WJEC GCE Computing CG4 - Extended Project

Candidate Name: Daniel Roberts
Candidate Number: 4699
Centre Name: Shrewsbury Sixth Form College
Centre Number: 29285

Contents

Part I

Analysis of the Business

This initial part of the documentation features the analysis that was performed on the business. It includes the business' background, as well as an in-depth investigation on their current system, featuring document inspections, interviews, and observations. Also included is a problem definition, wherein the broad aims of the project are outlined; this definition also makes reference to the limitations of the solution. Finally, detailed objectives are clearly laid out, providing an overview of exactly what the solution should achieve.

1 Background

1.1 About the Business

The Priory School is a medium sized secondary school located in Shrewsbury, Shropshire. Employing over 100 teaching staff, there are approximately 900 pupils on roll, ranging in age from 11 - 16. Each belongs to an individual form group. During it's previous two inspections, Ofsted judged the school to be Outstanding, the highest possible rating. The school is a founding member of the Salop Teaching Alliance, giving them high standing in the local area. Additionally, the school has the highest attendance rate in the county, and achieves exam results well above the national average.

Like many schools, a house system is in place, which is linked to the form system. Each student belongs to one of five houses, either Acton, Baxter, Clive, Houseman, Darwin or Webb (named after famous local figures). This house is then combined with the student's year to produce their form group - a Year 7 student who belongs to Baxter is in the form 7B, a Year 8 student in Houseman belongs to 8H, and so on. Each of these forms is led by a form tutor, a member of staff who calls the register, provides support, and generally acts as the first point of contact for a student.

A change of leadership in January 2015 resulted in the previous headteacher, Ms Candy Garbett, leaving the school; Mr Michael Barratt, previously of Adams Grammar School, Newport, became the new principal. Following this change in leadership, the school has sought to embrace the advantages of technology, and has invested in several new systems, including an online homework tracker, a virtual learning environment, and a library tracking system. This newfound acceptance of technology opens the way for this project.

Previously, interaction between forms was focused on year groups: students in 7A and 7B would be encouraged to spend time with one another, but getting together within houses - students from 7A and 9A, for example -

However, due to Michael Barratt's experience in leading a grammar school, and his subsequent fondness for the more traditional house system, this horizontal approach has now been modified to a vertical one: each previous head of year is now known as a head of house; and, rather than controlling an individual year group, they now control an individual house. For example, Nick Bucknall has been placed in charge of Acton, and so controls 7A - 11A.

1.2 About the Project

Like many schools, The Priory School makes use of a form time in the afternoon. During this process, students are registered, bulletins are read out, and a timetabled activity is carried out; these activities usually include silent reading, a group debate, and quizzes. These quizzes are usually designed by the head of year, and include a range of topics, from current affairs, educational matters, and simple trivia.

Currently, these quizzes are delivered to the forms on a Microsoft Word document, via the school's LAN. The members of the form work together to arrive at what they believe to be the correct answer, and once all questions have been answered, the form tutor marks the quiz and returns the result to the head of year, usually orally. Which would aid in following the school's new policy of "togetherness".

2 Investigation of the Current System

This section details the in-depth investigation that was performed on the current quiz system used by the school. Initial contact with the school was made via an email to Mr Nick Bucknall, currently head of Year 8, on the 30th June 2015. Following a brief email exchange, the details of which can be found below, Below can be found the different aspects that were investigated.

2.1 Interviews

A number of interviews were held with staff at the school. Two form tutors were interviewed, in order to gain an insight into how they believe form times could be improved through the use of interactive quizzes, and to get an idea of how they would like such a system to work. Additionally, Nick Bucknall, a head of year at the school, was interviewed; he was chosen in order to gain information on what results the system should calculate, and how best to integrate such a system within the school. He was also chosen to get a further idea of the faults with the current system. Full transcripts for each of the interviews are included below. In addition to the formal interviews held with teaching staff, two informal meetings were held with other key members of the school community - Tim Alexander, one of the IT technicians in charge of the school's network, and Michael Barratt himself. Due to the length of these meetings and the unwillingness of the participants to be recorded, full transcripts are not available. Instead, notes on the key topics covered have been listed below.

2.1.1 Bryan Warr

DR: So what would you say is the main focus of form time?

BW: Okay. So, there's several different types of form time. One could be focusing the students on silent reading, one could be on numeracy, one could be on literacy, another could be on a discussion - on something moral or spiritual. Obviously we've got assembly time as well, but it also depends on the audience - it depends on whether they're Year 7 or whether they're Year 11. So for Year 7's it would be different to Year 11's. For example, Year 11's might spend time revising, or focusing on exam technique; but Year 7's, a lot of time is spent with Year 7's building up the relationship with their tutor. So some of that could be activities that get them all working together.

DR: I see. How important is that during the later years?

BW: Yeah, it's still important, definitely. I suppose it depends on whether your tutor is new or not - you might have a tutor who's newer to Year 10 or Year 11, and wants to spend their time getting to know them. If you've had a tutor that's taken you all the way through, like my tutor group, then you don't need those ice breakers as much - you know each other quite well.

DR: And how often does it occur that form teachers leave halfway through?

BW: Here it's not very often. It's not as often as it is in other schools - in some schools it's all the time. My last school, I took a group through Year 11, and it was unusual, because every single tutor in that year group was with them from Year 7 to Year 11, which is very rare. But here for example, our Year 10 tutor group, pretty much similar tutor team from the start to the end. There's been a little change - so tutors do change, but there's always a set activity list from the head of year, head of house I should say.

DR: So does the head of year have more control than the individual form tutors?

BW: Yes. The form tutor is told by the head of year, well next year it will be the head of house, what to do. So we get given a timetable, it's on the wall, we get given a tutor program. Do you wanna see one?

DR: Yeah, that would be helpful.

**** DR and BW move to the classroom wall, to look at the timetable ****

BW: So these change every year. This is the Year 10 one - this is the summer one.

**** Microphone fails to pick up conversation for the next 30 seconds. BW prints out tutor timetables for Year 9, Year 10 and Year 11, available below. BW and DR sit back down again. ****

BW: So you'll see that that one's key stage 3, and then you've got Year 10 and Year 11. So is you're idea maybe that you'd trial it here, or were you thinking...?

DR: Yeah. The thing we've got to do is think of an idea for a system, and then go to a business - in my case it's here - and do interviews, with people who would use the system; hand out questionnaires, and perform observations. I've done my observation because I just used my experience here, and I'm doing interviews now.

BW: So, Mr Bucknall would be a good one to talk to.

DR: I've arranged one with him.

BW: He's a head of house next year. So he's obviously, he's directly - he writes that. So the tutors, heads of house, who else? Also SLT, any members of SLT - I'm on SLT next year as well. Obviously the assemblies are run by members of SLT, but you can see there's a theme running through them - Year 9, you've got literacy, numeracy, in each one; you've got group discussions or activities, you've got silent reading in each one, because it just needs to be established. In Year 11, that would be revision, or personal study. And in Year 10 we give them that opportunity as well - we say, "right, if you want to study, you do that." It's a lot more open than in Year 7.

DR: The idea I had focuses mainly on the quiz aspect - automating it, so to speak, and making it more competitive. How do you feel about quizzes? I can see they're fairly regular on the timetable, but less so in Year 11.

BW: Well, Miss Mitchell used to do them, and she was quite good at them. The quizzes are...I find them a bit hit and miss, on my own quizzes. The Year 10's do like them, my Year 10's. If they were automated, that would work. And if it was a competition between houses...

DR: Yes, that's what I was thinking.

BW: That's what we're...that would be really...I would have thought that would be quite well received, because obviously we're moving towards bigging up the house system next year.

DR: Okay.

BW: So the house system is going to be more prevalent, so the fact that you're competing into forms...and also, heads of house would like it, because you've halved their workload, if a quiz is being laid on every week, definitely would be useful. It's obviously going to be difficult that you've got 7, 8, 9, 10 and 11...

DR: All of whom need doing.

BW: With different quizzes. That wouldn't be too difficult.

DR: No. So, when I was in Year 11, Mr Bucknall seemed to control the quiz system quite heavily.

BW: Yep.

DR: So he'd make the quiz himself.

BW: Yes, that's right.

DR: And then handed out different quizzes. How do you feel about that, if that's what Mrs Shaw does with you?

BW: Happy with that. If it was centralised, it would be even easier. So if the heads of house didn't have to handle that, if it was in a central system, that you could access, that would be easier.

DR: So say there was an icon on the desktop, that opens up the program, showing the quiz for the week.

BW: Yes. Perfect. And if you chose Year 7, with a choice of the year groups, that would be easy. So the tutor could say "right, I want the Year 7 one to come up", that would be perfect. Yeah.

DR: Because what I was thinking was the different heads of year, or houses now...if they wanted a quiz, they'd write it using the system, and then they'd have access to their form groups, so then they could target...and then they could send it that way. Would that be something they'd be interested in?

BW: Possibly, but don't forget that now they're heads of house...so Mr Bucknall, who's got Darwin, he's now got five years. So again it's the same issue, each head of house has got to make lots of quizzes.

DR: So, there aren't heads of year anymore, there are houses?

BW: Right. So we've got Acton, Baxter, Clive, Darwin, Houseman and Webb; and then each one's a house with a vertical setup to it.

DR: I see.

BW: So you're going to have the same issue with each one, in that they've got to differentiate between the years. And what they wouldn't want to do is replicate their work. So I don't think they'd be happy if they were...say Mrs Smith was doing this, and she had to do all the years, and Mrs Heath had to do the same, that would be a lot of repetition, whereas we could probably do the same...you could do the same thing for each one, couldn't you? If you did Year 7, just...bam. Year 8...right, just like that. That would be more sensible.

DR: Okay. So, with assemblies and such, are the heads of houses taking each individual house each week? It wouldn't be say Mr Bucknall who speaks to the whole of Year 11 on a Tuesday?

BW: Occasionally there will be a need for us to speak to Year 11 or Year 10 to do with certain aspects. But the majority of assemblies are with the houses, so it would be Mr Bucknall talking with Darwin - all the years. Okay? So it will be a house assembly. And we have whole school assemblies now on Mondays, first Monday of the month, which Mr Barratt does. So that's every Monday, and it's whole school, in the sports hall.

DR: Kind of like it was at the end of term before?

BW: Exactly, yeah, yeah. He's got it every month now, so we get together as a whole.

DR: Right. So how do you feel...if the school's looking to improve cooperation between houses, how would you feel about that taking place in form time?

BW: How do you mean?

DR: Well, if, say, there was the quiz, say on that side of the screen, and on the other side was a view showing how the other forms are doing, in real time, would that be something you'd be interested in?

BW: It could be interesting, yeah. Could be interesting. Definitely brings in the element of competition, live, rather...

DR: Yeah, because that makes it a lot more interesting, than finding out say a week after that Baxter has won.

BW: The only problem you've got is that they wouldn't necessarily be the same years on the same days.

DR: How do you mean?

BW: Well you'll have Acton...you're Year 7 quiz could be on the same day, but Year 7 and Year 10 might not be. Because the tutor programs are different, because they have to be, because of the assemblies and rooms, so for example the KS4 assembly there is on a Tuesday, but there it's silent reading. So they could be doing a quiz, but these lot could be in assembly. So there could be a house assembly on, when you've got Acton doing it. So, you'd have to look...

DR: But horizontally, that wouldn't be an issue.

BW: Right, horizontally, it would work, yeah. All of Year 7 would be together at the same time, unless there's a house assembly - if there's a house assembly on, it means that house would be out. So you'd have to look for where there's that literacy or numeracy activity, or something like a silent reading activity, where there's something with a group discussion. But I'm sure the heads of house wouldn't mind if, once a month, one of those...you'd have to ask Mr Bucknall of course, I'm sure he wouldn't mind...if one of those was taken up with a house quiz. Because if you did that, I'm just thinking...within a half term, you could get one quiz in for every year group. You could take one silent reading or one literacy or numeracy activity, and replace that with a house quiz, for the year group, that would work.

DR: Okay. And what sort of topics would the quizzes best fit? Because when I did them, it kind of varied each week, so sometimes it would be culture, sometimes it would be general...stuff.

BW: General knowledge is always good. Music's always good. Popular culture. Anything like that you could have. You could make it so they were subject related. So you could have history, geography or maths.

DR: Right. So how would you, as head of maths, feel about there being a maths quiz?

BW: Yeah. I'm happy with that. We're doing inter-house competitions within the subject, so we're going to do a Countdown activity. But I'd be absolutely fine with that, a maths quiz now and again, because again, it raises maths in the school. I'm sure the other heads of department are the same.

DR: Okay. Trouble is trying to make it seem like a quiz, rather than a test, which I guess it would be.

BW: Exactly. And also, you've gotta be careful with differentiation, because some students are much better at maths than others.

DR: So would the top group have done things that the lower groups would have done?

BW: Yes. Yeah, they'd have done some different things, so...maths quiz. See, my tutor group's a good example. I've got a tutor group where I've got some brilliant mathematicians, and some who really struggle, and to give them maths quizzes is always difficult, because some are at an advantage straight away - even with Countdown they're at an advantage. That would be a tricky one. General knowledge is probably better because everyone can access that. I mean, even if you did a History quiz, some people would be at an advantage. That's a tricky one. You could make it so that it would be subject related quiz, on a one off, where there was bits of Maths, bits of History. I suppose you could do that, make it more general.

DR: Yeah, cause each form has people who are good at some things...

BW: That might make it more accessible.

DR: And how well would it work? If the form had to work with one another, to work out the answer?

BW: That would be fine. Yeah, it wouldn't be an issue. The forms are used to working together.

DR: And are all forms going to be happy working that way? Certain forms and year groups, I imagine, are less likely to work well with that sort of system.

BW: Depends on the form tutor really. But the majority of the time, the form tutors accept what's asked of them. Certainly the majority of cases.

DR: And how would participation be dealt with? There'd be some people at the back of the room who don't participate.

BW: Right. You could do...to ensure participation, could be something like use the iPads. You could do something either...we do work using Socrative, using that. It's an app where everybody has to answer. So, I set the Socrative quiz up recently with my Year 7's. It's a web based one, so something like this would work. So with Socrative, I set a quiz up for the Year 7's, and you can see the live results come in. What they do is they can login to your room, and the quiz comes up as you're doing it - you get live results, with a load of reports. So I've got these results here, and these have all come through, and that's how it comes through. So there's the questions and answers - I can see that who's answered, and as they were doing it, I knew that Izzy hadn't done it. So maybe something like that would ensure participation.

DR: So this is kind of like the old Quizdom system they used in Science.

BW: Quizdom, exactly.

DR: I never thought that worked very well.

BW: No, these work a lot better - there's less faffing about. It's a web based system, so the room is accessible to anyone - it's constant; that will always be my room, whatever quiz I do. But that would work. Other than that, you're sort of relying on the teacher just...getting everybody involved.

DR: Is that something the school is looking at - raising participation - or is that not really an issue?

BW: We're looking at raising participation with the house system, and anything that does that is going to be useful. I don't think engagement in general is a problem in this school.

DR: So if this sort of system were implemented in form time - purely in form time - would that be an issue, or should it be expanded for use in subjects?

BW: I think it would be useful in form times to start with. Purely to address the problem the heads of house have. It also makes it more unilateral, doesn't it? Yeah, I think that would work.

DR: How technical would you say that you are?

BW: I'm pretty good. No, I'm not too bad. I can deal with most things, but not everybody's the same.

DR: No. I'm trying to make it as simple as possible - thinking of people like Mrs England.

BW: Yeah, some people would struggle. So it literally needs to be...two clicks. One click to get into the program, click to choose your year, maybe another to start the quiz.

DR: How would you feel about a login system, where you had to sign in?

BW: A login system would be fine. It would prevent the students hacking it.

DR: Yeah, that's what I'm thinking.

BW: Yeah, that would work. We're used to logging in to stuff. So yeah, happy with that.

DR: This sounds a little crude, but would you be tempted to help your form in any with, if there was a reward for the best performing house?

BW: Right. Are you suggesting would I cheat?

DR: Yes.

BW: Right. No, it's a good question. No, I wouldn't help them with it. I don't think they'd need my help - they work quite well together. I'm not convinced that every teacher would feel the same. ****laughs****

DR: Okay. And is that going to be different depending on their year group?

BW: Possibly, yeah. For Year 7 we'd approach it differently to Year 11.

DR: And how suitable would this be if the school had, say, a theme of the term or theme of the week?

BW: They do come up, but they would be more weekly things - so sometimes you have Anti-Bullying Week, or Global Entrepreneurship Week - so they come up now and again.

DR: I see. And would this system come in useful for the quiz aspect of that?

BW: Yeah, absolutely. And I'll tell you who else would be interested in this as well: Mrs Hancox and Miss Morris who run Life. Life would be really interested in this. I'm just thinking outside the box, purely because Friday lesson 2, when you have Life, you would have all of the year group in the same lesson at the same time - everybody in the school. So there might be an opportunity in Life as well.

DR: I'll send them an email I think, thank you.

BW: That might be worth talking about. You know...did Miss Hancox or Miss Morris teach you?

DR: Miss Morris taught me for three years, yes.

BW: Right, so you know her. So they're the heads of Life - they'd be interested in this, and that would be a good opportunity to use it, potentially. And with Life, they've got all sorts of things, so there's a lot of opportunity for quizzes.

DR: If this were rolled out, how likely is it that it were actually used?

BW: I think if people are told to use it, they'll use it.

DR: Okay. And how would the heads of house tell you?

BW: When I get this from Mrs Shaw, I follow that, because she tells me to. We know what we have to do - this on a Monday, that on a Wednesday. It might not be that we do the same thing, but we do some literacy and numeracy, then we do a group discussion or quiz, and then we have assembly. So we have that programme, and we follow it. We're actually checked up on whether we do that. SLT will check, randomly - we don't know when they're coming, but it's with the head of house, and they check what you're doing. If you're not doing what you're meant to...trouble.

DR: Right. I remember last year, I had Mr Massey. And he wasn't...

BW: *laughs* You don't have to say any more!

DR: And that was an issue sometimes.

BW: Yeah. Okay. I feel your pain. ****laughs****

DR: So, how would you feel about the quiz...

BW: *laughs*

DR: About the quiz...****laughs****...not starting until all the teachers had signed on?

BW: Yeah, happy with that. I mean, it could be that you could trial it with one year group. You could ask the heads of house which year groups they feel most confident about giving it to. It might be that you've got a team of tutors in that year group who know they kids better, who've been together longer, who are maybe more confident dealing with this stuff, before you roll it out. Don't forget, you've got Year 7 tutors coming in September, who are gonna have to get to know their tutor groups. That'll be the priority for them. Year 11, pretty much starting them on exam technique...well, with the exception of your tutor ****laughs****. You want to get the ball rolling and get them into exam mode, so maybe a Year 9 team...****laughs**** yeah.

DR: Sorry. ****laughs****

BW: I dread to think! ****laughs**** Who was in your tutor group? Name me some interesting characters.

DR: Nick Porter.

BW: Jesus. Right. Go on.

DR: Doug Coull.

BW: Oh right, not too bad.

DR: Liam Davies.

BW: *laughs* Say no more - we're there!

DR: *laughs* So how would you feel about an email in the morning, reminding you of the quiz? Obviously you've got the timetable, but not all teachers are going to follow that.

BW: *laughs* Yeah, emails would work. Most of the teachers check their email in the morning.

DR: That wouldn't be seen as too overbearing?

BW: No, I don't think so. You could also set up a reminder on Outlook - that would work. Rather than an email, we could get a reminder, at whatever time the quiz is supposed to take place, so my Outlook is linked to my calendar. I get a reminder and it would just say "Year 10 quiz - 1:40pm". That might work as well.

DR: Well, I think that's pretty much it.

BW: Okay.

DR: But thank you very much, you've been very helpful.

BW: Anytime! I would set up a discussion with the heads of house - certainly Mr Bucknall.

This interview has given information about the praticality of the system, and how it can be integrated into the school day. Warr's reaction to the general idea of the application was positive, indicating that, as an idea, it could work very nicely. This was reinforced during a seperate visit to the school; upon bumping into him in the corridor, he stated that he had mentioned the idea to another teacher, and they had also very much liked it. In particular, vital knowledge was gained about the horiziontal restructuring of the form system, allowing vital changes to the idea to be made.

2.1.2 Wendy Blower

DR: So, my project is...I've got to come up with a system for a business, in this case Priory. It's to do with form times - a sort of quiz system, an automated one. The heads of house could write a quiz, and send it out to the different house groups, for it to be answered in form. The results would be calculated, reports made, etc. That's just a basic overview - what are your thoughts?

WB: It sounds like a good idea to me. If I can just clarify exactly what it is...so, during tutor time in an afternoon, I'd be told that there's gonna be a quiz, that would all be online...

DR: Yes.

WB: So then we'd do it as a form...

DR: Yes.

WB: We'd put our results into the system...

DR: Yes.

WB: And then somebody, one of the heads of house, could log into a system and see, well, 8A got this many, 9A got this many, or whoever...Yeah. I think that sounds like a great idea.

DR: Alright, so...I'm sorry, I've got so many questions...

WB: That's alright, don't worry!

DR: So would you prefer the system to be the form working together as a whole, or with each individual student, perhaps with an iPad?

WB: Right, I can see ads and disads to both. From an efficiency point of view, I think if it was the whole form...I think there would be less mistakes that could be made. Because then it's either the member of staff at the computer, or one student with an iPad, and then everything gets done in one go. If you then had each student with an iPad - which I think the students would prefer, because it's more fun - if a student doesn't bring their iPad with them, or it's dead, or...not everybody has one so we have to get some from IT, what happens if we can't book them? I know you could book ahead, so I think I would prefer it if it were just done centrally...although maybe if their was an option to do either, that might be good, because their might be some days where I know I can book iPads, and I know everybody's got one, so I might say "right, today, you're all going to do it on your own iPad". So it's a bit like...did you ever use...

DR: Quizdom?

WB: That's the one! I was thinking Quizlet and that didn't sound right. Yeah, Quizdom - it sounds a bit like that. Is that the kind of thing you were going for?

DR: Yeah. But I probably couldn't do both.

WB: Right, okay.

DR: Because it is all quite complicated, and it does take a while to...

WB: Is there one that's easier than another? From your point of view.

DR: Yes, the one with iPads would be quite a bit harder - but it would be more rewarding.

WB: Oh, God! Okay, so if I had to go with one...who is this, who is it supposed to benefit? Is it supposed to be about engaging students?

DR: Yes. I was told by Mr Warr that the school is trying to focus on the house system.

WB: Yeah. I think if your aim - if your main aim is about engaging students, I think you'd have to do it with the iPads. Where they each have their own iPad, feeding to a system. I think us doing it, one feeding into one thing...perhaps if the aim was more about collating points in an efficient way, maybe that. So I think it's gotta do with your aims, hasn't it? Whatever your aim is, you have to pick the one that's suited more to that. Does that make sense?

DR: Yeah, absolutely. What I thought would be nice if each student gave an answer to the quiz, and then the most popular answer in that form became that form's answer, but then if that's spread across 7A, 8A, 9A and so on, the most popular answer out of those becomes that house's answer.

WB: Right, I see. But then would you need all of 7A, 8A, 9A and so on - would we all have to do the quiz at the same time?

DR: Yes - and that's what I was speaking to Mr Warr about: getting the right time. With the form timetable, he says there could be a slot each term, where that could be done.

