │   │   ├── toolbarButton-pageUp.png
│   │   │   ├── toolbarButton-presentation.png
│   │   │   ├── toolbarButton-zoomIn.png
│   │   │   └── toolbarButton-zoomOut.png
│   │   ├── index.html
│   │   ├── pdf.js
│   │   ├── pdf.worker.js
│   │   ├── pdfjsversion.js
│   │   ├── text_layer_builder.js
│   │   ├── ui_utils.js
│   │   └── webodf.js
│   ├── background1.jpg
│   ├── background2.jpg
│   ├── background3.jpg
│   ├── cursor.svg
│   ├── mathjax-config.js
│   ├── pdf-sample.pdf
│   └── test-card.png
├── screenshot.png
└── server
    └── main.js

15 directories, 74 files
```

docker/

```
├── Dockerfile
├── README.md
├── build.sh
├── cert.key
├── copy.sh
├── install.sh
├── nginx.config
├── run.sh
├── startup.sh
├── uploads.html
└── windows-proxy.bat

0 directories, 12 files
```