WB: And I think that's the thing actually - with him saying each term, I think that would work, if we knew that...I don't know, on a day in October, and a day in January, and a day in say April...if we all knew, then yeah, I think it would work. But, see, we used to have a quiz on our timetable once a week. So I suppose in my head I was thinking that we'd be doing this once a week - it just wouldn't work, because things crop up, and then somebody gets called out, whereas if everybody knows that on this particular date everybody's gotta be doing it, then I think it could work really well, and I think that would be another reason why everybody should then...hold on, I've just thought of another problem...that's why everybody should then do it on their iPads, because they'd be like "oh yeah, we wanna beat them next door", because everybody would be doing it. But, if we all did it at the same time, there wouldn't be enough iPads.

DR: That's what I wasn't too sure about. Because, I'd heard the school has an iPad loaning scheme. How widespread is that?

WB: At the moment, we've got...I think it's approximately half of Year 8 have got them, and approximately half of Year 9. As far as I'm aware, I think that will happen...so this Christmas - this is how we did it last time - this Christmas, it will then be launched to current Year 7's, because they'll be Year 8 again. So ultimately, in a couple of years time, there will be at least half a year group in every year would have an iPad. So it's whether you could do it where, if it would work like this, it might be that you say "right, on Thursday, everybody in A, and everybody in B does it". So 7, 8, 9, 10, 11, A; 7, 8, 9, 10, 11, B. I mean, I'd still have to work out how many iPads that is, but it's whether you could do A and B on one day, C and D, H and W. If you could do that, we'd then have enough iPads. Could you still get it to work?

DR: I think I probably could do, yes.

WB: Right, okay. Or even if you...could you do just all of Acton on one day? If there wasn't enough iPads?

DR: See, I suppose you could do, but what I really wanted to focus on was the realtime nature...

WB: Oh right. Yeah...

DR: So you'd have a thing at the bottom saying "Webb has chosen this answer"...

WB: Right, I see.

DR: "Three quarters of Houseman have now answered" - it just makes it more interesting.

WB: Oh, okay. Did Mr Warr say anything about like...the number of iPads that we've got available?

DR: No, I haven't spoken to him about that.

WB: Right, because...is it only me that you're talking to about it, or have you got people...?

DR: No, I'm speaking to Mr Warr, you, Mrs Smith, Mr Bucknall, and the IT guys [Tim Alexander].

WB: Right, okay. So in fact...it might be the IT guys might be able to help you out a little bit more there - because they know exactly how many iPads we've got. Yeah, that's more the technical side then, isn't it? But yeah, I agree actually - if you're focusing on the real time...I mean, I've been in a school before where we did...it was a talent show, and everybody scored at the same time. I can't even remember how we did it, but everybody scored at the same time, so you know like you see it on the TV? When you look on the screen, and it's like "who's voted for person A", and all the scores start going up, and you all vote at the same time.

DR: That's kind of what I had in mind.

WB: Yeah. So I see how that would be so good, and everybody would love it, to be able to see...yeah. So the IT guys are your best bet for talking about it. I mean, how long do you think this would take to make?

DR: Well, the coursework officially has to be finished by the end of the Easter term. By that point, I should have a mostly working, fully tested system, but I'll need to do a bit of extra polish and work out a deployment strategy with the IT guys before it's ready for the school to use.

WB: Okay. So I think in terms of real life then, if we were going to use this moving forward, we're at least a year away then, aren't we? By which time we might have more iPads. I mean, I'm looking at getting rid of all my laptops, and replacing them with iPads, and that certainly can't be done by tomorrow. So it might be we've got more iPads in school by then anyway, so it might work.

DR: And bear in mind, in Mrs Smith's and Mr Edge's rooms, they can use the computers, because it would be web based.

WB: Oh, so we could use the computer rooms, like 22 and T1?

DR: Yes.

WB: And then of course we've got laptops in school. Oh, okay. Okay, yeah. You're idea is good then, that it's the real time.

DR: So, how well do you think students would take to this sort of system? I did the quizzes last year, just on paper, and I never thought it was that much fun.

WB: I've got a Year 8 tutor group, and I think they would love it. Have you thought about talking to any students?

DR: I emailed Mrs Smith a questionnaire to hand out, but I haven't heard back yet.

WB: We have got a Year 10 student you could ask in the room...

DR: That could be very helpful.

BETH, a Year 10 student, joins the interview.

WB: Beth!

BETH: Yes?

WB: I'll try and talk a bit louder so you can hear me on your ear phones. Right, just picture it. You know when you've done quizzes in form time before, and it's a piece of paper, or your teachers reading out a quiz, imagine that scenario in your form room. How does everybody feel about it? Are you bored? Do you like it? Does everybody get involved?

BETH: Not really.

WB: Do you know the reason why?

BETH: It's just quite boring. The questions aren't very good.

WB: Okay. So imagine the same questions - pretend the questions are the same - but this time, you're all voting for answers via...would it be an app?

DR: Yes.

WB: Via an app, or you're on a laptop doing it, and the whole school's doing it at the same time. So you do it in your house - who's your house?

BETH: Webb.

WB: So you'd be doing it for Webb, and real time you would be able to see on the board what everybody's answered, and who's winning?

DR: Yes.

WB: Would that make you more...want to do it more?

BETH: I think so, yeah. I've got quite a competitive form, so if we could see our results live against the other forms, that would make it a lot more interesting.

WB: Right. Is that from me saying that you're competing with everybody else, or would it help that it's real time and you're using apps and stuff?

BETH: I think it's more that we can see what everyone else is doing, and that it has a bit more importance. At the moment they just get thrown in the bin. But if it's for house points and actually matters, then that would be better.

WB: Right. So your age group...'cause I'm just saying to Dee here that my tutor group are in Year 8, and I know they'd love to do that, where they've got apps and they can answer. I know they'd love it, but of course they're a couple of years younger than you. So you think the competitive...

BETH: Yeah.

WB: It doesn't matter though that you'd be in Year 11, you'd still want to do it?

BETH: Yes, if it's competitive enough.

WB: Okay. Thank you!

BETH leaves the interview.

WB: Sorry, I feel like I'm taking over the interview! ****laughs****

DR: No, it's good! ****laughs****

WB: *laughs*

DR: Should the individual tutors be able to help make the questions, or should that be left down to the heads of house? Before, with the head of year system, this would have worked a bit more nicely, because there's one key figure per year, but...

WB: Hmmm. Good question. I don't think form tutors should be involved. I think we could be involved in...say like Beth's just been saying, they'd have like the questions, so it might be nice if, lets say...so I'm in Acton, so it would be good if in an Acton meeting, so there's like me, Mr Warr, Miss Jebb, Miss Hayman; and a new guy, who'll be starting in September; so there's the five of us and Mrs Smyth, she's head of house...it would be good if in a meeting we could say "right, can we make sure that we've got...a section on history, a section on reality TV", that kind of thing, so it's not all...like some of our quizzes, like Beth said, some of them are a bit boring. I don't know where the questions come from, whether they've just been taken of the internet, but they're a bit boring. So it'd be nice if we could have an input like that, but I wouldn't wanna have an input in the questions, because I know I wouldn't cheat ****laughs****

DR: I see ****laughs****

WB: Right! But of course, if I've got my form and they're all getting excited, it would be very hard to not guide them in the right direction, and I think it then would be unfair. Whereas if the heads of houses had come up with the questions, they could all be in their office, watching real time what's going on, and shouting at the screen! But they've got no input, and we didn't know what the questions were either, and it means we can get involved with what's going on, rather than it being them and us, kind of thing. We could get involved with them then.

DR: That could be really good.

WB: Yeah, because we're part of the house, you know, we're supposed to be joining in, with things that are going on, like, going off topic slightly, but there's a house run that's going to be happening, and we've got to do it as well. So... ****laughs*** ...I know! I'm not looking forward to that! But we can't expect them to do it if we're not going to do it. So it would be nice actually if we didn't know the questions.

DR: Okay. And how would you feel if there were some subject related questions? I spoke to Mr Warr, and he said he wouldn't mind there being maths questions in there, but it's slightly different with business, because there's less of an ability gap.

WB: I like the idea of there being subject questions...I think actually business could be alright, because all you'd have to do is make sure that there are only questions asked that you know students have studied in Years 7, 8 and 9. But then the only issue that brings up is if, if the form tutor is not supposed to know the questions, we'd have to have set them, wouldn't we? So...ooh, I don't know on that one. Be interesting to see what anybody else says. It depends what's more important - is it more important for tutors to get involved with their house, and try and work out the answers; or is it more important for us to ask questions around subjects? Like, on one hand, there's lots of questions you could ask in a quiz, but d'ya know what? History questions come up in quizzes more often than business. So there could be a history question that comes up, that is to do with something that they've learned. So unless we all had to put forward one question per department or something, and then we'd have to promise that we wouldn't give the answer out...****laughs****...'cause otherwise we'd have an unfair advantage, when the business question comes up, and I'm the business teacher. So we'd have to just say "right, I'm not gonna say anything for that".

DR: But, the subject questions could make it a bit like a test. I don't want to be rude, but business isn't the most exciting...

WB: *gasp* Dee!

DR: Well it isn't though!

WB: How dare you! But you did so well!

DR: Not...amazingly well!

WB: ****laughs**** Okay, so then I think what you could probably do is if there was a business question, I possibly then wouldn't ask a business question that was about Years 7, 8 and 9, but I could ask a question about Richard Branson. So maybe, still do something from topics, that's more real life. I'm sure...could everybody else do that? I'm sure they could. Maths: what's the size of a football pitch? So maybe.

DR: How easy should this be? The students in Year 7 aren't going to be able to cope with an overly detailed interface.

WB: Are you talking about how easy the questions should be, or how easy to use?

DR: Use.

WB: In my mind, something like this is as easy as...a question, you know I'm thinking of apps I've seen before, a question, and multiple choices? Have you thought about whether it's gonna be multiple choice?

DR: Yes, the people who make the quiz can choose between either multiple choice, multiple correct answers; or a custom input, where the students type in an answer.

WB: I think it depends on the type of question as to whether you input yourself. So if you do both, I think some should be multiple choice...but let's say there was a maths type question. If the answer is like...15, or 1974, you can't type it wrong, can you? The answer is what it is. Whereas if you're asking someone "what happened on the 7th July?", however many years ago it was, the way somebody would write "it was the day that London was bombed", or whatever, everybody would write it differently.

DR: But for that kind of question, multiple choice would be a better fit.

WB: Yeah. So if you could do both, depending on the answer whether it would be multiple choice or just an open, I think that would be good.

DR: Okay. And how quick are the iPads to get started with?

WB: They're really quick. Do you mean in terms of using it in a lesson, switch it on, start using it?

DR: Yes.

WB: They don't get switched off. So literally, when they have an iPad, and when they've got their own iPad, they're all on. That's why we prefer them really. With the computers, well, you remember using them - they're a pain in the bum. You switch them on, wait for them to connect...it's not like that with iPads. So you could literally do it straight away.

DR: Alright. And how long do you think the quiz should last?

WB: I think it should be in rounds. I think if you're doing it...if it would be something that we would do once a term, I think it would be nice if the head would say that it's going to last longer than an afternoon's form time, for example. I mean, I've just presumed it would be done in form time.

DR: Yes, that's what I was thinking.

WB: So I'm thinking that we'd extend form till, I don't know, 2:15. I think if it's made a big deal out of, I think they'd really get into it. And I think rounds really spices it up. Like, I've done quizzes before, I don't know if I ever did one with your class, but I've done quizzes before where you've got the picture round, so you'd have a picture round and it gets passed round, but then you start with Round 1, and Round 1 is whatever, and Round 2 is whatever, and then you've got the music round as well. So I think it would be nice to do it like that. So I think different rounds...obviously we don't want it too long. Four or five rounds, maybe? And if it was finished and everybody started lesson 5 by 2:15, that then gives you...'cause you'd want to give the results as well. I know it's live, but...there's still got to be a certain element of us all going "oh".

DR: Yes, I was thinking that could be done in the term assembly.

WB: Oh yeah, that's a good idea. If you showed live things going on, if you showed "well, Webb answered this and Acton answered this", you're not telling us the answers at that point? Would some people work out who'd won?

DR: That's a good point.

WB: I mean it might be, you know like they do on "Who Wants to be a Millionaire?", where you know when you ask the audience and everybody presses the buzzers, and it's like "this many people answered A, this many people answered B, C, D", whether you could do it like that. The only other thing is...I think students would want to know the answer straight away. Beth, would you wanna know the answer straight away for the quiz or would you be happy to wait until an assembly?

BETH: I'd want to know straight after I'd answered.

WB: Yeah, I think they would. Because they'd want to know whether they'd got it right. So that's the only thing. I think you might have to think about how you show the realtime and have them not figure out who's won. Unless you just did it random, on random questions.

DR: That would be a bit tricky to do.

WB: Oh, would it? Okay. See that's where I don't know the technical side. ****laughs****

DR: How would you feel about there being a timer, to ensure that all forms answer within, say, 10 seconds?

WB: I think that would be a good idea, because it keeps everybody on track, it means everybody's doing it at the same time, and then it just goes off to the next question if they haven't answered by then. But I think what you might have to do is do a tester. Perhaps if you did different rounds, maybe Round 1, is a test round, to show them how to do it, and show them what 10 seconds is. Because I think what might happen is, especially the very first time you do it, I think everybody would be sat ready, and it's all new, and they might go "what, what's that question, what" and then they're looking at you and somebody's not concentrating because their iPad's gone off. So I think you'd have to gee them up bit and give them an example and say "right, we're starting for real now" and then everyone will all ready to do it. But I think that's a good idea, otherwise you could have different people at different stages.

DR: Right. So the quiz shouldn't start until, say, 3/4 of the forms have joined?

WB: Oh, yeah, so of course you can get everybody to press a button or whatever to say that we're ready to start, like Quizdom?

DR: Yes. Well, it wouldn't even need to be a button - it would just detect that they've logged in.

WB: Yeah, I mean, if it was made a big deal, so like, you haven't experienced it, but we do whole school assemblies now, so every single person in the school goes to the sports hall, and we have an assembly. And it got timed one week, and the whole school got there and was sat down in 7 minutes.

DR: That's pretty good.

WB: Exactly. So I think if this was promoted correctly, and a couple of weeks before we were talking about it, and we were reminding our forms about it, I think we could make sure that everybody was sorted and you should get more or less 100% of people being logged in. But I see what you're saying, I mean...no, that's not gonna be an IT thing is it? Because what you don't want to do is see that there's still one form that hasn't done it yet. But if maybe the heads of house could be sat at a station, and they could see who's signed up, if they could see that 8A hadn't signed up yet they could get on the phone to me or somebody could run down and say "hurry up!". You could perhaps do it like that. That might work so at least everybody's taking part, and everybody knows they've got to take part. I mean, house things, as you've probably aware from Mr Warr, it's a big deal now. So everybody will be expected to be doing stuff, and I don't think the head would be happy if he thought that the quiz could start, and there's still five forms that haven't set up yet. So I think you've got to go where everybody's on, and then you start.

DR: And how likely is it that this sort of system would actually be used?

WB: I think it's very likely. As long as it's easy and there's no faff. If it's a case of "get your iPads out, open up the app"...we can get the IT guys to push a load of apps...if we said "right, get your apps open, we want this one open", and if it's literally and they don't...would they have to login?

DR: Yes.

WB: Right, that could be an issue with setting up in the first place. So we'd need time to set up in the first place...if they could perhaps do it with their school login?

DR: Yeah, that's what I was thinking.

WB: If they guys could sort that out, so yeah, if they could just login with their school login, then there wouldn't be an issue; there'd just be the technical things which I'll leave you and the IT guys to talk about, but I think that as long as you could just open the app, login, let's get ready to go, I don't see why people wouldn't want to use it. Because, like Beth said, it'll better than a piece of paper. Piece of paper's a bit boring - it's 2015 now. ***laughs***

DR: And how hard would it be to get this through management? I spoke to Mr Warr, and he's on SLT next year, so...

WB: Yeah.

DR: And he liked it, and I'm speaking to Mr Bucknall too, but...are they senior enough?

WB: Mr Warr, as he says, he's on SLT next year, and...he's said he'll take it to SLT, has he?

DR: No, but he seemed really interested.

WB: Right, yeah. If he's interested, he will take it to SLT. And I think it might be worth...have you arranged a meeting with Mr Barratt?

DR: No. I'm scared of him...

WB: ***laughs*** Don't need to be - he's really nice. And...I, from different conversations I've been involved with in him - see, I've been sitting on SLT this year, so I've got to know him quite well, and I think he would be really interested that you're an ex-student and you've come back, and you want to develop something for us, for one; number two, the house system. He loves the house system - we're doing the house system because of him. So I think you've got a double whammy there - it's house, plus ex-student. So I think it might be worth...if you can pluck up the courage, having a chat with him about it. Once you've done everything that you need to do, and you can say "well I've spoken to this person, spoken to this person", he'll see that you've used your initiative to find everything out. I couldn't imagine him saying no. Because actually, what have school gotta do? Have we actually gotta do anything? You've spoken to us, so we've given up some time, which is not a big deal, then you've got the IT guys who'll have to do the setting up of the iPads...

DR: And that wouldn't be too hard.

WB: Yeah. And is there anything else? Apart from the timing?

DR: It's timing really.

WB: Yeah. Apart from the timing, but I don't think that'll be a big issue. You know, we're doing a house run from 1:00 - 3:00 in an afternoon, so we're gonna have to change the timing of the school day...I can't see it being an issue to be honest - so I'd go for it.

DR: Alright. I think that's pretty much it.

WB: Okay.

DR: Thank you very much for your time.

WB: No problem Dee!

The interview with Blower proved helpful in understanding how form tutors and their students would react to the system. She provided helpful advice about the extent to which teachers would want to be involved, and, via Beth, provided invaluble access to a real student. Additionally, she helped to clarify whether or not the system would work more effectively via an iPad or other tablet, guiding the direction of the project. She also helped in choosing whether or not teachers should have free reign over which questions should be added, or if this should be left down to house captains.

2.1.3 Nick Bucknall

DR: Okay. So, for my Computing coursework, I've got to build a system, for a business. I've chosen here, because I like it here.

NB: Okay. ****sniggers****

DR: I thought a good idea for the system would be something that helps improve form times...

NB: Okay...

DR: And so I thought about a quiz system. I remember that last year quizzes were something that you did quite often.

NB: Yep.

DR: I thought that the way they were handled could have been improved.

NB: Okay. Are you aware that the structure's changing?

DR: Yes, with the heads of house.

NB: Yes. Alright. So we were horizontal - obviously in year groups, as you know - now we're going vertical, and we're in houses. We're running a weekly quiz, that's probably going to fit in... I say weekly, but it's probably going to be three times a half term. At the moment, we're asking the house captains to make the quiz, and then they're going to dish it out, and it's supposed to be relevant for years 7 - 11. They're going to dish it out, and it's going to be done, and then marked. The problem with that is trying to make it relevant to all year groups, and, if we try and do it on the computer, not all classrooms have access to computer rooms, and...but we are obviously doing the iPads thing now. So Years 8 and 9 do have iPads, so having it on a...an app would be really useful. So what's your plan?

DR: Well the idea was - and I've spoken to Mr Warr and Miss Blower about this, and they seemed to like it - part of the app would let you make the quizzes, and then the different houses would be able to find the quiz on the app, answer it, and then have the quiz marked automatically.

NB: Okay, good.

DR: It would then produce graphs and such. What are your initial thoughts on that?

NB: Yeah, sounds alright.

DR: Right. An extension of that idea, that I spoke to Miss Blower about, and she seemed really pleased with, was if there was an event, each term, where the school would get together, and each student would have an iPad, or a computer. They'd login to the system, and it would be like a live quiz. So, say the question is "how many wives did Henry VIII have?", that would come up on all the system's at once, and then each student would have their answer.

NB: So would it be on a timer?

DR: Yes.

NB: What if, just working in a school environment, where things go wrong, or somebody knocks on the door and you have to speak to somebody, what would happen if there was an interruption?

DR: How do you mean an interruption?

NB: So if you've got all of...everybody's on their iPad, let me see if I'm understanding you right. Everybody's on their iPad, they're all in different rooms, and it automatically is generated on everybody's iPad at the same time, the question? Yep. Then if you have a certain set time, is it going to be when you've put your answer in it generates the next question, or is it going to be on a timer for the whole school?

DR: Yes.

NB: Okay. So my question is...if it's on a timer, you're on question 1, and somebody knocks on your door, and says "sorry sir, so and so's just arrived in reception, can you send John down because his mum's here to pick him up", and I'm like "yeah okay, John go out", and I look down and now we're on question 4, because we've missed out question 2 and 3 because I've been distracted, or the class got distracted. Could that happen?

DR: Potentially, yes.

NB: Okay, so you'd have to really try and minimise interruptions. While the quiz was happening you'd have to almost treat it like exam conditions.

DR: Sort of. I suppose so.

NB: Over the whole school.

DR: I see where you're coming from.

NB: I'm just trying to be awkward...

DR: I'm sure.

NB: ...because when we do these things, it sounds good, but because it's a school environment, things do pop up, things do happen. Somebody passes out in the classroom, and then you have to go and fix them, and then everybody's flustered, and then you look back and then when you work on a timer, it's sometimes, it's tricky to stick to. It can be. It probably would work 8 or 9 times out of ten. I mean, I've only had two kids pass out in my classroom...

DR: That's pretty good!

NB: ...this year.

DR: Oh. Still.

NB: Yeah. That's the thing - it does happen, and that would spoil something that was done on a timer. How long start to finish would you envisage the quiz lasting?

DR: Well, I thought about 15 minutes. But...

NB: That's quite a long time.

DR: Yeah. But Miss Blower wanted half an hour, split up into rounds.

NB: Right. See the rounds...the rounds thing would work if you could have a pause and a break. But then everybody would start off at the same time again?

DR: Yes.

NB: Okay. So you could get your kind of...anything that...somebody knocked on your door you could be like "need two minutes". Then you'd have a breather time. Yeah, I think breaking it up would be more manageable for a school environment, just in case something did happen.

DR: So, if say the most popular answer in 8A was 4 wives, then that would become 8A's answer...

****NB gives out an appreciative "ohhhh" indicating that he has finally grasped the concept.****

DR: ...and if that's then spread through the house, so say 8A, 9A and 10A chose 4 wives, then that would become Acton's answer.

NB: Okay. I like that. What if they all choose something different?

DR: I'd have to think about that.

NB: Okay. Could there be a Year 7 winner, a Year 8 winner, a Year 9 winner? In the house? So you're running this. It's whole school. The house within Year 7 that gets the best answer, the most correct answers, rather than it all becoming Acton's answer, because the chances are, you're gonna get a lot...because there's only five year groups, chances are you're going to get probably an even split in there somewhere. Five is better than six, but still, you're probably gonna get some answers that are all the same.

DR: But if the questions were targeted in such a way that that wouldn't happen? The Henry VIII isn't a very good example, because everybody knows that.

NB: Okay. So you'd more like...what's the...like guessing population or something, that you'd...is it all going to be multiple choice?

DR: That would be for the people making the quiz to decide upon.

NB: Okay. I think it would probably work better if you didn't give the whole of Acton an answer holistically. If you tallied it instead, so that if...7A chose four wives, and then 8A chose two wives, and then they were wrong, they don't get any points for being wrong. The ones who are right get like one point per correct answer, so then Acton...three of the year groups got it right in Acton, so Acton got three points for that question, and I think that would probably be easier to tally up if you could do it that way. So Acton got three points for that question because three of the forms got it right, and two of the forms didn't get it right, two of the year groups didn't get it right, in Acton. Could that be...?

DR: That would work, absolutely. That would be a better way of handling it.

NB: Okay. You alright miss?

****SIAN JOAH, an art teacher at the school, enters the room. SIAN and NB briefly exchange words****

NB: And then that tally, that would just be totted up and worked out at the end.

DR: That would work, yeah. I was told that the school is really aiming to emphasise the house system. So each point using your system could be converted into a house point.

NB: It could, yeah. So you could tot it up, yeah. And that would probably fit in quite nicely with the value of a house credit. One form all gets one question, gets it correct, that would...and then, because there's going to be lots of questions...What you don't want to do is devalue the house credit, so you end up giving 300 house credits out for a quiz that lasted 10 minutes. But if throughout the year group, for each question, if the whole school, everybody is taking part and there's three house credits, maybe four, awarded per house, throughout the entire school, for each question, that could work out fine.

DR: And how would you feel about a real time element? Say an area on the side of the screen that says "most of Baxter have chosen this answer", "half of Clive have chosen this one".

NB: What, real time kind of...fluctuations in...hmm. Leading them to make different decisions?

DR: Yeah. Almost tactical.

NB: Yeah. Yeah, I like that. Yeah. Could you have the option of turning it off and on?

DR: Yeah, I could put that in.

NB: Yeah? Good. Yeah, that'd be wicked. Yeah.

DR: Cool.

NB: So you could have like a...little bar chart?

DR: That would work, yeah.

NB: *giddy laugh* That'd work, that'd be wicked! Yeah. Could you have it like a...answer A, answer B, answer C, and it would be going all...woooh.

DR: Yeah, that would be good.

NB: 50 people are here, 5 people are here, 5 people are there, but for the whole school rather than per house maybe? So that was what everybody in the school was...or do you think it would be better to do it per house? Could you do a house one?

DR: Yeah, I think so. I think doing everybody in the school on one screen would be a bit, so I...

NB: So you could have Acton...imagine it on its side or something, so this is the Acton, Year 11, Year 10, Year 9, Year 8, Year 7...no, don't do that. So just Year 11, sorry Acton, and then answer A, B, C and D, and then how many people in Acton have gone for that in real time. And then for Baxter and Clive and Darwin. So you could just see these bar charts going up.

DR: Or you could use dots and colours. So say each person or form in that house is a dot, then when they've picked, their dot would turn a certain colour, depending on their answer.

NB: Okay. That would be a heck of a lot of dots!

DR: Yeah.

NB: On a graphic the size of an iPad.

DR: I guess that's an issue.

NB: *laughs*

DR: You could make them small dots.

NB: Yeah! Okay, so that, like a...pull out panel?

DR: Yeah, that's a good idea.

NB: So the kids have got their iPad in front of them, and they've got the question, and then they can just "phwish", pull out the panel and then you can see what's happening as there's like 10 seconds left, you'd be like "Oooh, uuuhh, I need to pull out the panel to..."maybe you could give them half a point if they get the answer after looking at the panel.

DR: That could work.

NB: Could you do that?

DR: Yeah, definitely.

NB: Man, that'd be wicked. So this would be like a...you know, ask the audience sort of thing, isn't it? Pull the panel out, "right yeah, that's what I was gonna go for", go for B because everyone else has gone for B.

DR: But lose half a point for doing it.

NB: Exactly. Okay. That would be great. All of that, great. But definitely try and build in some breaks, just in case.

DR: Okay. And how much control would you want? Miss Blower envisioned it as all the heads of year - house, sorry - in their office, watching things on your iPad. So would you like to...

NB: Have an overview?

DR: Yes.

NB: Definitely. How easy would it be to set up the questions?

DR: That's what I was coming on to next. It would be like a...form building interface. I've not worked it all out yet - I'm still in the planning stages. But there'd be an "add question" button. You'd press the button, and be given an input to type the question, and then the question type. Then you'd enter the possible answers, and mark one as correct. Then you'd press the button again to add another question, or add a break, or a section, or finish the quiz.

NB: Yeah. Because we've used Quizdom before, haven't we?

DR: Yeah.

NB: But it was...it's never really come that much out of the science department. Because everybody's got the iPads, we could roll it out in form time...it'd be really good. It's just getting everybody to have an iPad. Year 7 haven't, and next years Year 7 won't, only Year 8, Year 9 and Year 10 will have. But then, there are iPads that we can book. There's lots of iPads we can book, but not enough for the whole year group. It might be...red, green, blue, white...red, green blue, so that's 45, and we've got white, then brown, that's another 30; so we've got 75 communal iPads. Which could do almost a whole year group, one between two. So we could have Year 11s with those, Year 10 with their own, but not everybody's bought into the scheme. It's just getting the hardware out that's gonna be the problem.

DR: Yeah. This wouldn't be done until September 2016, so the situation might have improved by then. And in classrooms with computers, like 16 and 17, they could just use the computers, because it will be web based.

NB: Aaah.

DR: But I'll package it up as an app for the iPads so it can be pushed for those who have them, and the rest can just access it via the web.

NB: Yeah. Yeah, that'd be much better.

DR: How many computers are there in the school?

NB: If we went with September 2016, in theory, you'd have Year 11 with their iPads, Year 10 with their iPads, Year 9 and Year 8 with their iPads. And then communal iPads, would fill in the rest of the gaps for the Year 7s, and computer rooms, that would probably be doable then.

DR: Right. That's good.

NB: Probably.

DR: And how easy would it be to...

NB: What about making it...it's cross platform, web based, so if they had a smartphone they could just do it on their smartphone?

DR: Yeah.

NB: Right. Perfect. Yeah, I don't see a problem with that then at all.

DR: And how easy is it going to be to integrate into the current form time system?

NB: Well, we've got a quiz on the form time system at the moment, and like I said we're getting the Year 11 house captains to write it. If we do a quick trial next year, of this, and we're happy with it, and the teachers are au fait with the technology, then we would just, if it were good to go, and everybody was happy with it, we'd just roll it out instead of printed pieces of paper, we'd just do that.

DR: Would you and the other heads of house want to write the questions, or would that be better left to the house captains?

NB: Probably the house captains. Just because they're...we need to give them something to do! ****laughs**** No, it's probably better with the house captains, because they're going to have a better knowledge of what questions the kids are going to be more interested in. If we were gonna...I know Mr Warr is writing a program for a house day, and something like this would fit really well for that. Everybody's walking round, and they could do a set task with everybody at the same time. For the house day, this would work really well. He would write the questions for that, probably. So it could be used for both: the house day, where everybody is off timetable and doing house competitions, and form times. It could be both.

DR: How simple should the whole thing be? I don't want to make it so simple that you can't do anything, but I'm worried that if it's too complicated then people won't be able to work out what to do. What's the general standard of technology members amongst the staff?

NB: Well, because everybody has had to use iPads this year - every teacher knows how to use an iPad, and they're familiar with the interface, and has had to use it for teaching this year - everybody is clued up on iPads. But if the questions were already written, and the teachers were told "right, it's quiz day today - get yourself logged in, we're starting the quiz at 5 past", all they've got to do is make sure the kids are all on the right app. Isn't that right?

DR: Yeah, that's it. But would all the form tutors necessarily do that? I remember Mr Massey was never the most...

NB: Hang on. Just thinking. There isn't...a day...I don't think...there's a day without an assembly. Because this needs to be whole school, at the same time. You can't have a whole house in an assembly, or this wouldn't work. Or a year group in a key stage assembly, or two. This is only going to work if there's a day of a week where the whole school is not in assembly. Because we've got two assembly days, key stage 3, key stage 4, and then we've got Acton and Baxter, Clive and Darwin, Houseman and Webb, and that's our five days. So if we were going to have to...if we were going to put it in as a whole school competitive against each other, it wouldn't work on the current assembly timetable. It would work if you did it in a house, but that kind of defeats the object. Well, I mean it could still work.

****SIAN enters the room, and converses with NB for a number of minutes about a child's attendance****

NB: Sorry. So we've got Monday, Year 7 and 8 in an assembly, so you can't do it then. Tuesday, Year 9, 10 and 11 in an assembly; Wednesday Acton and Baxter are in an assembly, Thursday and Friday are Clive and Darwin; Houseman and Webb. So whole school, it's not going to work in form time.

DR: Right. Miss Blower said that if a big enough deal was made, then some change could be made to the structure.

NB: Yeah. I can see that. I can see if we made enough noise to say that we liked it, and we wanted to do it, it would work. But...I can't really see how. Because they want...they definitely want the house assemblies, which means we've only got two halls. So the house assemblies have to take three days. And the key stage assemblies are done by the same person...it would mean that the key stage assemblies would have to be done on the same day, and then we could have the quiz day on the other day.

DR: It wouldn't be every week. That would get a bit tedious. No, tedious is the wrong word, but it would...dilute it. It would work better, I think, if it were once a term. Because you've got the house run coming up, which is disrupting the schedule, so could that sort of thing be done again, for the quiz?

NB: Yep. Yeah, we could drop a key stage assembly, or assemblies, for a week, and program in a whole school quiz. Yeah, we could do that. I think. I think they would buy into that more.

DR: Who's "they"?

NB: SLT.

DR: Gosh.

NB: They're the ones that are gonna...they would write the policy on assemblies. Mrs Massey would be the one that would say yes or no to...she writes the calendar you see, so she has to write the assembly rota, and things like that. But, like you said, the run, Mr Warr's idea of doing a house day - all lessons are cancelled for the whole school, for the whole day. The whole thinking...with Mrs (sic) Garbett [the previous headteacher], it was "do not disrupt lessons, at any cost - especially for Year 11", but now it's kind of flipped. Mr Barratt is much more in favour of team building and whole school activities at the expense of a couple of lessons.

DR: Alright. How easy would it be to get this sort of thing passed? Miss Blower said that I should set up a meeting with Mr Barratt. Do you think that would be a good idea?

NB: *An extremely long silence* Yeah.

DR: Yeah?

NB: Yeah.

DR: I'm seeing the IT guys next week to talk about the network and other technical details.

NB: What's the cost of this?

DR: It shouldn't...in terms of money, it shouldn't cost the school anything. It might cost the IT guys a bit of time setting it up, but...

NB: Oh, alright.

DR: In terms of money, it shouldn't cost anything. If it did, the cost would be on running the servers, which would be an absolute maximum of £10 a year. So really not a lot.

NB: Sounds great Dee, to be honest. I think...I really do. I think it would be really useful. I think it's right up Mr Barratt's street. I think it's exactly the sort of thing he's looking for at the moment. It's another thing that's whole school. I love the idea of the...being able to track other people's answers in realtime. If you could get that going as a graphic, that would be brilliant. Yeah. Pitch it to Mr Barratt, I think it'd go down really well. I mean, me and the heads of house - I can speak for them I suppose - they would be more than happy to do it. Definitely.

DR: Thanks. Should each student have their own login, or would that complicate it a bit?

NB: Would you just have a form login instead?

DR: Yeah, that would probably be better.

NB: I think so. Because you're going on a time...the whole point of this is that it's on a time limit. And, you know, the good thing about that time limit is that Mr Massey knows that he's going to be accountable. He knows that at five past this has got to happen, because it is going to happen. No matter whether he moans and groans, and refuses to press the button, it's going to happen. That's what I like about it. I know in a school environment you're thinking that it might not work because somebody's going to knock on the door and then it ruins it, but...it is going to pressure people into doing their job, which I quite like. ****laughs**** Which'll be good. That would be good. And I think it would, like you said, if you do it once a term, and it's very well publicised, and everybody knows about it, and it starts at 5 past, and you know you've got to have all of the stuff ready to go because it is going to start, and question one is going to be followed by question two, no matter what you do. Yeah, I think that would really work.

DR: Originally I thought that it wouldn't start until, say, 75% of forms had joined, to pressure the forms to get going, but you think it would be better if it just started anyway?

NB: Yeah. Or, can you have...yeah. I think it would just be better to start it. In fairness, you've got a form time, you've got to do the register. The register is the most important thing in a form time; it's a safeguarding issue. That has to be done. People coming in are late if they're two minutes after the bell, and of course there are people who are late. Then the register's taken. So at five past the start of reg, you're kind of eating, and you're...because afternoon reg is only 20 minutes. If you wanted to squeeze it into that 20 minutes, you'd be looking at making the quiz 15 minutes maximum. You've got to have time to get that register done. But if everybody knows about it, and they're there, and the form tutor says, "right, get here early, because we've got to get this quiz started, blah blah blah". Would the questions close? Question 1, when it goes on to question 2, can you go back?

DR: No.

NB: So if somebody's late...that's quite nice actually. Because they're late, they miss the question and they get no marks.

DR: Yes.

NB: Or, rather than...what would be good, if you could set it up, for...if somebody misses the question, or doesn't answer the question, it would need to default for the wrong answer.

DR: Oh, that's clever.

NB: Because if only three people in 7A answered correctly, and everybody else didn't know the answer, and they didn't answer, then 7A's majority answer for just the three people that knew, would be the correct answer, and that would feed forward, and they would get one point. So you would need to default everybody who doesn't answer a question to be the wrong answer. Otherwise you could just have the form tutor saying "right, nobody do anything, you just do it because you're clever." Do you see what I'm saying? And if a kid comes in late, or if half the form don't turn up, then. It would only count the people that have logged in, though? You wouldn't say, "7A have got 28 logins available" would you? Because chances are somebody wouldn't be in, and it wouldn't be fair to give them the wrong answers automatically just because they're not in school. So if somebody missed their login, they just wouldn't take part in the quiz would they? That'd be a nice encouragement for them, to stop them being late. Could be. Could be the other way round. Might be late on purpose. I don't think that would be it. I'd think that would be a very small minority of people that would do that. Yeah.

DR: And when should the results be known? The issue with the live thing is that people would be able to work out who's won and who's lost.

NB: I would say straight away.

DR: Straight away?

NB: Yeah. Finish the quiz, bang. Everybody knows straight away. It's there on their screen, Houseman have won, Baxter came second. Something like that. If you could do it straight away, that would be better. Doesn't matter the running total overall for the house event, but just for that on that day, to give them that immediate response, and they know that they're going to get that immediate response, is going to give them more incentive to do as well as possible.

DR: And would you like to view all of the quizzes and the results when they're finished? I only ask because I'd have to build a separate database.

NB: I don't think we'd need to really. Because if there's...there's not student logins, just form tutor logins, so you're never going to need...you're not going to use the data to track individual students, just forms.

DR: So it's just the form data that's being stored, saying that this form won the last quiz, came second on the other one? So that on the results screen, you can show that data from previous quizzes.

NB: Yes.

DR: So the questions wouldn't actually need to be stored?

NB: No. No, I don't think so. Just that bit of data.

DR: Okay. And how long would the gap between the questions being made and the quiz taking place be? They obviously can't be on the same date.

NB: No. No, sure. Don't know really. Couple of weeks?

DR: Right. So...

NB: Because you'd need to store them on the website I suppose?

DR: Yeah, the data will probably be stored on the school's servers, or somewhere in the cloud.

NB: Right, so we can do it as advanced as we want then?

DR: Yes.

NB: As long as it's more than a few days. That would work. Yeah.

DR: I spoke about with some questions there being an input box, where students can type their answer. Is that going to be a problem with some children swearing?

NB: Could be. It's not going to go anywhere though, is it?

DR: No.

NB: So it's only if they type the right answer...How would that work? So if you've got, if you're...

DR: Ah! That complicates things with the live thing actually.

NB: Ooh, it does come to think of it. Yeah. I'd probably just stick to multiple choice.

DR: Yeah. That would be a lot easier.

NB: Yeah. I would. Stick to multiple choice. If you start...somebody' going to...or a clever something is going to have to mark an inputted answer, haven't they?

DR: Yes. Well, marking would be done automatically.

NB: On a multiple choice.

DR: Yeah.

NB: But if they were going to have to input, somebody would have to look at that, wouldn't they? Because if they've put the right answer, but with a spelling mistake, the computer's not going to...Yeah, I'd stick to multiple choice. Yeah.

DR: Alright. I think that's it actually.

NB: Okay.

DR: Thank you.

NB: No problem. Sounds good. Really good. When do you think you're going to have a beta version ready?

DR: Well, I don't know. I'll start developing it probably September, and I should have a fairly decent version done by Easter. It might need a bit more polish to be fully ready for use though, and of course I'll need to set it up with the IT guys. I think a realistic implementation date would be September 2016.

NB: Okay. For the trial? Or for the rolling out?

DR: Rolling out.

NB: Would you like us to trial it? We could.

DR: Yeah, that would be good. Testing is going to be a bit of an issue on my end, because I don't have access to 900 iPads. And I can't fully simulate the school environment. Most of it will be fully tested by me, but you'll probably want to do a trial before committing to it.

NB: Alright.

DR: Right.

NB: I would set up a meeting with Mr Barratt. Once you've got it completely clear on what it...don't go in there asking him questions. Have it completely clear about what you want. What is is you're going to be doing. I wouldn't go in there and say "would you prefer this or this". I would say "I've got this, this and this". Obviously if he says "can you do this?" then be open to that, but don't go in wanting a discussion. Go in with a pitch - try and sell it to him, and then remind him it's free. ****laughs****

DR: Okay.

NB: Alright?

DR: Alright.

NB: Okay. Excellent.

DR: Thank you very much.

NB: Yeah, no problem. I'll show you out.

The interview with Bucknall proved the most helpful out of all interviews. He helped solidfiy the idea that the system should work on a timer that cannot be stopeed by anyone - questions should just keep moving, no matter how many people are connected to the quiz, in order that the more rebellious teachers cannot try to stop the quiz. His reaction to the live aspect was also deeply positive, and he helped come up with some initial design thoughts around this area.

2.1.4 Tim Alexander

Alexander was questioned over the strength of the school's Wi-Fi network, and it's ability to cope with a very large number of simultaneous connections, all sending and receiving data. He replied that the network should easily be able to cope with such a high demand, as the school had recently upgraded its network infrastructure to a quality comparable to that found in the enterprise. The network interface was brought up on a nearby screen, displaying the usual number of connections and how this affects the network. A forecast feature was also shown, indicating that the network will be able to cope with up to 5000 simultaneous connections at once before an upgrade is needed. Alexander also stated that almost all of the school site - including every classroom and even a decent way across the football pitch - has access to the network.

Secondly, a discussion was held over the school's iPad policy, and the feasibility of pushing an application to all of the iPads. Alexander stated that, though in most years the majority of students are in possession and make use of an iPad, most of these are their own, and so cannot be accessed by the school. Through his knowledge of deployment methods, Alexander revealed that, even if the school was able to push the application to all of the students, the application would have to published in Apple's App Store. This would cost a fee of £99 a year, which Alexander stated the school would be unwilling to pay, and would also result in the application having to be listed publicly; this would not be a good idea, for obvious reasons. Following this revelation, Alexander advised that the wisest method would be to develop the system purely as a web application, and not worry on packaging it up for distribution.

Following this, Alexander was asked about the feasibility of deploying the system source itself directly on to the school system. Many words were spoken on this topic, and the result can be summarised in one (admittedly rather long) sentence: though technically possible, it would be rather difficult to setup; the system in place at the school is not 100% compatible with the methods proposed to build the system, and this could impact the quality; additionally, it could represent a security risk if the application were placed too near to the school's student database; though this could be mitigated by using a virtual machine, it would result in some licensing issues; to summarise, the entire endeavour would almost certainly fail. Instead Alexander recommended that the system be deployed on a hosted cloud provider such as Amazon EC2 or Heroku. Though this would cost money, the school would be happy to pay for this; a visit to a man named Duncan, ostensibly the school's finance manager, confirmed this.

2.1.5 Michael Barratt

Coming soon.

2.2 Observations

Having been a member of the school community for over five years, I am well placed to provide an observation on how the school currently goes about creating, setting and analysing quizzes used in form times. Currently, no formal system is in place; an ad-hoc system is used, following this general pattern:

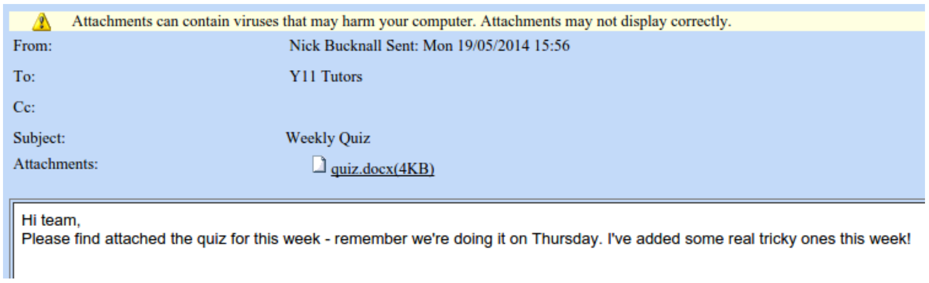
1. The head of year creating the quiz thinks of a set of questions and possible answers, usually following a theme, and then writes them down on a Microsoft Word document. The correct answer is marked out, to aid the form tutor in marking the quiz. This document is then saved to a drive on the school's LAN.
2. The head of year then notifies the individual form tutors of the quiz, usually at one of their weekly meetings, and tells them to conduct the quiz with their form group on a certain date.
3. When the date is reached, the form tutor opens the document from the network, ensuring that the document is kept hidden to avoid members of the form viewing the correct answers.
4. The form tutor reads each question out in turn, and the members of the form work together to attempt to work out the answer. They either come up with their own answer or choose from a list of options, depending on whether or not the question is multiple choice. The form tutor marks down the answer they chose, and this process repeats until the quiz is completed.
5. Once all the questions have been answered, the form tutor adds up the total number of marks achieved by the form.
6. The form tutor then passes the mark onto the head of year, either by email or when passing them in the corridor or the staffroom. This task is sometimes performed by a member of the form themselves, occasionally with the expectation that the mark achieved will be exaggerated somewhat.
7. After receiving all the results, the head of year works out which form achieved the highest result, and which the lowest. This result is reported

back to the year in the weekly assembly, often with a small reward for the highest achieving form.

2.3 Document Inspections

A number of documents were collected from the school, relating to both how the quizzes are currently set, and the management structure of the school; these will aid in creating the best possible solution that fits the needs of the school.

2.3.1 Weekly Email



Attachments can contain viruses that may harm your computer. Attachments may not display correctly.

From: Nick Bucknall Sent: Mon 19/05/2014 15:56
To: Y11 Tutors
Cc:
Subject: Weekly Quiz
Attachments: quiz.docx(4KB)

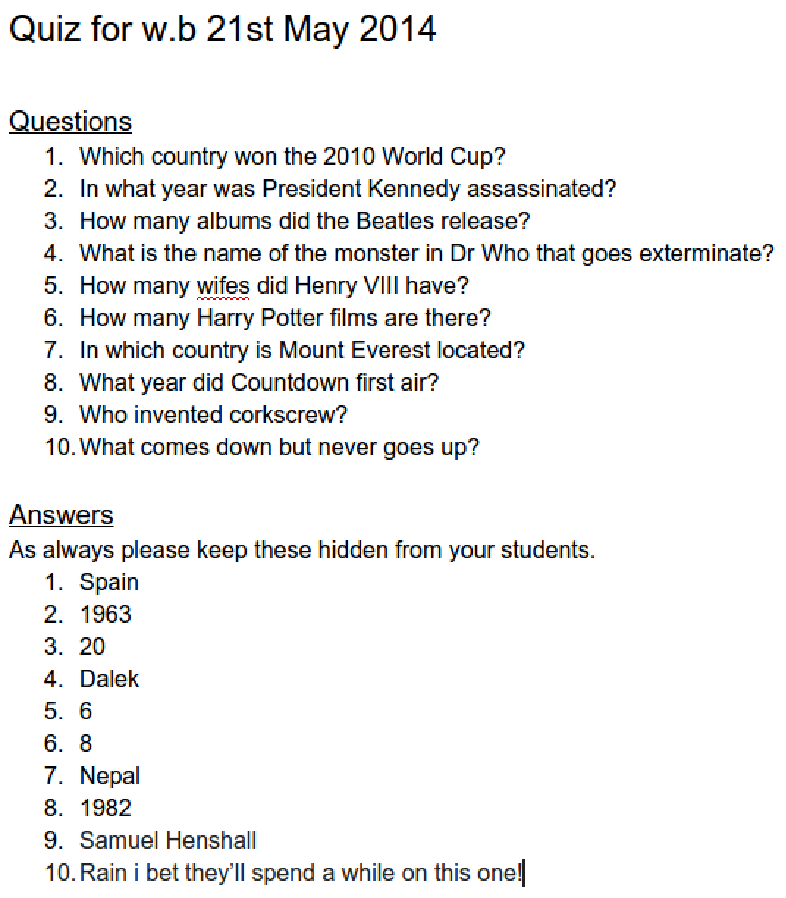
Hi team,
Please find attached the quiz for this week - remember we're doing it on Thursday. I've added some real tricky ones this week!

Figure 1: Email sent by Nick Bucknall on the 19th May 2014

This capture of an email sent by Nick Bucknall (a head of year, soon to be head of house) to his tutor team demonstrates the inefficiency with which the quiz system is currently handled. Sending an email with an attachment once a week is an approach fraught with issues, the most glaring of which is the shocking lack of organisation. As can be gleaned from the shot, the school's email system is not the most modern, and this can make it difficult to find particular emails - such as the weekly quiz. A far better solution would be to have the correct quiz appear automatically.

This document was chosen in order to demonstrate how unorganised the current system is, and how it can benefit from the introduction of a more centralised system where quizzes can just be chosen from a list.

2.3.2 Weekly Quiz


Quiz for w.b 21st May 2014

Questions

1. Which country won the 2010 World Cup?
2. In what year was President Kennedy assassinated?
3. How many albums did the Beatles release?
4. What is the name of the monster in Dr Who that goes exterminate?
5. How many wifes did Henry VIII have?
6. How many Harry Potter films are there?
7. In which country is Mount Everest located?
8. What year did Countdown first air?
9. Who invented corkscrew?
10. What comes down but never goes up?

Answers

As always please keep these hidden from your students.

1. Spain
2. 1963
3. 20
4. Dalek
5. 6
6. 8
7. Nepal
8. 1982
9. Samuel Henshall
10. Rain i bet they'll spend a while on this one!

Figure 2: Quiz created by Nick Bucknall on the 14th May 2014

The above capture of a quiz created by Nick Bucknall, and distributed to form tutors, demonstrates the form that that the quizzes currently take. As can be discerned, though the current method undoubtedly works, the existence of the answers so close to the questions causes a number of difficulties; most notably, it means that the questions must be read out by the form tutor, and the actual question sheet cannot be shown. Additionally, it no doubt takes time for Nick Bucknall to lay out the document in such a manner each week; a better solution would be if a standard template could be used (though not, it should be noted, in the form of another word processed document).

This document was chosen in order to show how unappealing the actual quiz is at the moment, to demonstrate how easy it would be for students to view the answers (and for teachers to inadvertantly show them), and how it could be much improved.

2.4 Similar Systems

There are a number of systems available, both free and at a cost, that would allow the school to improve their current method of quiztribution (*quiz distribution*). Several popular options are outlined below.

2.4.1 Quiz Creation Websites

A number of websites exist that allow users to design, play and share their own quizzes. These websites, including *QuizWorks*, *ExamTime* and *QuizBean* generally follow the same pattern: the user creates an account, is directed to an interface wherein they can design a quiz, and is then given a link with which they can share the quiz with others. For basic quiz creation, these websites are free, though for more advanced usage (*QuizWorks* defines an “advanced” quiz as one containing more than 15 questions), paid plans are available.

As these systems are websites, they can be accessed from practically any computer or mobile device, as long as there is an internet connection in range. This means that users can continue to work on their quizzes, whether designing or answering them, outside of their place of work.

Though these systems are undoubtedly useful, and could, with a few compromises, be easily integrated into the school’s routines, they lack an awareness of the structure of a school. There is no concept of “form groups” or “heads of year”, both are which are vital concepts if the system is to meet what the school desires. Additionally, they lack the ability to display a detailed analysis of the results (at least, not without paying a somewhat exorbitant fee - £60 per month in the case of *QuizWorks*), a side effect of their focus on individuals as opposed to groups.

2.4.2 Quiz Creation Software Packages

Similar to quiz creation websites, quiz creation software packages allow the user to design and play a quiz. However, these systems are desktop applications (the majority are designed for Microsoft Windows), and so can only be accessed from a single desktop or laptop system. Examples of these systems include *Wondershare Quiz Creator*, *Tanida QuizBuilder*, and *Articulate Storyline 2*. Unlike the mostly free websites, these software packages are often very expensive: the three systems mentioned range in price from $99 - $1846 for a single license, with additional licenses costing even more.

To compensate for the high prices, these desktop applications contain a vast feature set. Quizzes of every imaginable type can be created, from drag-and-drop, multiple choice, word bank quizzes, and many more. Images can be included, points assigned, and complex animations can be set to make the quiz as visually appealing as possible. In addition, reports can be generated with tremendous amounts of data, showcasing practically every data point imaginable.

Useful though these features are, they are a touch overkill for what the school's purposes. The systems are not the easiest things to use in the world, something that, considering the teacher's relative lack of IT skills, is quite a drawback. Additionally, the high costs make the systems prohibitively expensive, considering the school's status as the worst funded school in the county.

2.5 Justification of Methods

As is custom in such situations, a range of investigatory methods have been used to gain a deep insight into the current methods used by the school in their management of quizzes. By using a range of methods, including interviews, document inspections and observations, it is possible to uncover a wide range of information relating to the topic, all of which can be used to develop a suitable range of objectives that result in a system that meets the school's requirements.

2.5.1 Interviews

The most useful method of investigation used was interviews. Specific members of staff were chosen for interview, ranging from normal teachers, to more senior figures, to the headmaster himself. By doing this, it was possible to gain an appreciation for how staff would like an automated quiz system to work, and how it could fit into the school structure. This has allowed a set of objectives to be created that fully meet the school's demands. Additionally, the chance to interview a student provided additional information on what would make the system suitable. Through these interviews, therefore, all users of the system were covered. Interviews are particularly suited to this sort of task because they allow for a real conversation to be developed (as can be seen in the above transcripts). The interviewee can be asked to expand on interesting points, ensuring that as much information as possible is gleaned from the session. Of course, some members of staff would be uncomfortable with such a direct meeting (some use teaching styles that are best described as indirect), but all of the staff chosen for these interviews were perfectly willing to speak, and a number of spoke candidly and honestly about their opinions regarding the management of the school and how this could impact the success of the project.

2.5.2 Questionnaires

Though questionnaires are very useful and can provide a wide range of data, it was felt that, for this investigation, they would not be appropriate. Part of this comes down to the fact that other methods investigation are enough: interviews, document inspections and observations all provide a wealth of information, and there is little that a questionnaire could provide that the other methods could not. The second issue comes down to the policies in place within the school regarding questionnaires from external parties. Though the cause is a good one, and the questionnaire would originate from a trusted source, the school has a policy of declining questionnaire requests, to prevent students being exposed to ideas not in keeping with those officially sanctioned by the leadership team.

2.5.3 Observations

Due to many years of experience with the system, it was felt that no formal observation was required. However, these years of experience have resulted in a deep familiarity of the current solution, providing an accurate picture of its advantages and disadvatages. These can then be applied to the new system, in order to produce a product that builds on the good aspects of the current system, whilst building on areas that require solution. As a result of the informal nature of the observation (indeed, neither myself nor the teaching staff knew it was going on) there was no chance of bias, therefore avoiding some of the pitfalls common in observations, such as a non-standard routine being followed in order to provide a less than honest view of events to the observer.

2.5.4 Document Inspections

Whilst, at first glance, the inspection of documents related to a system is perhaps not as useful as other, potentially more active methods, such as observations or interviews, there is no doubt that they do serve a purpose. In this case, inspecting documents led to an understanding as to the suitability of the new system, and the ease with which it could be integrated into the school environment. Additionally, inspecting documents served as a supplement to the research gathered during the interviews and observations, providing hard evidence of a sort that backs up statements made in these sections. If the analysed system were more dependent on documents, then this method of investigation would likely have come in more useful.

2.6 IPSO Chart

This IPSO chart describes the inputs, outputs, storage locations and general processes that are associated with the system.

2.7 Limitations of Current System

There are evidently a large number of issues with the above method. Firstly, distributing the quizzes via a word-processed document is not a particularly

Table 1: My caption

Inputs	**Processes**
Quiz questions. Answers from each form.	Form tutors add up all the questions their form got right. Nick Bucknall calculates which form got the most questions correct.
Storage	**Outputs**
Quiz sent to forms via email, stored on a Word document.	Forms told in weekly assembly who won the quiz.

efficient method. It results in the network drive being cluttered with a variety of documents, perhaps with a non-existent naming scheme. This makes it harder for the form tutors to find the correct quiz for the week, slowing the whole process down. A more effective solution would be to have everything in it's own self contained system, with its own dedicated quiz screen, which points out the correct quiz to the tutors.

Additionally, having the questions and possible answers on the same document puts the integrity of the quiz at risk. Currently, form tutors get around this by hiding the document, but this can cause complications where students forget the possible answers, as well as other issues. It word be far more effective to always have the quiz displayed on screen on the interactive whiteboard, but the current system prohibits this.

Often, teachers forget about the quiz altogether, or believe it to be on a different date than when it is actually scheduled. This is a relatively common occurrence, and means that the quiz either has to be rescheduled (which those tutors who did remember find annoying), or that particular form has to miss out on the quiz that week; this can damage their overall reputation in the school community. A dedicated system could provide them a notification, perhaps via an email, that they should hold the quiz that afternoon. Additionally, the system could be set up in such a way that the quiz only begins once all the appropriate forms have connected.

Furthermore, the current system is not particularly fair. Students can spend as long as they wish on a single question, as long as the quiz is completed within the 25 minutes given to the form time. It would be fairer if the form was given a time limit of, say 60 seconds, after which the system automatically moves on to the next quiz.

The current system is also very isolated. Following the appointment of the new principal, the school has sought to implement the principle of "togetherness", whereby students work together more often. Though the current quiz system aligns itself with this philosophy to a degree (each form works together to come up with the answer), it could be improved by allowing a degree of interoperability between the forms. For example, if a quiz was being answered by all the forms in Year 8, one form could be given the opportunity to pose a question to the other forms, perhaps referencing one of the jokes sanctioned by the school.

By allowing the form tutors themselves to mark the quiz, their is a large risk of inaccurate results being reported back, possibly altered in such a way that favours the form. Though this allows the head of year to display trust to his team of form tutors, there exists in the school a very competitive atmosphere, increasing the chance that such malpractice will occur. A safer approach would be to allow the system to mark the form's answers, and then report this directly to the head of year.

Though the heads of year throughout the school possess many fine and admirable qualities, it would be remiss to apply to them the label of "mathematician". For simply calculating the best and worst performing forms for any given quiz, there are few issues with the current system (though it would be convenient if this was worked out automatically). It is when attempting to work out more complex results, such as the average score of a form over a period of several years, that the humble head of year falls short. A dedicated system would be able to perform a complicated analysis on the entire set of data it collects, allowing for a far more interesting report to be generated. This data could then be presented at an end of year, or even school, assembly, showcasing the best form in each category (or some other arbitrary statistic) throughout their entire school career.

Finally, the fact that the school is completely replacing the head of year system with new heads of house means that the entire approach is no longer possible.

3 Problem Definition

3.1 Broad Aims

1. Allow staff to create multiple choice quizzes for the year groups.
2. Give each individual form group their own user account.
3. Allow every student in the school to participate in the quiz together.
4. Display a live visualisation showing the current state of the quiz.
5. Calculate the winning house of the quiz overall and within each year group.

3.2 Possible Limitations

There are a number of limitations that the system will undoubtedly run into, mostly due to the limited amount of time available to develop the application, or because of the complexity of a feature..

One such limitation is the inability to provide an in-depth administration view for the heads of year. Such a view would allow them to do things like update the details of individual form groups, and add new groups through a dedicated interface. Though this would undoubtedly come in useful, and would not be especially difficult to implement, it would take additional time to build and test, time which would be better spent on improving the core quiz building and answering functionality.

Another feature that the application will not include will be the ability to automatically generate a quiz, using constraints set by the head of year. Whilst such functionality would help save the head of year time - they would not have to spend time creating their own individual quiz - it would be difficult to implement, as an algorithm would have to be written that selects the most appropriate questions for the year group, and so on. It would be a more beneficial use of time to make the actual quiz creation page easier to use, effectively avoiding the problem mentioned above.

Additionally, the application will not include quiz sharing functionality. Several other systems of this type include the ability to upload one's quiz to a publicly available site, allowing others to view and use the quiz. Though, once again, this feature would be helpful - and would actually be trivial to implement - it goes somewhat beyond the scope of the application, targeted, as it is, at a single school. A more suitable feature would be showing a list of recent quizzes and questions.

4 Objectives

The following objectives are based on research gathered in the interviews, observations and questionnaires that were carried out on the school, and so present an accurate picture of the features the school would find most desirable. Certain additional features have been added, as they would increase the utility of the system to even greater heights. If, at the end of development, these features have not been implemented, it would be impossible to refer to the system as a real success. The system must:

- Provide staff with an attractive user interface with which they can create, update and delete quizzes.
  - Quizzes should allow for an unlimited number of multiple choice questions to be included, each of which should contain four possible answers - one of which is correct.
  - Staff should be able to set how long they want to give students to answer the question, before the quiz moves on.

  - Staff should have the ability to define an unlimited number of categories for the quiz, such as “history” or “sport”; questions should be able to be placed in one of these categories.
  - Staff should be able to specify the exact date and time that a quiz should begin.
- Allow students of the school to answer the quiz in real time and compete against one another.
  - The system should support up to 1000 concurrent students partaking in the quiz without failing.
  - The quiz should begin at the specified time no matter how many users are connected. A five minute grace period should be given to take into account valid excuses for lateness.
  - The students should be presented with a clear and attractive interface that displays the current category, the question, and the possible answers. It should show the remaining time that students have to choose an answer.
  - When the time has elapsed for a question, the system should highlight the correct answer, and then immediately move onto the next question.
  - Staff should have the ability to define an unlimited number of categories for the quiz, such as “history” or “sport”; questions should be able to be placed in one of these categories.
  - All of the above should take place simultaneously, on every individual screen.
  - At the end of the quiz, the system should display the winning house, as well as the number of house points earned by each house.
- Display a real time visualisation of how other participants in the quiz are answering.
  - During the quiz, a pull out section should be available that displays the answers that other form groups in the school have chosen.
  - If the student chooses to look at the panel before answering the question, they should only receive half a mark for their answer.
  - All of the above should take place simultaneously, kept up to date on every individual screen.
- Work effectively across a range of different devices and display types. The school plans to use the system across tablets and desktop computers. No matter which device a user is using, all aspects of the system should be available and fully functional.

- Theme itself to match the house colours of the logged in user. For example, the interface should be orange for those who belong to Baxter, green for Clive, blue for Acton, and so on.

Part II

Application Design

This next part of the documentation contains the designs for multiple parts of the system, created before production began. Designs for the input and output screens are included, as well as validation routines, data structure designs (including methods of access), and designs for the processses that will be used, written in pseudo-code.

5 Data Structures

Due to the ease with which a quiz fits a nested structure, an object based, or NoSQL database, will be used. This type of database differs from a standard relational database in that it does not make use of tables; instead, all information pertaining to an entity is stored in a single object. This lack of traditional relationships causes issues when attempting to create things like ERD diagrams. Whilst NoSQL databases are traditionally schemaless, modelling tools will be used in this project in order to better suit the course specification.

5.1 Quiz Structure

Below is the structure that the actual quiz will take. The following displays the schema for the quiz, including the data types for each element. The methods of access taught in this specification are not applicable to this structure.

5.1.1 Quiz

This is the data dictionary / schema for the main quiz. It will be the base level of the quiz object, containing the meta-data, and a link to the categories schema.

Field Name	**Data Type**	**Length**	**Validation**
title	string	variable	Presence check
startTime	datetime	variable	After current date
questionLength	integer	variable	Greater than 0
intervalLength	integer	variable	Greater than 0
categories	Array<schema>	variable	Presence check

Table 2: Quiz Schema

5.1.2 Categories

This is the data dictionary / schema for the categories section of the quiz. It will be the first level of the quiz object, containing the category name and a link to the questions schema.

Field Name	Data Type	Length	Validation
title	string	variable	Presence check
questions	Array<schema>	variable	Presence check

Table 3: Categories Schema

5.1.3 Questions

This is the data dictionary / schema for the questions. It will be the second level of the quiz object, containing the individual questions and a link to the answers schema.

Field Name	Data Type	Length	Validation
title	string	variable	Presence check
answers	Array<schema>	variable	Presence check

Table 4: Questions Schema

5.1.4 Answers

This is the data dictionary / schema for the questions. It will be the third and final level of the quiz object, containing the individual answers to a question.

Field Name	Data Type	Length	Validation
text	string	variable	Presence check
correct	Boolean	1	Presence check

Table 5: Answers Schema

5.1.5 Completed Quiz Object

By combining the above schemas, one ends up with a whole quiz, containing the different areas of the object combined into one. Taking just the field names without any data, the structure will look like this:

```
{
  title: String,
  start: Data,
  questionIntervals: Integer,
  intervalLength: Integer,
  categories: [
    {
      title: String,
      questions: [
        {
          title: String,
          answers: [
            { text: String, correct: Boolean },
            { text: String, correct: Boolean },
            { text: String, correct: Boolean },
            { text: String Schema, correct: Boolean }
          ]
        }
      ]
    }
  ]
}
```

As can be seen, at the beginning of the structure lies some metadata pertaining to the quiz, such as when it is due to start and the title. This is then followed by a list of categories, each of which is named, each of which itself contains the questions and answers. This nested structure fits the idea of a quiz rather nicely. The categories array holds a series of objects, each of which have a *title* attribute. Next to this title lies an array of questions - the nesting indicates that a question belongs to a category - each of is another object, themselves containing an array of answers.

5.1.6 Example Quiz

An example quiz has been included below in order to give the reader an impression of how a real quiz might look in the database:

```
{
  title: 'Summer Term House Quiz',
  start: new Date(),
  questionIntervals: 10000,
  realtimeGraphics: true,
  intervalLength: 300000,
  categories: [
    {
      name: 'History',
      questions: [
        {
          title: 'Who killed JFK?',
          answers: [
            { text: 'Bill Osmond', correct: false },
            { text: 'Lee Harvey Oswald', correct: true },
            { text: 'Henry Kissinger', correct: false }
          ]
        },
        {
          title: 'Who was the British Prime Minister in 1916?',
          answers: [
            { text: 'David Lloyd George', correct: true },
            { text: 'Lee Harvey Oswald', correct: false },
            { text: 'Lord Liverpool', correct: false }
          ]
        }
      ]
    },
    {
      name: 'Sport',
      questions: [
        {
          title: 'Who won the 1966 world cup?',
          answers: [
            { text: 'England', correct: true },
            { text: 'Wales', correct: false },
            { text: 'Scotland', correct: false },
            { text: 'Uzbekistan', correct: false }
          ]
        },
        {
          title: 'Which city hosted the 1924 Summer Olympics?',
          answers: [
            { text: 'Brussels', correct: false },
            { text: 'Orelans', correct: false },
            { text: 'Madrid', correct: false },
            { text: 'Paris', correct: true }
          ]
        }
      ]
    },
    {
      name: 'Literature',
      questions: [
        {
          title: 'Who serves as the main character in "Crime and Punishment"?',
          answers: [
            { text: 'Raskolnikov', correct: true },
            { text: 'Razumikhin', correct: false },
            { text: 'Porifry Petrovich', correct: false },
            { text: 'Svidrigailov', correct: false }
          ]
        },
        {
          title: 'Which literary genre does "Ulysses" belong to?',
          answers: [
            { text: 'Modernist', correct: true },
            { text: 'Post-modernist', correct: false },
            { text: 'Romance', correct: false },
            { text: 'Thriller', correct: false }
          ]
        }
      ]
    }
  ]
}
```

5.2 Answer Packet

The following is the structure of the data packet that will be emitted when an answer is chosen. Due to the number of packets that will be sent in any one quiz - 22,5000 on a 25 question quiz containing 900 students - this structure is very small. It will serve as an input to practically all the algorithms throughout the system.

```
{
  house: Character,
  year: Integer,
  questionID: Integer,
  answer: Character
}
```

By splitting the *house* and *year* items, normalisation is achieved. This approach means that the form and house can be treated seperately - but computed as one value when needed - allowing for more complicated statistics to be gathered.

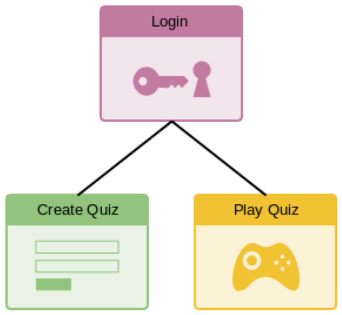
6 API Endpoints

The following table is a plan of the API endpoints that will be used to receive some of the less volatile data, containing the HTTP method used to access the route, the route itself, any parameters used, and an explanation of what the endpoint will return.

Method	Endpoint	Role
GET	/api/quizzes	Return all the quizzes.
GET	/api/quizzes/:id	Return a specific quiz.
POST	/api/quizzes	Add a new quiz.
PUT	/api/quizzes/:id	Update a specific quiz.
DELETE	/api/quizzes/:id	Delete a specific quiz.

7 Route Structure

The following hierarchy diagram depicts the route strucure that the application will follow, demonstrating how each section of the application will link. As



shown, there will be only three views - the login view, where the user chooses their year and form group to begin the quiz; a section for administrative staff to create a quiz, and the actual quiz playing interface. The quiz creation interface will be accessed via a separate link on the login view, as depicted in the interface designs.

8 User Interface

This section features designs for the different areas of the system. Wireframes of each screens are provided, giving a general impression of what the screen will look like once implemented. It should be remembered, of course, that design, particularly that of a visual nature, is an iterative process, and, as such, the final product may differ, perhaps wildly, from the screens featured here.

8.1 Output Data

The system will output the number of house points that each house has earned, displaying a running total after each question. At the end of the quiz, the overall winner will be displayed (the house with the most points). This is needed to effectively integrate the quiz system into the school's routine and provide a winner in a form with which the staff and students can relate. During operation, the system will produce a large number of the answer packets that have been described above. These are needed to inform the system of how many house points each house gets and whether they have got the answer correct.

8.2 Input Methods

The school plans to use the system on both desktop PCs and tablets. As such, the system will support input through both a mouse and keyboard, as well as a touchscreen. For the quiz creator, input boxes will be used to allow users to enter information like the quiz title, questions, answers, etc. Buttons will also be used, allowing users to perform actions. In the quiz playing section, buttons will be used, as all the information needed for the quiz will have already been entered.

8.3 Colours and Typography

As revealed through interviews conducted with staff, the school is placing a great deal of emphasis on the design of the application. As such, a unique approach to the interface will be taken, one that resonates with the new found focus on houses within the school.

8.3.1 Typography

In order to make the system yet more appealing and approachable, the font "Dosis" will be used throughout. As show by the example below, Dosis is a friendly font, one that invites the user to perform actions. This is in line with the objectives for the system, and will make the system suitable for all ages in the school. An example can be seen in the next subsection.

8.3.2 Colours

Each house has their own colour, and this colour will be used to theme all elements of the interface, depending on the house that a student is logged in as. The colour mappings are as follows:

- **Acton (Blue)**
- **Baxter (Orange)**
- **Clive (Turqoise)**

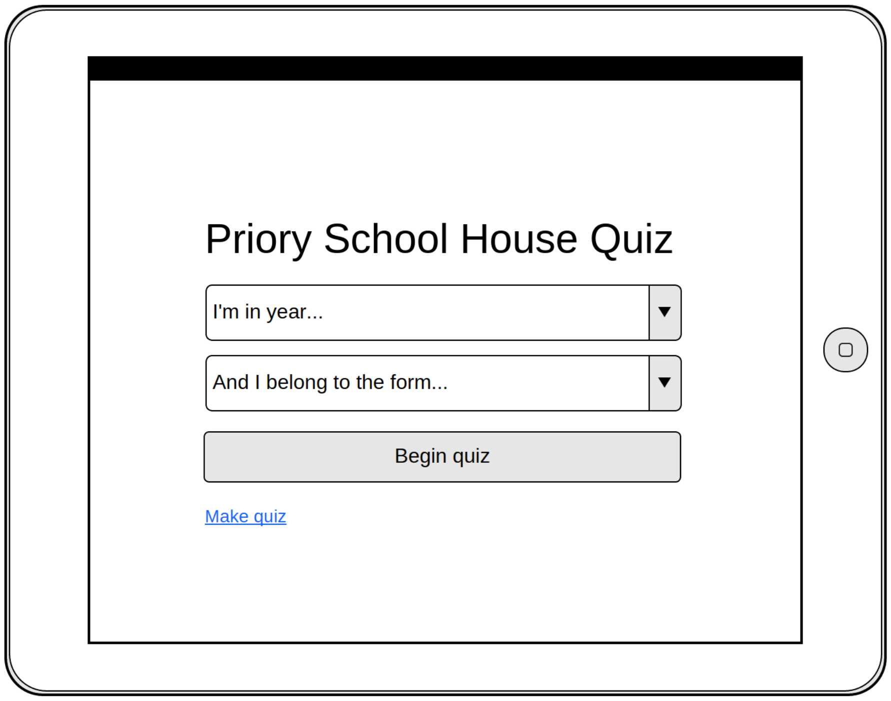
- Darwin (Purple)
- Houseman (Red)
- Webb (Yellow)

In order to ensure that each user sees a subtedly different design, even if they are in the same house, the colours will be randomised. For example, a user in Acton might see buttons in a light blue and drop down boxes in a dark blue, whilst another user in Acton would see slightly darker or lighter shades. In order to make the system more appealing to use, the luminosity of each colour will be locked to a certain brightness.

8.4 Input Screens

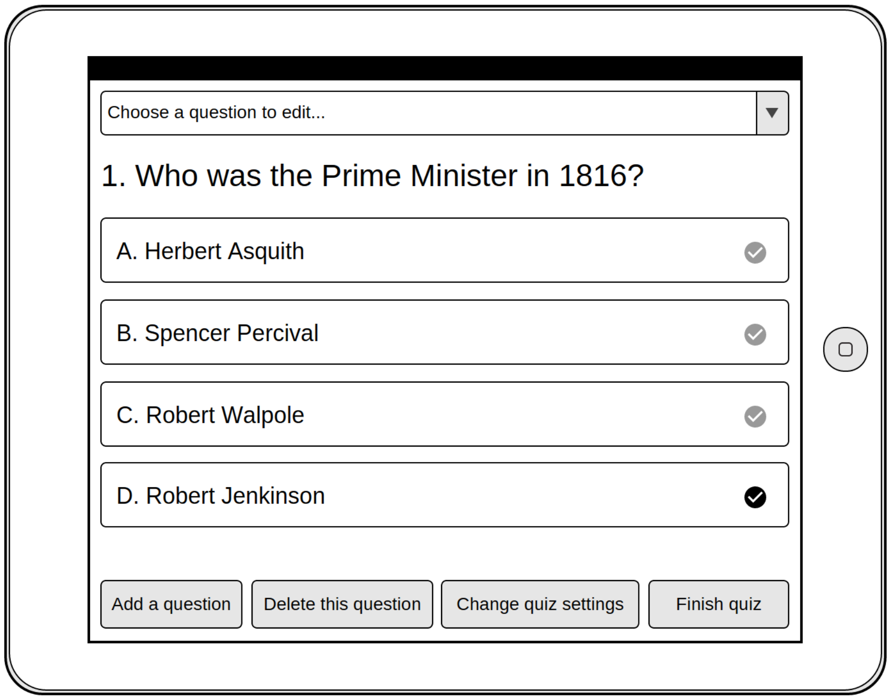
These are all the screens that will allow the user to input or select data of some description.

8.4.1 Login Screen



Due to the way the school names individual form groups, a simplified approach to the login form can be used. Two drop down boxes will be used, one for the year - containing the items 7, 8, 9, 10 and 11 - and one for the house - Acton, Baxter, Clive, Darwin, Houseman, Webb. By doing this, any combination of form group can be computed. Interviews with students revealed that they would not be likely to pose as a member of another form group, so no authentication is needed - students need simply enter their year group and house, and then press the “Begin quiz” button. If a senior member of staff needs to create a quiz, clicking the link will bring them to a password prompt, in order to prevent errant students making a quiz themselves. Because the user will not have chosen their house, house colours will not be available. Instead, the colours of all the houses will feature, with the interface elements changing every few seconds to the next house.

8.4.2 Quiz Creation Screen

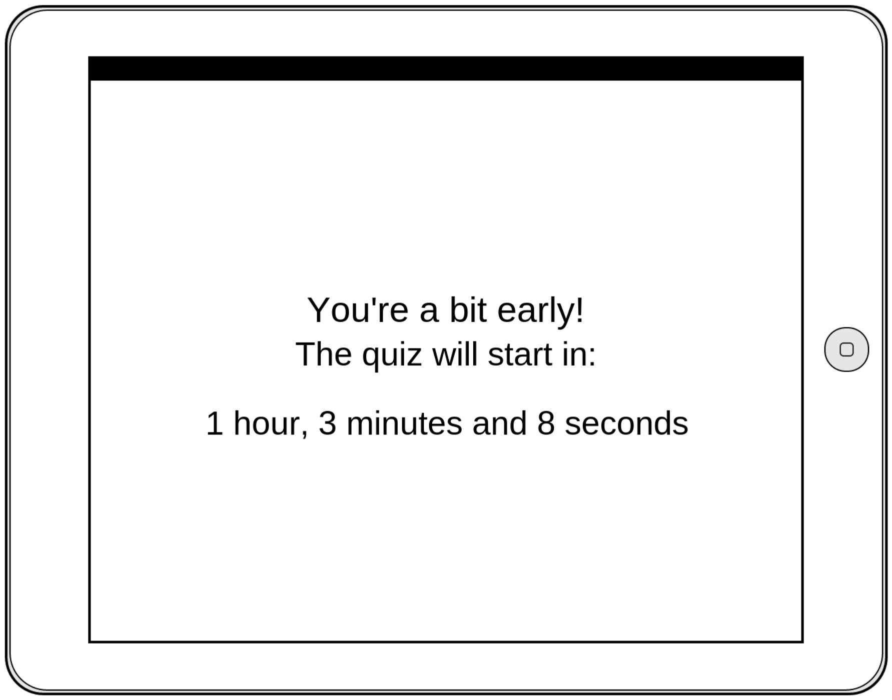


This screen will be available to members of staff, and will allow them to create quizzes for the students to consume. A dropdown box at the top will allow staff to switch between questions in the quiz, enabling them to easily go back and make changes. The question title is displayed just below this; tapping on it will allow staff to make changes. Just below this will be the four possible answers to the question (each question must have four answers), which staff can add in appropriately. By the side of each possible answer will be a complication to allow staff to mark an answer as correct; only one answer can be correct at a time. Below this are a series of buttons, granting the ability to add or delete questions, modify the quiz settings, such as the title; and leave the quiz maker.

8.5 Output Screens

These are the screens that output data of some description, often performing some sort of calculation on the data to modify the way it is output.

8.5.1 Countdown Screen



You're a bit early!
The quiz will start in:

1 hour, 3 minutes and 8 seconds

In the event that a student logs onto the system and a quiz is yet to start, this screen will be displayed, informing them how long they have to wait. The countdown timer will display the time till the next scheduled quiz in hours, minutes and seconds, and will update accordingly.

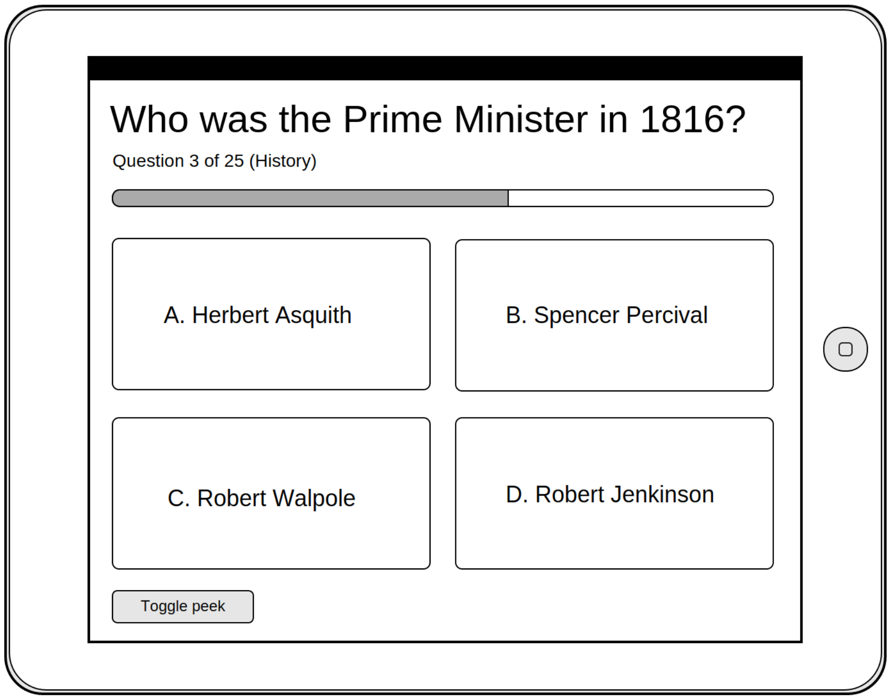
8.5.2 Results Screen

This screen will display the results of the quiz after all the questions have been answered.

8.6 Input and Output Screens

One screen, the actual quiz playing interface, will both allow users to input data - choosing the correct answer - and output data - displaying the quiz and the current live results.

8.6.1 Play Quiz Screen



This is the main screen of the system, and the one that users will spend the most time on - the actual quiz interface. At the top, the current question will be displayed, enabling users to gain a clear impression of what they are meant to be answering. Below, the current position in the quiz, including the current category, will be shown, letting users see how much longer the quiz will last. Next to this wil be a progress bar acting as the question timer - it will fill up for each question, providing a visual representation of how long the user has left to answer the question. Below this will be the actual question answers; to select one, the user need simply tap on it. Just below this will be a button to activate "peek mode". This will activate the view so exalted by Nick Bucknall in the interviews, displaying a real time overlay on the question options of how other people in the school are voting. Pressing the button again will turn this view off, though it will be activated automatically once the user chooses an answer.

9 Process Design

This section features pseudocode for the of processes that will be used throughout the system. Due to the functional methods with which the application will be programmed, many of these processes (referred to hereafter as functions) will be used multiple times, and combined with other functions. To aid in readability, pseudocode for each function will be listed only once. The following is a list of all the processes that will be used:

- Create, read, update and delete quizzes.
- Display time until the quiz begins.
- Validation for presence, range and date.
- Calculate suitable colour palette for each house.
- Calculate the number of house points for each packet.
- Calculate / report the most common answer in each house / year.
- Report if the student chose the correct answer.
- Report the winner of the quiz, in both the year and house.
- Keep state of quiz in sync between each device.

9.1 Validation

Validation will be used on every input throughout the system. This section lists the psuedocode that will power the different validation functions.

9.1.1 Presence Check

This function will return true if a field contains a value.

```
process presenceCheck (field) {
  if (field.length) {
    return true
  } else {
    return "A value must be entered."
  }
}
```

9.1.2 Range Check

This function will return true if a field's value is greater than a specified value.

```
process rangeCheck (field, max)
  if (field.body > max) {
    return true
  } else {
    return "Value must be greater than 0."
  }
}
```

9.1.3 Date Check

This function will return true if a field's date is before the current date.

```
process validDate (field) {
  set currentDate = the current date
  if (field.date before currentDate) {
    return "Date must be after current date."
  } else {
    return true
  }
}
```

9.2 API Functions

Though a relatively small section of the application, the API nevertheless serves a vital role. Reading and writing from the database, the API routes will return the raw data for the quiz.

9.2.1 Create Entity

This function will save an entity to the database.

```
process POST (entity, request) {
  create new instance of entity
  set properties of entity to request properties
  call database save function
  if error occurs, return error
  else return "Entity created!"
}
```

9.2.2 View Entities

This function will return all entities of a specified type in a JSON file.

```
process GET (entity) {
  call database find query for entity
  if error occurs, return error
  else return JSON of entities
}
```

9.2.3 View Single Entity

This function will return a single entity of a specified type in a JSON file.

```
process GET (entity, id) {
  call database find query for entity using ID
  if error occurs, return error
  else return JSON of entity
}
```

9.2.4 Update single entity

This function will update a single entity in a database.

```
process PUT (entity, id, props) {
  call database find query for entity using ID
  go through new properties and assign to the entity.
  return JSON of entity
}
```

9.2.5 Delete single entity

This function will delete entity in a database.

```
process DELETE (entity, id) {
  call database find query for entity using ID
  delete entity from database
  return "entity has been deleted!"
}
```

9.3 Reducers

These processes will be used to manage the internal state of the application, storing information that the individual components need to function properly.

9.3.1 Colour Reducer

This reducer will control the colour theme, storing all the different sections in an object that can be passed down to all the application's sub-components.

```
process colours(state = call colourScheme(acton), action) {
  if (action = change colours) {
    replace colours with colours returned from colourScheme(action.house)
  }
}
```

9.3.2 User Reducer

This reducer will control the functions on the login screen, managing common state like the house and year of the user and the themed colours for the application.

```
process user(state = blank, action) {
  if (action = change house) {
    return the action's house
  }
  if (action = change year) {
    return the action's year
  }
  if (action = request quizzes) {
    return current state
    set requestingQuizzes to true
    set requestingQuizzesFailed to false
  }
  if (action = request quizzes failure) {
    return current state
    set requestingQuizzes to false
    set requestingQuizzesFailed to true
  }
  if (action = receive quizzes) {
    return current state
    set quizzes to action.quizzes
    set requestingQuizze to false
  }
  if (action = delete quiz success) {
    return current state
    filter out quizzes where quiz.id != action.id
  }
  if (action = quiz is ready) {
    return current state
    set quizIsReady to action.quizIsReady
  }
}
```

9.3.3 Quiz Reducer

```
process settings(state = empty array, action) {
  if (action = update ID) {
    replace current ID with action ID
  }
  if (action = update title) {
    replace current title with action title
  }
  if (action = update start date) {
    replace current start date with action start date
  }
  if (action = update start time) {
    replace current start time with action start time
  }
  if (action = update question length) {
    replace current question length with action question length
  }
  if (action = update break length) {
    replace current break length with action break length
  }
  if (action = update is finished) {
    replace current is finished with action is finished
  }
  if (action = update all settings) {
    replace current all settings with action all settings
  }
}

process categories(state = empty array, action) {
  if (action = add category) {
    return new object combining current
    state and keys id and name set to the
    action properties
  }
  if (action = edit category) {
    map over each category in the array
    if category id = action.id replace the category name
    else add the category untouched
  }
  if (action = delete category) {
    filter out categories where category id = action id
  }
  if (action = delete all categories) {
    filter out categories where category id != -1 (all)
  }
}

process questions(state = empty array, action) {
  if (action = add question) {
    return new object combining current
    state keys, id, name and category id set to the
    action properties
  }
  if (action = edit question) {
    map over each question in the array
    if question id = action.id replace the question body
    else add the question untouched
  }
  if (action = delete question) {
    filter out questions where question id = action id
  }
  if (action = delete all questions) {
    filter out questions where question id != -1 (all)
  }
}

process answers(state = empty object, action) {
  if (action = add answer) {
    return new object combining current
    state keys, id, name, correct and question id set
    to the action properties
  }
  if (action = edit answer) {
    map over each answer in the array
    if answer id = action.id replace the answer name and
    correct keys else add the answer untouched
  }
  if (action = delete answer) {
    filter out answers where answer id = action id
  }
  if (action = delete all answers) {
    filter out answers where answer id != -1 (all)
  }
}
```

9.3.4 Current Quiz Reducer

This reducer controls all the data needed for the running quiz, and allows it to be accessed by all the necessary sub-components.

```
process currentQuestion(state = 0, action) {
  if (action = show next answer) {
    set state.questionId to action.questionId
  }
}

process timeLeft(state = 10000, action) {
  if (action = decrement tine keft) {
    set state.timeLeft to action.timeLeft
  }
}

process inProgress(state = false, action) {
  if (action = begin quiz) {
    set quizInProgress to true
  }
  if (action = leave quiz) {
    set quizInProgress to false
  }
}

process answerStatistics(state = {
  acton: 0,
  baxter: 0,
  clive: 0,
  darwin: 0,
  houseman: 0,
  webb: 0
  }, action) {
    if (action = receive answer) {
      set state[answer.house] to answer.stats
    }
  }
```

9.4 Scoring

Much of the application will focus on scoring the quiz, using the answer packets that each answer to a question emits.

9.4.1 Calculate House Points

This function will calculate the number of house points that an answer packet generates, and then compute an up to date list of the current score for either each year or house.

```
process housePoints(packet, correct, keys, prop, state = {}) {
  if state is empty {
    loop through each key
    set state[key] = 0
  }

  if answer is correct and peek is false {
    set score = 1
  } else if answer is correct and peek is true {
    set score = 0.5
  } else set score = 0

  return clone of state
  set packet[prop] to state[packet[prop]] add score
}
```

By taking a property value as an argument, the function can be used to calculate house points for any specified group, making it more versatile. Additionally, by only computing the most recent state for a single answer packet, and using the previous state as a starting point, it becomes easier to apply the function in the real time scenario it will be used in - an average of 22,500 calls per quiz, many happening at once.

9.4.2 Highest Value in Object

This function will calculate the highest value in a given object. It will be used in the most common answer function, displayed below.

```
process highest(object) {
  set variable keys to array of object keys

  set variable vals to map over object using
  keys variable for an array of object values

  call max function from standard library on variable vals
  to get the highest value
}
```

9.4.3 Correct Answer

Using an answer packet, returns true if the user chose the correct answer.

```
process isCorrect(packet, correctAnswer) {
  if packet.answer = correctAnswer then
    highlight box green.
  else highlight box red.
}
```

9.4.4 Maximum Keys

This function will take an object where every key is a number. It will return the key of the highest number.

```
process maxKey(object) {
  perform reduce on object ->
  if obj[a] > obj[b] then return a
  else return b
}
```

9.4.5 Most Common Answer

Using the answere packet, this function will work out the most common answers for each question at any one point in the quiz. It works in much the same way as the function for calculating house points, but the appending operations differ so a new function is needed.

```
process commonAnswer(packet, keys, state) {
  set variable innerTree to object containing keys: {
    mode: undefined,
    array with houses: 0
  }

  if the state is empty, set state to innerTree

  set newState to a clone of state, but set the answer
  frequency for the house the user chose to itself + 1

  set the mode property on newState to the largest
  value out of the house frequencies

  return newState
}
```

9.4.6 Display Time Left

This function will display the time left until the quiz begins.

```
Send GET request for latest quiz.
Access startTime property on quiz.
Express startTime as a quantity of hours, minutes and seconds.
Display this on the screen.
When the countdown has elapsed, begin the quiz.
```

9.4.7 Calculate Colour Palette

This function will calculate a themed colour palette for the system, depending on the user's house.

```
process randomColours(house) {
  if (house = 'acton') set hue to blue
  if (house = 'baxter') set hue to orange
  if (house = 'clive') set hue to green
  if (house = 'darwin') set hue to purple
  if (house = 'houseman') set hue to red
  if (house = 'webb') set hue to yellow

  generate 10 random colours based on hue

  modify luminance for each one based on complementing
  factors, by a random number between 0.1 - 0.5

  return set of colours

}
```

10 Evaluation Critera

When evaluating the system, several factors will to be taken into account to determine if the system itself, and the development process, is a success. They are listed below:

- Performance of the system should be considered. The system should be able to cope with at least 1000 connections at any one time, all communicating with each other. This communication should all be real time and seamless, and there should not be any lags or stutters caused directly by the system.

- Staff and students should find the user interface simple and easy to use; they should instinctively know how to use the system, with a minimum amount of training required. This will be measured by observing users as they attempt to use the different sections of the system.
- As the school requires the system relatively quickly, it is important that development of the system is completed within a reasonable period of time. To achieve this, it is important that time is spent only on features that the school have asked for, and not on "useful" extras.
- The application code should be as high quality as possible - where possible, functional paradigms, such as using *map* as opposed to a loop, should be followed, and code should be kept as modular and reusable as possible, to bring in all the advantages that such a development model provides.
- The cost of the system to run should also be taken into account. Though the school will not be charged directly for the system, there will be costs relating to its continued maintenance and uptime. If these are overly expensive, there will be a negative impact on the success of the project.
- Stability is an important factor, and it will also be considered. If the system is constantly prone to crashing, it could not be called stable; likewise, it would also indicate a failure of the test plan.
- The impact of the system will also be considered. If the school decides that there are no benefits to using the system - in other words, it has failed to solve the problems laid out in the analysis - the system cannot be considered a success.
- The suitability of the test plan will also be evaluated. A test plan is important because it provides a way of ensuring the parts of the system function correctly.
- The system must meet all of the objectives, as follows:
    - Provide staff with an attractive user interface with which they can create, update and delete quizzes.
        - Quizzes should allow for an unlimited number of multiple choice questions to be included, each of which should contain four possible answers - one of which is correct.
        - Staff should be able to set how long they want to give students to answer the question, before the quiz moves on.
        - Staff should have the ability to define an unlimited number of categories for the quiz, such as "history" or "sport"; questions should be able to be placed in one of these categories.
        - Staff should be able to specify the exact date and time that a quiz should begin.

- Allow students of the school to answer the quiz in real time and compete against one another.
  * The system should support up to 1000 concurrent students partaking in the quiz without failing.
  * The quiz should begin at the specified time no matter how many users are connected. A five minute grace period should be given to take into account valid excuses for lateness.
  * The students should be presented with a clear and attractive interface that displays the current category, the question, and the possible answers. It should show the remaining time that students have to choose an answer.
  * When the time has elapsed for a question, the system should highlight the correct answer, and then immediately move onto the next question.
  * Staff should have the ability to define an unlimited number of categories for the quiz, such as “history” or “sport”; questions should be able to be placed in one of these categories.
  * All of the above should take place simultaneously, on every individual screen.
  * At the end of the quiz, the system should display the winning house, as well as the number of house points earned by each house.
- Display a real time visualisation of how other participants in the quiz are answering.
  * During the quiz, a pull out section should be available that displays the answers that other form groups in the school have chosen.
  * If the student chooses to look at the panel before answering the question, they should only receive half a mark for their answer.
  * All of the above should take place simultaneously, kept up to date on every individual screen.
- Work effectively across a range of different devices and display types. The school plans to use the system across tablets and desktop computers. No matter which device a user is using, all aspects of the system should be available and fully functional.
- Theme itself to match the house colours of the logged in user. For example, the interface should be orange for those who belong to Baxter, green for Clive, blue for Acton, and so on.

If, after the completion of the project, these criteria are not met, it would be impossible, or at least very difficult, to label the system a complete success.

Part III

Program Documentation

This second part of the documentation contains a number of sections concerning the finished application. It includes aspects like screenshots of the finished system, as well as a brief discussion of how they are fit for purpose; detailed information on the final database tables, including a diagram detailing their links; a list of every variable used throughout the system, both global and local in scope; and annotated, self-documenting code listings for the entire source of the system, detailing exactly how it works.

11 Annotated Listings

This section contains all of code for system, split into several logical categories. The system is made up of a very large number of Python functions, as well as some additional aspects, such as Jinja2 HTML templates to display interface, and CSS to provide styling.

11.1 API Routes

This subsection lists the RESTful API routes - the section of the program that manages the database.

11.1.1 server.js

This file bootstraps the backend server and loads all the different modules.

```
import path from 'path';
import express from 'express';
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import webpack from 'webpack';
import webpackConfig from './webpack.config';
import config from './config';
import apiRoutes from './src/api/index';
import quizSockets from './src/sockets/quizSockets';

// Begin connection to database.
mongoose.connect(`mongodb://localhost:27017/${config.database}`);

let app = express();
let compiler = webpack(webpackConfig);

// Initialise essential middleware.
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// Begin the server; listen on the defined port.
let server = app.listen(config.port, '0.0.0.0', err => {
  if (err) throw err;
  console.log('Listening on port ' + config.port);
});

// Set up development middleware.
app.use(require('webpack-dev-middleware')(compiler, {
  noInfo: true,
  publicPath: webpackConfig.output.publicPath
}));

app.use(require('webpack-hot-middleware')(compiler));

// Set root route (pun!) to serve static index.html file.
app.get('/', (req, res) => res.sendFile(path.join(__dirname, 'index.
    html')));

// Fix the bloody annoying error message when refreshing.
app.get('/create', (req, res) => res.redirect('/'));
app.get('/play', (req, res) => res.redirect('/'));
app.get('/results', (req, res) => res.redirect('/'));

// Set the /api endpoint to the route logic in apiRoutes.
app.use('/api', apiRoutes);

// Start the socket processes.
quizSockets(server);
```

Listing 1: Bootstraps the backend.

11.1.2 quizzes.js

```
import express from 'express';

import Quiz from '../models/Quiz';

let router = express.Router();

/**
 * GET /api/quizzes
 * Returns all quizzes.
 */
router.get('/', (req, res) => {
  Quiz.find({}, '-__v', (err, quizzes) => {
    if (err) throw err;
    res.send({ quizzes });
  });
});

/**
 * POST /api/quizzes
 * Adds a new quiz to the database.
 */
router.post('/', (req, res) => {
  let quiz = new Quiz();

  quiz.title = req.body.title;
  quiz.startDate = req.body.startDate;
  quiz.questionLength = req.body.questionLength;
  quiz.breakLength = req.body.breakLength;
  quiz.isFinished = req.body.isFinished;
  quiz.categories = req.body.categories;

  quiz.save((err, quiz) => {
    if (err) throw err;
    res.send({ message: quiz.title + ' was saved!', quiz });
  });
});

/**
 * GET /api/quizzes/:id
 * Returns a specific quiz.
 * @param  {quizId} id The id of the quiz to return.
 */
router.get('/:quizId', (req, res) => {
  Quiz.findById(req.params.quizId, (err, quiz) => {
    if (err) throw err;
    res.send(quiz);
  });
});

/**
 * PUT /api/quizzes/:id
 * Updates a specific quiz.
 * @param  {Object_id} id The id of the quiz to update.
 */
router.put('/:quizId', (req, res) => {
  Quiz.findById(req.params.quizId, (err, quiz) => {
    if (err) throw err;

    for (let property in req.body) {
      quiz[property] = req.body[property];
    }

    quiz.save(err => {
      if (err) throw err;
      res.send({ message: 'Quiz updated!', quiz });
    });
  });
});

/**
 * DELETE /api/quizzes/:id
 * Deletes a specific quiz.
 * @param  {Object_id} id The id of the quiz to delete.
 */
router.delete('/:quizId', (req, res) => {
  Quiz.remove({ _id: req.params.quizId }, (err, quiz) => {
    if (err) throw err;
    res.send({ message: 'Quiz was deleted!' });
  });
});

export default router;
```

Listing 2: Defines the resources available at /api/quizzes.

11.1.3 index.js

```
import express from 'express';

import quizzes from './routes/quizzes';

let router = express.Router();

router.use('/quizzes', quizzes);

export default router;
```

Listing 3: Exports API routes for easier importing.

11.2 API Models

This subsection lists the RESTful API resource models, defining exactly what a quiz, form, etc. look like.

11.2.1 Quiz.js

```
import mongoose from 'mongoose';
import categorySchema from './Category';

const quizSchema = new mongoose.Schema({
  title: String,
  startDate: Date,
  questionLength: Number,
  breakLength: Number,
  isFinished: Boolean,
  categories: [categorySchema]
});

export default mongoose.model('Quiz', quizSchema);
```

Listing 4: Defines a quiz.

11.2.2 Category.js

```
import mongoose from 'mongoose';
import questionSchema from './Question';

const categorySchema = new mongoose.Schema({
  body: String,
  questions: [questionSchema]
});

export default categorySchema;
```

Listing 5: Defines a category.

11.2.3 Question.js

```
import mongoose from 'mongoose';
import answerSchema from './Answer';

const questionSchema = new mongoose.Schema({
  body: String,
  answers: [answerSchema]
});

export default questionSchema;
```

Listing 6: Defines a question.

11.2.4 Answer.js

```
import mongoose from 'mongoose';

const answerSchema = new mongoose.Schema({
  body: String,
  correct: Boolean
});

export default answerSchema;
```

Listing 7: Defines an answer.

11.3 Utility Functions

This subsection lists the utility functions used through the system. These functions, often used multiple times in different places, are used to extract out common functionality into its own module.

11.3.1 maxKey.js

```
function maxKey(obj) {
  return Object.keys(obj).reduce((a, b) =>
    obj[a] > obj[b] ? a : b
  );
}

export default maxKey;
```

Listing 8: Returns the largest key in an object.

11.3.2 nextBiggest.js

```
function nextBiggest(array, key = 'id') {
  // Collect all the ids in the array.
  const ids = array.reduce((arr, obj) => [...arr, obj[key]], []);
  // Finds the next biggest value in an array.
  return Math.max(...ids) + 1;
}

export default nextBiggest;
```

Listing 9: Returns the next largest value given an object.

11.3.3 choice.js

```
'use strict'
let choice = array => array[Math.floor(Math.random() * array.length)];

module.exports = choice;
```

Listing 10: Returns a random element from an array.

11.3.4 backgroundStyle.js

```
export default house => {
  const backgroundMap = {
    'acton': '#8dd6f8',
    'baxter': '#FFF7E9;',
    'clive': '#78ecf6',
    'darwin': '#cd99f5',
    'houseman': '#f88e8c',
    'webb': '#fae07c'
  };

  return {
    // height: window.innerHeight + 'px',
    // width: '100%',
    // backgroundColor: '#fff',
    padding: '10px'
  };
};
```

Listing 11: Returns the correct background style depending on the house.

11.3.5 settingsStyle.js

```
function settingsStyle(borderColour) {
  return {
    content: {
      top: '50%',
      left: '50%',
      right: 'auto',
      bottom: 'auto',
      padding: '35px',
      marginRight: '-50%',
      transform: 'translate(-50%, -50%)',
      border: `4px solid ${borderColour}`
    }
  };
}

export default settingsStyle;
```

Listing 12: Returns the correct settings styles depending on the house.

11.3.6 defaultQuiz.js

```
import moment from 'moment';

const defaultQuiz = {
  settings: {
    id: '',
    title: 'Priory School Quiz',
    startTime: moment().format('hh:mm:ss'),
    startDate: moment().format('ddd MMM D YYYY'),
    questionLength: 10000,
    breakLength: 300000,
    isFinished: false
  },
  categories: [
    { id: 0, body: 'Default' }
  ],
  questions: [
    { id: 0, categoryId: 0, body: 'I\'m the question title - tap to
        edit me!' }
  ],
  answers: [
    { id: 0, questionId: 0, body: 'I\'m the first possible answer!',
        correct: false },
    { id: 1, questionId: 0, body: 'You can edit any of us by tapping on
         our text.', correct: false},
    { id: 2, questionId: 0, body: 'See that bold check mark? It means I
        \'m the correct answer.', correct: true },
    { id: 3, questionId: 0, body: 'Tapping on another check mark will
        make that the correct answer.', correct: false }
  ]
};

export default defaultQuiz;
```

Listing 13: Returns the default quiz and is loaded into state whenever a new quiz is created.

11.3.7 colourScheme.js

```
function colourScheme(house) {
  const colours = {
    acton: {
      'answer': {
        'body': {
          'backgroundColor': '#f78f85',
          'borderColor': '#c6726a',
          'color': '#945650'
        },
        'check': {
          'color': '#633935'
        }
      },
      'select': {
        'backgroundColor': '#85edf7',
        'borderColor': '#6abec6',
        'color': '#508e94'
      },
      'button': {
        'backgroundColor': '#a0ffff',
        'borderColor': '#80cccc',
        'color': '#609999'
      },
      'text': {
        'primary': {
          'color': '#71c9d2'
        },
        'secondary': {
          'color': '#78d5de'
        }
      },
      'settings': {
        'borderColour': '#c6726a',
        'label': {
          'color': '#c6726a'
        }
      }
    },
    baxter: {
      'answer': {
        'body': {
          'backgroundColor': '#9fdffc',
          'borderColor': '#7fb2ca',
          'color': '#5f8697'
        },
        'check': {
          'color': '#405965'
        }
      },
      'select': {
        'backgroundColor': '#fcbc9f',
        'borderColor': '#ca967f',
        'color': '#97715f'
      },
      'button': {
        'backgroundColor': '#ffe2bf',
        'borderColor': '#ccb599',
        'color': '#998873'
      },
      'text': {
        'primary': {
          'color': '#d6a087'
        },
        'secondary': {
          'color': '#e3a98f'
        }
      },
      'settings': {
        'borderColour': '#7fb2ca',
        'label': {
          'color': '#7fb2ca'
        }
      }
    },
    clive: {
      'answer': {
        'body': {
          'backgroundColor': '#f98f8e',
          'borderColor': '#c77272',
          'color': '#955655'
        },
        'check': {
          'color': '#643939'
        }
      },
      'select': {
        'backgroundColor': '#8ef8f9',
        'borderColor': '#72c6c7',
        'color': '#559595'
      },
      'button': {
        'backgroundColor': '#aaffff',
        'borderColor': '#88cccc',
        'color': '#669999'
      },
      'text': {
        'primary': {
          'color': '#79d3d4'
        },
        'secondary': {
          'color': '#80dfe0'
        }
      },
      'settings': {
        'borderColour': '#c77272',
        'label': {
          'color': '#c77272'
        }
      }
    },
    darwin: {
      'answer': {
        'body': {
          'backgroundColor': '#a1ed80',
          'borderColor': '#81be66',
          'color': '#618e4d'
        },
        'check': {
          'color': '#405f33'
        }
      },
      'select': {
        'backgroundColor': '#cc80ed',
        'borderColor': '#a366be',
        'color': '#7a4d8e'
      },
      'button': {
        'backgroundColor': '#f59aff',
        'borderColor': '#c47bcc',
        'color': '#935c99'
      },
      'text': {
        'primary': {
          'color': '#ad6dc9'
        },
        'secondary': {
          'color': '#b873d5'
        }
      },
      'settings': {
        'borderColour': '#81be66',
        'label': {
          'color': '#81be66'
        }
      }
    },
    houseman: {
      'answer': {
        'body': {
          'backgroundColor': '#76fce8',
          'borderColor': '#5ecaba',
          'color': '#47978b'
        },
        'check': {
          'color': '#2f655d'
        }
      },
      'select': {
        'backgroundColor': '#fc768a',
        'borderColor': '#ca5e6e',
        'color': '#974753'
      },
      'button': {
        'backgroundColor': '#ff8ea6',
        'borderColor': '#cc7285',
        'color': '#995564'
      },
      'text': {
        'primary': {
          'color': '#d66475'
        },
        'secondary': {
          'color': '#e36a7c'
        }
      },
      'settings': {
        'borderColour': '#5ecaba',
        'label': {
          'color': '#5ecaba'
        }
      }
    },
    webb: {
      'answer': {
        'body': {
          'backgroundColor': '#8e9dff',
          'borderColor': '#727ecc',
          'color': '#555e99'
        },
        'check': {
          'color': '#393f66'
        }
      },
      'select': {
        'backgroundColor': '#fff08e',
        'borderColor': '#ccc072',
        'color': '#999055'
      },
      'button': {
        'backgroundColor': '#ffffaa',
        'borderColor': '#cccc88',
        'color': '#999966'
      },
      'text': {
        'primary': {
          'color': '#d9cc79'
        },
        'secondary': {
          'color': '#e6d880'
        }
      },
      'settings': {
        'borderColour': '#727ecc',
        'label': {
          'color': '#727ecc'
        }
      }
    }
  };

  return colours[house];
}

export default colourScheme;
```

Listing 14: Returns a random colour scheme based on the user's house.

11.4 Library Functions

This subsection lists the more advanced functions that are used. Focusing on aspects like scoring, these algorithms form the backbone of the processing that takes place for the quiz.

11.4.1 housePoints.js

```
import isEmpty from 'lodash/lang/isEmpty';

function housePoints({ packet, correct, keys, prop }, state = {}) {
  // If an empty object is passed in, initialise the state by
  // creating an object where each key is a prop, set to 0.
  isEmpty(state) ? keys.forEach(key => state[key] = 0) : null;

  return {
    ...state,
    [packet[prop]]: state[packet[prop]] += (
      // If the answer is correct and they didn't peek.
      packet.answer === correct && packet.peek === false ? 1 :
      // If the answer is correct but they did peek. Otherwise
      // don't give them anything.
      packet.answer === correct && packet.peek === true ? 0.5 : 0
    )
  };
}

export default housePoints;
```

Listing 15: Returns the number of house points earned from an answer.

11.4.2 answerStatistics.js

```
import maxKey from '../utils/maxKey';
import isEmpty from 'lodash/lang/isEmpty';

function answerStatistics({ packet, keys, prop }, state = {}) {
  const initialState = {
    mostCommon: undefined,
    answerFreqs: { A: 0, B: 0,  C: 0, D: 0 }
  };

  // If no state object is passed, loop through the keys and set each
  // one to initialState.
  isEmpty(state) ? keys.forEach(key => state[key] = initialState) : 0;

  // Return a clone of the previous state, but set the answer frequency
  // for the answer the user chose, in their house, to itself + 1.
  const newState = {
    ...state,
    [packet[prop]]: {
      ...state[packet[prop]],
      answerFreqs: {
        ...state[packet[prop]].answerFreqs,
        [packet.answer]: state[packet[prop]].answerFreqs[packet.answer]
             + 1
      }
    }
  };

  // Set the mostCommon property for the packet's house to freq max.
  newState[packet[prop]].mostCommon = maxKey(newState[packet[prop]].
      answerFreqs);

  return newState;
}

export default answerStatistics;

/*
{
  acton: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  },
  baxter: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  },
  clive: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  },
  darwin: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  },
  houseman: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  },
  webb: {
    mostCommon: 0,
    answerFreqs: { A: 0, B: 0, C: 0, D: 0 }
  }
}
*/
```

Listing 16: Returns a tree of the most common answers in each form / year.

11.4.3 constructQuiz.js

```
// Transforms the quiz structure generated by the
// creator into one that can be saved to the database.
function constructQuiz(quiz) {
  return {
    ...quiz.settings,
    startDate: quiz.settings.startDate + ' ' + quiz.settings.startTime,
    categories: quiz.categories.map(category => ({
      ...category,
      questions: quiz.questions.filter(question =>
        question.categoryId === category.id
      ).map(question => ({
        body: question.body,
        answers: quiz.answers.filter(answer =>
          answer.questionId === question.id
        ).map(answer => ({
          body: answer.body,
          correct: answer.correct
        }))
      }))
    }))
  };
}

export default constructQuiz;
```

Listing 17: Transforms the quiz made by the quiz creator into one suitable for consumption by the API.

11.4.4 flattenQuiz.js

```
function flattenQuiz(quiz) {
  let categories = quiz.categories.map((category, id) => ({ body:
      category.body, id })).reduce((a, b) => a.concat(b), []);

  let questions = quiz.categories.map((category, categoryId) =>
    category.questions.map(question => ({ body: question.body,
        categoryId }))
  ).reduce((a, b) => a.concat(b), []);

  let answers = quiz.categories.map(category =>
    category.questions.map((question, questionId) =>
      question.answers.map(answer => ({ body: answer.body, correct:
          answer.correct, questionId }))
    )
  ).reduce((a, b) => a.concat(b), []).reduce((a, b) => a.concat(b), [])
      ;

  return {
    settings: {
      id: quiz._id,
      title: quiz.title,
      startTime: quiz.startDate,
      startDate: quiz.startDate,
      breakLength: quiz.breakLength,
      questionLength: quiz.questionLength
    },
    categories,
    questions,
    answers
  };
}

export default flattenQuiz;
```

Listing 18: Normalises the quiz returned from the API so it can be read by the internal state.

11.5 Action Creators

This subsection lists the action creators. These either construct an object that can be consumed by the reducers, or dispatch actions with side-effects, such as calling the API, which in turn affects an element in state.

11.5.1 actions.js

```
// Colour scheme actions.
export const CHANGE_COLOURS = 'CHANGE_COLOURS';

// Login page actions.
export const CHANGE_HOUSE = 'CHANGE_HOUSE';
export const CHANGE_YEAR = 'CHANGE_YEAR';

export const CHECK_IF_QUIZ_READY = 'CHECK_IF_QUIZ_READY';
export const QUIZ_IN_PROGRESS = 'QUIZ_IN_PROGRESS';
export const QUIZ_IS_SCHEDULED = 'QUIZ_IS_SCHEDULED';
export const NO_QUIZ_READY = 'NO_QUIZ_READY';

export const DELETE_QUIZ = 'DELETE_QUIZ';
export const DELETE_QUIZ_SUCCESS = 'DELETE_QUIZ_SUCCESS';
export const DELETE_QUIZ_FAILURE = 'DELETE_QUIZ_FAILURE';

export const REQUEST_QUIZZES = 'REQUEST_QUIZZES';
export const REQUEST_QUIZZES_FAILURE = 'REQUEST_QUIZZES_FAILURE';
export const RECEIVE_QUIZZES = 'RECEIVE_QUIZZES';

export const LOAD_DEFAULT_QUIZ = 'LOAD_DEFAULT_QUIZ';
export const QUIZ_IS_READY = 'QUIZ_IS_READY';

// Quiz creator actions.
export const UPDATE_ID = 'UPDATE_ID';
export const UPDATE_TITLE = 'UPDATE_TITLE';
export const UPDATE_START_DATE = 'UPDATE_START_DATE';
export const UPDATE_START_TIME = 'UPDATE_START_TIME';
export const UPDATE_QUESTION_LENGTH = 'UPDATE_QUESTION_LENGTH';
export const UPDATE_BREAK_LENGTH = 'UPDATE_BREAK_LENGTH';
export const UPDATE_IS_FINISHED = 'UPDATE_IS_FINISHED';
export const UPDATE_ALL_SETTINGS = 'UPDATE_ALL_SETTINGS';

export const ADD_CATEGORY = 'ADD_CATEGORY';
export const DELETE_CATEGORY = 'DELETE_CATEGORY';
export const EDIT_CATEGORY = 'EDIT_CATEGORY';
export const DELETE_ALL_CATEGORIES = 'DELETE_ALL_CATEGORIES';

export const ADD_QUESTION = 'ADD_QUESTION';
export const DELETE_QUESTION = 'DELETE_QUESTION';
export const EDIT_QUESTION = 'EDIT_QUESTION';
export const DELETE_ALL_QUESTIONS = 'DELETE_ALL_QUESTIONS';

export const ADD_ANSWER = 'ADD_ANSWER';
export const DELETE_ANSWER = 'DELETE_ANSWER';
export const EDIT_ANSWER = 'EDIT_ANSWER';
export const DELETE_ALL_ANSWERS = 'DELETE_ALL_ANSWERS';

export const UPLOAD_QUIZ = 'UPLOAD_QUIZ';

// Play quiz actions
export const JOIN_QUIZ = 'JOIN_QUIZ';
export const BEGIN_QUIZ = 'BEGIN_QUIZ';
export const SHOW_RESULTS = 'SHOW_RESULTS';
export const LEAVE_QUIZ = 'LEAVE_QUIZ';

export const DECREMENT_TIME_LEFT = 'DECREMENT_TIME_LEFT';
export const SHOW_NEXT_QUESTION = 'SHOW_NEXT_QUESTION';
export const MOVE_TO_CATEGORY = 'MOVE_TO_CATEGORY';

export const SELECT_ANSWER = 'SELECT_ANSWER';
export const RECEIVE_ANSWER = 'RECEIVE_ANSWER';
export const RECEIVE_HOUSE_POINTS = 'RECEIVE_HOUSE_POINTS';
```

Listing 19: Application action constants.

11.5.2 LoginActions.js

```
import axios from 'axios';
import flattenQuiz from 'libs/flattenQuiz';
import * as types from 'constants/actions';
import colourScheme from 'utils/colourScheme';
import isQuizReady from 'utils/isQuizReady';
import * as actions from 'actions/CreatorActions';
import { beginQuiz } from 'actions/PlayQuizActions';
import { actions as notifActions } from 're-notif';

const socket = require('socket.io-client')(`http://localhost:5000`);

const { notifSend } = notifActions;
let dismissAfter = 2000;

/**
 * Change the colour scheme of the quiz.
 * @param  {String} house The user's house.
 * @return {Object}       Action dispatcher with colour.
 */
function changeColours(house) {
  return { type: types.CHANGE_COLOURS, colours: colourScheme(house) };
}

/**
 * Updates the user's house and calls new colours.
 * @param  {String} house The new house.
 * @return {Object}       Action dispatcher with house.
 */
export function changeHouse(house) {
  return dispatch => {
    dispatch(changeColours(house));
    dispatch({ type: types.CHANGE_HOUSE, house });
  };
}

/**
 * Update the user's year.
 * @param  {String} year The new year.
 * @return {Object}      Action dispathcer with year.
 */
export function changeYear(year) {
  return { type: types.CHANGE_YEAR, year: parseInt(year) };
}

```

```
/**
 * Sends a DELETE request to /api/quizzes at the
 * current quiz, deleting it from the database.
 * @param  {Number} id ID of quiz to delete.
 * @return {Function}    Call success or delete.
 */
export function deleteQuiz(id) {
  return dispatch => {
    dispatch({ type: types.DELETE_QUIZ });
    return axios.delete(`/api/quizzes/${id}`)
      .then(() => socket.emit(types.DELETE_QUIZ, id))
      .then(() => dispatch(deleteQuizSuccess(id)))
      .catch(() => dispatch(deleteQuizFailure()));
  };
}

/**
 * Shows success message if quiz deleted successfully.
 * @param  {Number} id ID of deleted quiz.
 * @return {Object}    Action dispatcher with year.
 */
function deleteQuizSuccess(id) {
  return dispatch => {
    dispatch(notifSend({
      message: 'Quiz succesfully deleted.',
      kind: 'success',
      dismissAfter
    }));
    dispatch({ type: types.DELETE_QUIZ_SUCCESS, id });
  };
}

/**
 * Shows error message if quiz failed to delete.
 */
function deleteQuizFailure() {
  return dispatch =>
    dispatch(notifSend({
      message: 'Quiz failed to delete.',
      kind: 'danger',
      dismissAfter
    }));
}

/**
 * Removes all categories, questions and
 * answers from the current quiz.
 * @return {Object} Appropriate action dispatchers.
 */
function removeCurrentQuiz() {
  return dispatch => {
    dispatch({ type: types.DELETE_ALL_CATEGORIES });
    dispatch({ type: types.DELETE_ALL_QUESTIONS });
    dispatch({ type: types.DELETE_ALL_ANSWERS });
  };
}

```

```

/**
 * Checks if a quiz is ready.
 *
 * 1. Fetch an array of the quizzes.
 * 2. Begin a loop through the array.
 * 3. If the quiz is due to start within 35 minutes, load it.
 *
 * @return {Object} Action dispatcher to loadQuiz()
 */
export function checkIfQuizReady() {
  return dispatch => {
    socket.emit(types.CHECK_IF_QUIZ_READY);

    socket.on(types.QUIZ_IS_SCHEDULED, (quiz) => {
      dispatch(loadQuiz(quiz, false));
      dispatch(quizIsReady(true));
    });

    socket.on(types.QUIZ_IN_PROGRESS, (quiz) => {
      dispatch(quizIsReady(true));
      dispatch(loadQuiz(quiz, false));
      dispatch(beginQuiz());
    });
  };
}

export function quizIsReady(quizIsReady) {
  return { type: types.QUIZ_IS_READY, quizIsReady };
}

/**
 * Loads a quiz into the store.
 *
 * Quizzes returned from the server have to be
 * normalised into a form that can be loaded
 * into the Redux store.
 *
 * 1. Loop through all the settings and add them to store.
 * 2.
 *
 * @param  {Object}  quiz      The quiz to load.
 * @param  {Boolean} normalise Whether the quiz should be normalised.
 * @return {Object}            Action dispatchers to load the quiz.
 */
export function loadQuiz(quiz, normalise = true ) {
  if (normalise) {
    quiz = flattenQuiz(quiz);
  }

  return dispatch => {
    // Remove the current quiz.
    dispatch(removeCurrentQuiz());
    // Load the quiz's settings.
    dispatch(actions.updateAllSettings(quiz.settings));
    // Map through every category and add them to the quiz.
    quiz.categories.map(category => dispatch(actions.addCategory(
        category.body)));
    // Map through ever question and add them to the quiz.
    quiz.questions.map(question => dispatch(actions.addQuestion(
      question.categoryId,
      question.body
    )));
    // Map through ever answer and add them to the quiz.
    quiz.answers.map(answer => dispatch(actions.addAnswer(
      answer.questionId,
      answer.body,
      answer.correct
    )));
  };
}

/**
 * Request the saved quizzes.
 *
 * If they are received, call receiveQuizzes.
 * Otherwise call requestQuizzesFailure().
 */
export function requestQuizzes() {
  return dispatch => {
    dispatch({ type: types.REQUEST_QUIZZES });
    return axios.get('/api/quizzes')
      .then(response => dispatch(receiveQuizzes(response.data.quizzes))
          )
      .catch(error => dispatch(requestQuizzesFailure()));
  };
}

/**
 * Dispatch request failure action.
 */
function requestQuizzesFailure() {
  return { type: types.REQUEST_QUIZZES_FAILURE };
}

/**
 * Dispatch request success action.
 */
function receiveQuizzes(quizzes) {
  return { type: types.RECEIVE_QUIZZES, quizzes };
}
```

Listing 20: Login action creators.

11.5.3 CreatorActions.js

```
import axios from 'axios';
import * as types from 'constants/actions';
import { actions as notifActions } from 're-notif';

const socket = require('socket.io-client')(`http://localhost:5000`);

export function updateId(id) {
  return { type: types.UPDATE_ID, id };
}

export function updateTitle(title) {
  return { type: types.UPDATE_TITLE, title };
}

export function updateStartDate(startDate) {
  return { type: types.UPDATE_START_DATE, startDate };
}

export function updateStartTime(startTime) {
  return { type: types.UPDATE_START_TIME, startTime };
}

export function updateQuestionLength(questionLength) {
  return { type: types.UPDATE_QUESTION_LENGTH, questionLength };
}

export function updateBreakLength(breakLength) {
  return { type: types.UPDATE_BREAK_LENGTH, breakLength };
}

export function updateIsFinished(isFinished) {
  return { type: types.UPDATE_IS_FINISHED, isFinished };
}

export function updateAllSettings(settings) {
  return { type: types.UPDATE_ALL_SETTINGS, settings };
}

export function addCategory(body) {
  return { type: types.ADD_CATEGORY, body };
}

export function editCategory(id, body) {
  return { type: types.EDIT_CATEGORY, id, body };
}

export function deleteCategory(id) {
  return { type: types.DELETE_CATEGORY, id };
}

export function addQuestion(categoryId, body) {
  return { type: types.ADD_QUESTION, categoryId, body };
}

export function editQuestion(id, body) {
  return { type: types.EDIT_QUESTION, id, body };
}

export function deleteQuestion(id) {
  return { type: types.DELETE_QUESTION, id };
}

export function addAnswer(questionId, body, correct) {
  return { type: types.ADD_ANSWER, questionId, body, correct };
}
```

```

export function editAnswer(id, body, correct) {
  return { type: types.EDIT_ANSWER, id, body, correct };
}

export function deleteAnswer(id) {
  return { type: types.DELETE_ANSWER, id };
}

// Quiz saving async actions.
const { notifSend } = notifActions;
let dismissAfter = 2000;

/**
 * Updates a quiz that's already in the database.
 * @param  {Object} quiz Quiz to save.
 */
function updateQuiz(quiz) {
  return dispatch =>
    axios.put(`/api/quizzes/${quiz.id}`, quiz)
      .then(() => {
        console.log(quiz);
        socket.emit(types.UPLOAD_QUIZ, quiz);
      })
      .then(response => dispatch(notifSend({
        message: 'Quiz successfully updated!',
        kind: 'success',
        dismissAfter
      })))
      .catch(error => dispatch(notifSend({
        message: 'Quiz failed to update.',
        kind: 'danger',
        dismissAfter
      })));
}

/**
 * Saves a new quiz to the database.
 * @param  {Object} quiz Quiz to save.
 */
function saveQuiz(quiz) {
  return dispatch =>
    axios.post('/api/quizzes', quiz)
      .then(response => dispatch(updateId(response.data.quiz._id)))
      .then(response => socket.emit(types.UPLOAD_QUIZ, { ...quiz, _id:
          response.id }))
      .then(response => dispatch(notifSend({
        message: 'Quiz successfully saved!',
        kind: 'success',
        dismissAfter
      })))
      .catch(error => dispatch(notifSend({
        message: 'Quiz failed to save.',
        kind: 'danger',
        dismissAfter
      })));
}
```

```

/**
 * Checks if a new quiz should be created,
 * or an existing one updated.
 * @param  {Object} quiz The quiz to check.
 */
export function saveOrUpdateQuiz(quiz) {
  return dispatch => {
    if (quiz.id.length) {
      dispatch(updateQuiz(quiz));
    } else {
      dispatch(saveQuiz(quiz));
    }
  };
}
```

Listing 21: Quiz creator action creators.

11.6 Reducers

This subsections lists the application's state reducers.

11.6.1 configureStore.js

```
import { applyMiddleware, createStore, compose } from 'redux';
import DevTools from 'containers/DevTools';
import rootReducer from '../reducers';
import thunk from 'redux-thunk';

function configureStore() {
  const store = compose(
    applyMiddleware(thunk),
    DevTools.instrument()
  )(createStore)(rootReducer);

  if (module.hot) {
    module.hot.accept('../reducers', () => {
      const nextRootReducer = require('../reducers');
      store.replaceReducer(nextRootReducer);
    });
  }

  return store;
}

export default configureStore;
```

Listing 22: Creates the store containing application state and initialises middleware.

11.6.2 userReducer.js

```
import * as types from 'constants/actions';

```

```
let defaultState = {
  house: 'acton',
  year: 7,
  quizzes: [],
  quizIsReady: false,
  requestingQuizzes: false,
  requestingQuizzesFailed: false
};

function user(state = defaultState, action) {
  switch (action.type) {
    case types.CHANGE_HOUSE:
      return { ...state, house: action.house };

    case types.CHANGE_YEAR:
      return { ...state, year: action.year };

    case types.REQUEST_QUIZZES:
      return {
        ...state,
        requestingQuizzes: true,
        requestingQuizzesFailed: false
      };

    case types.REQUEST_QUIZZES_FAILURE:
      return {
        ...state,
        requestingQuizzes: false,
        requestingQuizzesFailed: true
      };

    case types.RECEIVE_QUIZZES:
      return {
        ...state,
        quizzes: action.quizzes,
        requestingQuizzes: false
      };

    case types.DELETE_QUIZ_SUCCESS:
      return {
        ...state,
        quizzes: state.quizzes.filter(quiz =>
          quiz._id !== action.id
        )
      };

    case types.QUIZ_IS_READY:
      return { ...state, quizIsReady: action.quizIsReady };

    default:
      return state;
  }
}

export default user;
```

Listing 23: User reducer controlling creation state.

11.6.3 quizReducer.js

```
import { combineReducers } from 'redux';
import * as types from 'constants/actions';
import nextBiggest from 'utils/nextBiggest';
import isFinite from 'lodash/lang/isFinite';

// The default data that the quiz will show upon
// initialisation.
let defaultState = {
  settings: {},
  categories: [],
  questions: [],
  answers: []
};

function settings(state = defaultState.settings, action) {
  switch (action.type) {
    case types.UPDATE_ID:
      return { ...state, id: action.id };

    case types.UPDATE_TITLE:
      return { ...state, title: action.title };

    case types.UPDATE_START_DATE:
      return { ...state, startDate: action.startDate };

    case types.UPDATE_START_TIME:
      return { ...state, startTime: action.startTime };

    case types.UPDATE_QUESTION_LENGTH:
      return { ...state, questionLength: action.questionLength };

    case types.UPDATE_BREAK_LENGTH:
      return { ...state, breakLength: action.breakLength };

    case types.UPDATE_IS_FINISHED:
      return { ...state, isFinished: action.isFinished };

    case types.UPDATE_ALL_SETTINGS:
      return { ...state, ...action.settings };

    default:
      return state;
  }
}

function categories(state = defaultState.categories, action) {
  switch (action.type) {
    case types.ADD_CATEGORY:
      return [...state, {
        id: isFinite(nextBiggest(state)) ? nextBiggest(state) : 0,
        body: action.body
      }];

    case types.EDIT_CATEGORY:
      return state.map(category =>
        category.id === action.id ?
          { ...category, body: action.body } :
          category
      );

    case types.DELETE_CATEGORY:
      return state.filter(x => x.id !== action.id);

    case types.DELETE_ALL_CATEGORIES:
      return state.filter(x => x.id === -1);

    default:
      return state;
  }
}

function questions(state = defaultState.questions, action) {
  switch (action.type) {
    case types.ADD_QUESTION:
      return [...state, {
        body: action.body,
        id: isFinite(nextBiggest(state)) ? nextBiggest(state) : 0,
        categoryId: action.categoryId
      }];

    case types.EDIT_QUESTION:
      return state.map(question =>
        question.id === action.id ?
          { ...question, body: action.body } :
          question
      );

    case types.DELETE_QUESTION:
      return state.filter(x => x.id !== action.id);

    case types.DELETE_ALL_QUESTIONS:
      return state.filter(x => x.id === -1);

    default:
      return state;
  }
}

function answers(state = defaultState.answers, action) {
  switch (action.type) {
    case types.ADD_ANSWER:
      return [...state, {
        body: action.body,
        correct: action.correct,
        id: isFinite(nextBiggest(state)) ? nextBiggest(state) : 0,
        questionId: action.questionId
      }];

    case types.EDIT_ANSWER:
      return state.map(answer =>
        answer.id === action.id ?
          { ...answer, body: action.body, correct: action.correct } :
```

```
          answer
      );

    case types.DELETE_ANSWER:
      return state.filter(x => x.id !== action.id);

    case types.DELETE_ALL_ANSWERS:
      return state.filter(x => x.id === -1);

    default:
      return state;
  }
}

export default combineReducers({
  settings,
  categories,
  questions,
  answers
});
```

Listing 24: Quiz reducer controlling creation state.

11.7 Sockets

This subsection lists the socket controllers used in the application. In order to allow the different clients (players) to communicate with one another, a concept known as web sockets is utilised. Web sockets allow a continual, bi-directional connection to be set up between each client and the server, facilitating the broadcasting and receiving of messages. In this manner, all of the multiplayer components to the quiz are controlled through these files.

11.7.1 quizSockets.js

```
import * as types from '../constants/actions';
import axios from 'axios';
import moment from 'moment';
import config from '../config';
import schedule from 'node-schedule';
import flattenQuiz from '../libs/flattenQuiz';
import isQuizReady from '../utils/isQuizReady';
import calculateHousePoints from '../libs/housePoints';
import calculateAnswerStatistics from '../libs/answerStatistics';

let util = require('util');

async function quizSockets(server) {
  let io = require('socket.io').listen(server);
  let quizAddress = `http://localhost:${config.port}/api/quizzes`;

  // Get initial set of quizzes on server start and schedule them.
  (await axios.get(quizAddress)).data.quizzes
            .map(x => flattenQuiz(x))
```

```
              .forEach(quiz => scheduleQuiz(quiz));

let jobs = {};
let currentQuiz = {};
let quizStatus = types.NO_QUIZ_READY;
let players = [];
let answers = {};
let housePoints = {};

io.on('connection', (socket) => {
  // If they upload a new quiz, ensure the server finds
  // and schedules it automatically.
  socket.on(types.UPLOAD_QUIZ, (quiz) => {
    // console.log(quiz);
    quiz = flattenQuiz(quiz);
    scheduleQuiz(quiz);
  });

  socket.on(types.DELETE_QUIZ, (id) => {
    if (jobs[id]) {
      // Cancel all the jobs related to that quiz.
      jobs[id].forEach(job => job.cancel());
      // Delete the quiz entry.
      delete jobs[id];
      quizStatus = types.NO_QUIZ_READY;
    }
  });

  // Inform the client about the current status of the quiz.
  socket.on(types.CHECK_IF_QUIZ_READY, () => socket.emit(quizStatus,
      currentQuiz));
  // Add the player to the array of connections.
  socket.on(types.JOIN_QUIZ, (form) => players.push({ socket, form })
      );

  socket.on(types.SELECT_ANSWER, (packet) => {
    const houses = ['acton', 'baxter', 'clive', 'darwin', 'houseman',
        'webb'];
    // Calculate the answer for each house.
    answers[packet.questionId] = calculateAnswerStatistics({
      packet,
      keys: houses,
      prop: 'house'
    }, answers[packet.questionId]);

    // Calculate the current house points.
    housePoints = calculateHousePoints({
      packet,
      correct: 'A',
      keys: houses,
      prop: 'house'
    }, housePoints);

    io.emit(types.RECEIVE_ANSWER, housePoints);
  });

  socket.on('disconnect', () => {
    // Remove the player from the array of connections.
    players = players.filter(player => player.socket.id !== socket.id
        );
    io.emit(types.REMOVE_PLAYER, socket.id );
  });
});

function scheduleQuiz(quiz) {
  const quizStart = new Date(quiz.settings.startDate);
  console.log(quiz);
  if (moment(quizStart).isAfter(moment())) {
    let countdownStart = moment(quizStart).subtract(20, 'minutes')._d
        ;
    let totalQuizDuration = quiz.questions.length * quiz.settings.
        questionLength;
    let showResults = moment(quizStart).add(totalQuizDuration + 1000,
         'milliseconds')._d;
    let quizFinish = moment(quizStart).add(totalQuizDuration + 9000,
        'milliseconds')._d;

    jobs[quiz.settings._id] = [
      schedule.scheduleJob(countdownStart, () => {
        currentQuiz = quiz;
        quizStatus = types.QUIZ_IS_SCHEDULED;
      }),

      schedule.scheduleJob(quizStart, () => {
        quizStatus = types.QUIZ_IN_PROGRESS;
        io.emit(types.BEGIN_QUIZ);
      }),

      schedule.scheduleJob(showResults, () => {
        quizStatus = types.NO_QUIZ_READY;
        io.emit(types.SHOW_RESULTS, housePoints);
      }),

      schedule.scheduleJob(quizFinish, () => {
        answers = {};
      })
    ];

    questionTimer(quiz);
  }
}

function questionTimer(quiz) {
  const questionLength = quiz.settings.questionLength;
  const quizStart = new Date(quiz.settings.startDate);
  console.log(quiz.settings.id);
  console.log(jobs);
  quiz.questions.forEach((question, i) => {
    const changeQuestion = moment(quizStart).add(questionLength * i,
        'milliseconds')._d;

    jobs[quiz.settings._id].push(
      schedule.scheduleJob(changeQuestion, () => {
        io.emit(types.SHOW_NEXT_QUESTION, i);
```

```
          countdown(questionLength);
        })
      );
    });
  }

  function countdown(questionLength) {
    let timeLeft = questionLength;

    io.emit(types.DECREMENT_TIME_LEFT, timeLeft);

    let countdown = setInterval(() => {
      if (timeLeft <= 1000) clearInterval(countdown);
      console.log(timeLeft);
      timeLeft -= 1000;
      io.emit(types.DECREMENT_TIME_LEFT, timeLeft);
    }, 1000);
  }
}

export default quizSockets;
```

Listing 25: Main socket controller.

11.7.2 quizEvents.js

```
import * as types from 'constants/actions';
import * as actions from 'actions/PlayQuizActions';

function quizEvents(dispatch, historyProp) {
  const socket = require('socket.io-client')(`http://localhost:5000`);

  socket.on(types.BEGIN_QUIZ, () =>
    dispatch(actions.beginQuiz())
  );

  socket.on(types.SHOW_RESULTS, (housePoints) =>
    dispatch(actions.showResults(historyProp, housePoints))
  );

  socket.on(types.LEAVE_QUIZ, () =>
    dispatch(actions.leaveQuiz(historyProp))
  );

  socket.on(types.ADD_PLAYER, (players) =>
    dispatch(actions.addPlayer(players))
  );

  socket.on(types.REMOVE_PLAYER, (socketId) =>
    dispatch(actions.removePlayer(socketId))
  );

  socket.on(types.DECREMENT_TIME_LEFT, (timeLeft) =>
    dispatch(actions.decrementTimeLeft(timeLeft))
  );

  socket.on(types.SHOW_NEXT_QUESTION, (questionId) =>
    dispatch(actions.showNextQuestion(questionId))
  );

  // socket.on(types.RECEIVE_ANSWER, (answer) =>
  //   dispatch(actions.receiveAnswer(answer))
  // );
}

export default quizEvents;
```

Listing 26: Controls interface based on socket emissions.

11.8 Components

This subsections lists the application's components. Components make up the vast majority of the codebase, as they combine both markup - a description of what should be shown - and logic. The first three components are more advanced, as they interact directly with the applications's state. Other components are less intelligent, and rely on properties passed down from the more advanced components to perform actions and read data.

11.8.1 LoginContainer.js

```
import { connect } from 'react-redux';
import React, { Component } from 'react';
import first from 'lodash/array/first';
import defaultQuiz from 'utils/defaultQuiz';
import backgroundStyle from 'utils/backgroundStyle';
import LoginForm from 'components/LoginForm';
import { joinQuiz } from 'actions/PlayQuizActions';
import LoadQuizPanel from 'components/LoadQuizPanel';
import {
  changeHouse,
  changeYear,
  loadQuiz,
  deleteQuiz,
  requestQuizzes,
  checkIfQuizReady
} from 'actions/LoginActions';

const houses = ['acton', 'baxter', 'clive', 'darwin', 'houseman', 'webb
    '];
const years = [7, 8, 9, 10, 11];

class LoginContainer extends Component {
  constructor(props) {
    super(props);

    this.state = { panelIsOpen: false };
  }

  componentDidMount() {
    this.props.dispatch(checkIfQuizReady());
  }

  changeYear = (e) => {
    this.props.dispatch(changeYear(e.target.value));
  };

  changeHouse = (e) => {
    this.props.dispatch(changeHouse(e.target.value));
  };

  newQuiz = () => {
    this.props.dispatch(loadQuiz(defaultQuiz, false));
    this.props.history.pushState('create', '/create');
  };

  playQuiz = () => {
    this.props.dispatch(joinQuiz(this.props.user));
    this.props.history.pushState('play', '/play');
  };

  deleteQuiz = (quizId) => {
    this.props.dispatch(deleteQuiz(quizId));
  };

  loadQuiz = (quizId) => {
    const quiz = this.props.user.quizzes.find(quiz => quiz._id ===
        quizId);
    this.props.dispatch(loadQuiz(quiz));
    this.closeQuizSelect();
    this.props.history.pushState('create', '/create');
  };

  openQuizSelect = () => {
    this.props.dispatch(requestQuizzes());
    this.setState({ panelIsOpen: true });
  };

  closeQuizSelect = () => {
    this.setState({ panelIsOpen: false });
  };

  render() {
    return (
      <div style={backgroundStyle(this.props.user.house)}>
        <div className="container">
          <LoginForm changeHouse={this.changeHouse}
                     changeYear={this.changeYear}
                     colours={this.props.colours}
                     house={this.props.user.house}
                     playQuiz={this.playQuiz}
                     quizIsReady={this.props.user.quizIsReady}
                     newQuiz={this.newQuiz}
                     openQuizSelect={this.openQuizSelect}
                     houses={houses}
                     years={years} />

          <LoadQuizPanel colours={this.props.colours}
```

```
                          panelIsOpen={this.state.panelIsOpen}
                          loadQuiz={this.loadQuiz}
                          deleteQuiz={this.deleteQuiz}
                          closePanel={this.closeQuizSelect}
                          requestingQuizzes={this.props.user.
                              requestingQuizzes}
                          requestingQuizzesFailed={this.props.user.
                              requestingQuizzesFailed}
                          quizzes={this.props.user.quizzes} />
        </div>
      </div>
    );
  }
}

function mapStateToProps(state) {
  return {
    user: state.user,
    colours: state.colours
  };
}

export default connect(
  mapStateToProps
)(LoginContainer);
```

Listing 27: Advanced component controlling the login screen.

11.8.2 LoginForm.js

```
import React, { Component, PropTypes } from 'react';
import Button from 'components/Button';
import Select from 'components/Select';

class LoginForm extends Component {
  static propTypes = {
    changeHouse: PropTypes.func.isRequired,
    changeYear: PropTypes.func.isRequired,
    houses: PropTypes.array.isRequired,
    newQuiz: PropTypes.func.isRequired,
    playQuiz: PropTypes.func.isRequired,
    quizIsReady: PropTypes.bool.isRequired,
    years: PropTypes.array.isRequired
  };

  render() {
    return (
      <div className="loginForm">
        <div className="animated bounceInDown">
          <h1 style={this.props.colours.text.primary} className="centre
              ">
            Priory School Quiz
          </h1>
          <p style={this.props.colours.text.secondary} className="
              centre">
            Hello! Choose your house and year to get started!
          </p>
        </div>

        <form className="animated bounceInUp">
          <Select changeEvent={this.props.changeHouse}
                  customClass="hover"
                  colours={this.props.colours.select}
                  options={this.props.houses}
                  placeholder="I belong to..."
                  size="full" />

          <Select changeEvent={this.props.changeYear}
                  customClass="hover"
                  colours={this.props.colours.select}
                  options={this.props.years}
                  placeholder="And I'm in year..."
                  prefix="Year"
                  size="full" />

          {this.props.quizIsReady ?
            // If there is a quiz scheduled in the next 30 minutes,
                display a
            // button to join the room. Otherwise show button to create
                 a quiz.
            <Button colours={this.props.colours.button}
                    customClass="login-button join-quiz-button"
                    text="Join the quiz!"
                    clickEvent={this.props.playQuiz} /> :

            <ul>
              <Button customClass="login-button"
                      colours={this.props.colours.button}
                      text="Create a new quiz!"
                      clickEvent={this.props.newQuiz} />

              <Button customClass="login-button"
                      colours={this.props.colours.button}
                      text="Load a quiz!"
                      clickEvent={this.props.openQuizSelect} />
            </ul>
          }
        </form>
      </div>
    );
  }
}

export default LoginForm;
```

Listing 28: Login form component.

11.8.3 PlayQuizContainer.js

```
import { connect } from 'react-redux';
import React, { Component } from 'react';
import * as types from 'constants/actions';
import quizEvents from '../sockets/quizEvents';
import PlayQuiz from 'components/play/PlayQuiz';
import Countdown from 'components/play/Countdown';
import * as actions from 'actions/PlayQuizActions';

class PlayQuizContainer extends Component {
  componentDidMount() {
    quizEvents(this.props.dispatch, this.props.history);
  }

  selectAnswer = (answer) => {
    if (!this.props.currentQuiz.selectedAnswer) {
      const packet = {
        year: this.props.user.year,
        house: this.props.user.house,
        answer: ['A', 'B', 'C', 'D'][answer],
        questionId: this.props.currentQuiz.currentQuestion
      };

      this.props.dispatch(actions.selectAnswer(packet));
    }
  };

  render() {
    const questionId = this.props.currentQuiz.currentQuestion;
    const categoryId = this.props.currentQuiz.currentCategory;

    let quizInfo = {
      totalQuestions: this.props.quiz.questions.length,
      questionLength: this.props.quiz.settings.questionLength,
      category: {
        id: categoryId,
        body: this.props.quiz.categories[categoryId].body,
        currentPosition: this.props.quiz.questions
                           .filter(x => x.categoryId === categoryId)
                           .map(x => x.id)
                           .indexOf(questionId) + 1,
        length: this.props.quiz.questions.filter(x => x.categoryId ===
            categoryId).length
      },
      question: {
        id: questionId
      }
    };

    return (
      this.props.currentQuiz.inProgress ?

      <PlayQuiz question={{...this.props.quiz.questions[questionId]}}
                answers={this.props.quiz.answers.filter(x => x.
                    questionId === questionId)}
                colours={this.props.colours}
                quizInfo={quizInfo}
                timeLeft={this.props.currentQuiz.timeLeft}
                selectAnswer={this.selectAnswer} /> :

      <Countdown startTime={this.props.quiz.settings.startTime} colours
          ={this.props.colours.text} />
    );
  }
}

function mapStateToProps(state) {
  return {
    user: state.user,
    quiz: state.quiz,
    colours: state.colours,
    currentQuiz: state.currentQuiz
  };
}

export default connect(
  mapStateToProps
)(PlayQuizContainer);
```

Listing 29: Advanced component controlling the play quiz screen.

11.8.4 PlayQuiz.js

```
import React, { Component, PropTypes } from 'react';
import Button from 'components/Button';
import QuizAnswer from 'components/play/QuizAnswer';
import QuestionTimer from 'components/play/QuestionTimer';

class PlayQuiz extends Component {
  static propTypes = {
    answers: PropTypes.array.isRequired,
    colours: PropTypes.object.isRequired,
    question: PropTypes.object.isRequired,
    quizInfo: PropTypes.object.isRequired,
    selectAnswer: PropTypes.func.isRequired,
    timeLeft: PropTypes.number.isRequired
  };

  constructor(props) {
    super(props);
  }

  render() {
    // Used to prefix each answer.
    let letters = ['A', 'B', 'C', 'D'];
    let quizInfo = this.props.quizInfo;

    return (
      <div style={{ padding: '10px' }}>
        <h1 className="question-title" style={this.props.colours.text.
            primary}>
          {this.props.question.body}
        </h1>

        <h3 style={this.props.colours.text.secondary}>
        {`Question ${quizInfo.category.currentPosition}
          out of ${quizInfo.category.length}
          in the ${quizInfo.category.body} category •
          Question ${quizInfo.question.id + 1} out of ${quizInfo.
              totalQuestions} overall`}
        </h3>

        <QuestionTimer colours={this.props.colours.answer.body}
                       questionLength={quizInfo.questionLength}
                       timeLeft={this.props.timeLeft} />

        <ul>
          {this.props.answers.map((answer, i) =>
            <QuizAnswer key={i}
                        body={`${letters[i]}. ${answer.body}`}
                        colours={this.props.colours.select}
                        correct={answer.correct}
                        selectAnswer={this.props.selectAnswer.bind(this
                            , i)} />
          )}
        </ul>
      </div>
    );
  }
}

export default PlayQuiz;
```

Listing 30: Play quiz component.

11.8.5 Countdown.js

```
import React, { Component, PropTypes } from 'react';
import moment from 'moment';

class Countdown extends Component {
  static propTypes = {
    startTime: PropTypes.string.isRequired
  };

  constructor(props) {
    super(props);

    this.state = { timeLeft: moment().to(this.props.startTime) };
  }

  componentDidMount() {
    this.timer = setInterval(() => this.tick(), 1000);
  }

  componentWillUnmount() {
    clearTimeout(this.timer);
  }

  tick = () => {
    this.setState({ timeLeft: moment().to(this.props.startTime) });
  };

  render() {
    return (
      <div className="countdown">
        <h1 style={this.props.colours.primary}>You're a bit early!</h1>
        <h2 style={this.props.colours.secondary}>The quiz will begin {
            this.state.timeLeft}...</h2>
      </div>
    );
  }
}

export default Countdown;
```

Listing 31: Displays the time until the quiz is due to start.

11.8.6 QuestionTimer.js

```
import React, { Component, PropTypes } from 'react';
import { SHOW_NEXT_QUESTION } from 'constants/actions';

class QuestionTimer extends Component {
  static propTypes = {
    colours: PropTypes.object.isRequired,
    questionLength: PropTypes.number.isRequired,
    timeLeft: PropTypes.number.isRequired
  };

  constructor(props) {
    super(props);

    this.state = { timeLeft: this.props.timeLeft };
  }

  componentWillReceiveProps(props) {
    const receivedTime = props.timeLeft;

    let count = receivedTime;

    let countdown = setInterval(() => {
      if (count <= receivedTime - 1000) clearInterval(countdown);
      this.setState({ timeLeft: count });
      count -= 10;
    }, 10);
  }

  render() {
    let width = (this.state.timeLeft / this.props.questionLength) * 100
         + '%';

    return (
      <div className="question-timer"
           style={{ ...this.props.colours, width }}>
      </div>
    );
  }
}

export default QuestionTimer;
```

Listing 32: Counts down the time until the next question.

11.8.7 CreateQuizContainer.js

```
import React, { Component, PropTypes } from 'react';
import * as actions from 'actions/CreatorActions';
import { connect } from 'react-redux';

import findIndex from 'lodash/array/findIndex';
import last from 'lodash/array/last';
import moment from 'moment';

import backgroundStyle from 'utils/backgroundStyle';
import constructQuiz from 'libs/constructQuiz';
import CreateQuiz from 'components/CreateQuiz';

const { notifSend } = notifActions;

import { Notifs, actions as notifActions } from 're-notif';
import 're-notif/lib/re-notif.css';

class CreateQuizContainer extends Component {
  constructor(props) {
    super(props);

    this.state = { currentQuestion: 0 };
  }

  /**
   * Adds a new category to the quiz.
   */
  addCategory = () => {
    let categoryBody = prompt('Category name:');

    if (categoryBody.length) {
      this.props.dispatch(actions.addCategory(categoryBody));
      // Generate a first question for the category.
      this.addQuestion(this.props.quiz.categories.length);
    }
  };

  /**
   * Edits the name of a category.
   * @param  {Element} e Information on the category.
   */
  editCategory = (id) => {
    let currentBody = this.props.quiz.categories.find(x => x.id === id)
        .body;
    let newBody = prompt('Enter a new category body:');

    if (newBody.length && newBody !== currentBody) {
      this.props.dispatch(actions.editCategory(id, newBody));
    }
  };

  /**
   * Deletes a category and all it's questions.
   * @param  {Number} id ID of category to delete.
   */
```

```
deleteCategory = (id) => {
  // Get all the questions in the category.
  this.props.quiz.questions
    .filter(question => question.categoryId === id)
    .map(question => this.deleteQuestion(question.id));

  // Delete the actual category entry.
  this.props.dispatch(actions.deleteCategory(id));
};

/**
 * Adds a question to the quiz, into the currently
 * selected category.
 * @param {Number} categoryId The ID of the current category.
 */
addQuestion = (categoryId) => {
  const categoryBody = this.props.quiz.categories.find(x => x.id ===
      categoryId).body;

  this.props.dispatch(actions.addQuestion(
    categoryId,
    `New question in the ${categoryBody} category`
  ));

  // The ID of the newly created question.
  let newId = last(this.props.quiz.questions).id + 1;
  // Generate four sample answers for the question.
  for (let i = 0; i <= 3; i++) {
    this.props.dispatch(actions.addAnswer(
      newId,
      'Tap to edit me!',
      i === 0 ? true : false // Set the first answer as correct.
    ));
  }
  // Switch the interface over to the new question.
  this.changeQuestion(newId);
};

/**
 * Deletes a question from the quiz.
 */
deleteQuestion = () => {
  // Ensure they are not deleting the only question.
  if (this.props.quiz.questions.length > 1) {
    // The id of the question to be deleted.
    let deleteId = this.state.currentQuestion;
    // The index in the questions array of the question to be deleted
      .
    let deleteIndex = findIndex(this.props.quiz.questions, x => x.id
        === deleteId);

    // Create an array containing all the answers associated
    // with the question to be deleted and then delete them.
    this.props.quiz.answers
      .filter(x => x.questionId === deleteId)
      .map(x => this.props.dispatch(actions.deleteAnswer(x.id)));

    let nextQuestionIndex = findIndex(this.props.quiz.questions, x => x.id === deleteIndex - 1);
    let nextQuestionId = this.props.quiz.questions[nextQuestionIndex].id;

    // Update the view to the previous question.
    // this.changeQuestion(deleteIndex - 1);
    this.changeQuestion(nextQuestionId);

    // Delete the actual question entry.
    this.props.dispatch(actions.deleteQuestion(deleteId));
  }
};

/**
 * Edits a question's body.
 * @param  {Number} id   ID of the question to edit.
 * @param  {String} body New question body.
 */
editQuestion = (id, body) => {
  this.props.dispatch(actions.editQuestion(id, body));
};

/**
 * Marks a possible quiz answer as correct.
 * @param  {Number} answerId   ID of the answer to make correct.
 * @param  {Number} questionId ID of the question the answer belongs to.
 * @param  {String} body       The body of the answer.
 * @param  {Boolean} correct   Whether the answer is correct.
 */
markCorrect = (answerId, questionId, body, correct) => {
  // Get the question that's currently marked as correct.
  const currentlyCorrect = this.props.quiz.answers.find(x =>
    x.correct === true && x.questionId === questionId
  );

  // Ensure they don't try to mark the current answer.
  if (answerId !== currentlyCorrect.id) {
    // Update the state tree to make the currently correct answer incorrect.
    this.editAnswer(currentlyCorrect.id, currentlyCorrect.body, false);
    // Update the state tree to mark the newly selected answer as correct.
    this.editAnswer(answerId, body, correct);
    // Update the container to state to cause a re-render of the interface.
    this.setState({ currentQuestion: questionId });
  }
};

/**
 * Edits a question answer with new properties.
 * @param  {Number} id      ID of answer to edit/
 * @param  {String} body    New body of answer.
 * @param  {Bool} correct   New correct status of answer.
 */
editAnswer = (id, body, correct) => {
  this.props.dispatch(actions.editAnswer(id, body, correct));
};

/**
 * Called whenever a new question needs to be displayed.
 * Updates the state containing the current question ID,
 * making the other components update to show the new question.
 * @param  {Number|String} e The ID of the new question.
 */
changeQuestion = (e) => {
  // Check if the question is being changed manually or
  // by an override, such as adding a new question.
  const id = typeof e === 'number' ? e : e.target.value;
  this.setState({ currentQuestion: parseInt(id) });
};

finishQuiz = () => {
  const finished = confirm(`Do you want to set this quiz to play on $
      {moment(this.props.quiz.startDate).format('dddd Do MMMM')}, at
      ${moment(this.props.quiz.startTime).format('HH:mm A')}?`);

  if (finished) {
    this.props.dispatch(actions.updateIsFinished(true));
  }

  let quiz = constructQuiz(this.props.quiz);
  this.props.dispatch(actions.saveOrUpdateQuiz(quiz));
};

leaveQuiz = () => {
  this.props.history.pushState('/', '/');
};

/**
 * Called whenever a quiz settings panel property is modified.
 * @param  {String} setting The setting that has been modified.
 * @param  {String|Numbe}r value   The new setting data.
 */
saveSettings = (settings) => {
  // Loop through every setting key.
  for (let setting in settings) {
    // Get the value set in the settings panel.
    let value = settings[setting];
    // If they've made any changes compared to the value we have
    // stored in the state tree.
    if (value.length && value !== this.props.quiz.settings[setting])
        {
      // Call the correct action to update the state.
      switch (setting) {
        case 'title':
          this.props.dispatch(actions.updateTitle(value));
          break;
        case 'startDate':
          this.props.dispatch(actions.updateStartDate(value));
          break;
        case 'startTime':
          this.props.dispatch(actions.updateStartTime(value));
          break;
        case 'questionLength':
          this.props.dispatch(actions.updateQuestionLength(parseInt(
              value * 1000)));
          break;
        case 'breakLength':
          this.props.dispatch(actions.updateBreakLength(parseInt(
              value * 60000)));
          break;
        default:
          return value;
      }
    }
  }

  this.props.dispatch(notifSend({
    message: 'Settings successfully updated!',
    kind: 'success',
    dismissAfter: 2000
  }));
};

render() {
  const id = this.state.currentQuestion;

  const currentQuestion = {
    id,
    body: this.props.quiz.questions.find(x => x.id === id).body,
    answers: this.props.quiz.answers.filter(x => x.questionId === id)
  };

  const notificationThemes = {
    defaultClasses: 'notification',
    successClasses: 'notification-success',
    dangerClasses: 'notification-danger'
  };

  return (
    <div style={backgroundStyle(this.props.user.house)} className="
        animated fadeIn">
      <CreateQuiz addQuestion={this.addQuestion}
                  addCategory={this.addCategory}
                  categories={this.props.quiz.categories}
                  changeQuestion={this.changeQuestion}
                  colours={this.props.colours}
                  currentQuestion={currentQuestion}
                  deleteQuestion={this.deleteQuestion}
                  deleteCategory={this.deleteCategory}
                  editAnswer={this.editAnswer}
                  editCategory={this.editCategory}
                  editQuestion={this.editQuestion}
                  saveSettings={this.saveSettings}
                  finishQuiz={this.finishQuiz}
                  house={this.props.user.house}
                  leaveQuiz={this.leaveQuiz}
                  markCorrect={this.markCorrect}
                  quizSettings={this.props.quiz.settings}
                  questions={this.props.quiz.questions} />
      <Notifs theme={notificationThemes} />
    </div>
  );
}
}

function mapStateToProps(state) {
return {
  user: state.user,
  quiz: state.quiz,
  colours: state.colours
};
}

export default connect(
mapStateToProps,
)(CreateQuizContainer);
```

Listing 33: Advanced component controlling the create quiz screen.

11.8.8 CreateQuiz.js

```
import React, { Component, PropTypes } from 'react';
import findIndex from 'lodash/array/findIndex';
import QuizSettingsPanel from 'components/QuizSettingsPanel';
import EditableText from 'components/EditableText';
import Button from 'components/Button';
import Select from 'components/Select';
import Answer from 'components/Answer';

class CreateQuiz extends Component {
  static propTypes = {
    addCategory: PropTypes.func.isRequired,
    addQuestion: PropTypes.func.isRequired,
    categories: PropTypes.array.isRequired,
    changeQuestion: PropTypes.func.isRequired,
    colours: PropTypes.object.isRequired,
    currentQuestion: PropTypes.object.isRequired,
    deleteCategory: PropTypes.func.isRequired,
    deleteQuestion: PropTypes.func.isRequired,
    editAnswer: PropTypes.func.isRequired,
    editCategory: PropTypes.func.isRequired,
    editQuestion: PropTypes.func.isRequired,
    finishQuiz: PropTypes.func.isRequired,
    house: PropTypes.string.isRequired,
    leaveQuiz: PropTypes.func.isRequired,
    markCorrect: PropTypes.func.isRequired,
    questions: PropTypes.array.isRequired,
    quizSettings: PropTypes.object.isRequired
  };;

  constructor(props) {
    super(props);

    this.state = {
      correctAnswer: undefined,
      currentCategory: 0,
      settingsAreOpen: false
    };
  }

  openSettings = () => {
    this.setState({ settingsAreOpen: true });
  };

  closeSettings = () => {
    this.setState({ settingsAreOpen: false });
  };

  changeCategory = e => {
    let categoryId = parseInt(e.target.value);

    this.setState({ currentCategory: categoryId });

    this.props.changeQuestion(this.props.questions.filter(x =>
      x.categoryId === categoryId
    )[0].id);
  };

  editQuestion = body => {
    this.props.editQuestion(
      this.props.currentQuestion.id,
      body
    );
  };

  render() {
    // Used to prefix each answer.
    let letters = ['A', 'B', 'C', 'D'];

    // The name of the current category.
    let currentCategoryName = this.props.categories.filter(x =>
      x.id === this.state.currentCategory
    )[0].body;

    // The length of the current category.
    let currentCategoryLength = this.props.questions.filter(x =>
      x.categoryId === this.state.currentCategory
    ).length;

    // The index of the current question in relation to the
    // current category.
    let currentQuestionIndex = findIndex(this.props.questions.filter(x
        =>
      x.categoryId === this.state.currentCategory
    ), question => question.id === this.props.currentQuestion.id) + 1;

    return (
      <div>
        <div className="select-group">
          <Select changeEvent={this.changeCategory}
```

```
                currentlySelected={this.props.currentCategory}
                colours={this.props.colours.select}
                direction="left"
                options={this.props.categories}
                placeholder="Choose a category..."
                size="half"
                suffix="questions" />

        <Select changeEvent={this.props.changeQuestion}
                colours={this.props.colours.select}
                currentlySelected={this.props.currentQuestion.id}
                direction="right"
                indexes={true}
                options={this.props.questions.filter(x =>
                  x.categoryId === this.state.currentCategory
                )}
                placeholder="Choose a question..."
                size="half"/>
      </div>

      <EditableText colours={this.props.colours.text}
                    finishFunction={this.editQuestion}
                    house={this.props.house}
                    icon="pencil"
                    iconClass="question-pencil"
                    inputClass="question-input"
                    text={this.props.currentQuestion.body}
                    textType="h2" />

      <h3 style={this.props.colours.text.secondary}>
        {`Question ${currentQuestionIndex} out of ${
            currentCategoryLength} in the ${currentCategoryName}
            category•
         Question ${this.props.currentQuestion.id + 1} out of ${this.
             props.questions.length} overall•
         Quiz will take ${this.props.questions.length * this.props.
             quizSettings.questionLength / 1000} seconds `}
      </h3>

      <ul>
        {this.props.currentQuestion.answers.map((answer, i) =>
          <Answer answer={answer.body}
                  correct={answer.correct}
                  colours={this.props.colours.answer}
                  editAnswer={this.props.editAnswer}
                  house={this.props.house}
                  id={answer.id}
                  markCorrect={this.props.markCorrect}
                  letter={letters[i]}
                  key={answer.id}
                  questionId={this.props.currentQuestion.id} />
          )
        }
      </ul>

      <ul className="button-group">
        <Button clickEvent={this.props.addQuestion.bind(this, this.
            state.currentCategory)}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Add a question"
                icon="plus"
                house={this.props.house} />

        <Button clickEvent={this.props.deleteQuestion}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Delete question"
                icon="trash"
                house={this.props.house} />

        <Button clickEvent={this.props.addCategory}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Add a category"
                icon="plus"
                house={this.props.house} />

        <Button clickEvent={this.props.deleteCategory.bind(this, this
            .state.currentCategory)}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Delete category"
                icon="trash"
                house={this.props.house} />

        <Button clickEvent={this.props.editCategory.bind(this, this.
            state.currentCategory)}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Rename category"
                icon="recycle"
                house={this.props.house} />

        <Button clickEvent={this.openSettings}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Quiz settings"
                icon="cog"
                house={this.props.house} />

        <Button clickEvent={this.props.finishQuiz}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Save quiz"
                icon="graduation-cap"
                house={this.props.house} />

        <Button clickEvent={this.props.leaveQuiz}
                colours={this.props.colours.button}
                customClass="create-quiz-button"
                text="Leave editor"
                icon="sign-out"
                house={this.props.house} />
```

```
        </ul>


        <QuizSettingsPanel colours={this.props.colours}
                           closeSettings={this.closeSettings}
                           currentSettings={this.props.quizSettings}
                           editSettings={this.props.editSettings}
                           house={this.props.house}
                           saveSettings={this.props.saveSettings}
                           settingsAreOpen={this.state.settingsAreOpen}
                                />

      </div>
    );
  }
}

export default CreateQuiz;
```

Listing 34: Create quiz component

11.8.9 Button.js

```
import React, { Component, PropTypes } from 'react';

class Button extends Component {
  static propTypes = {
    clickEvent: PropTypes.func.isRequired,
    colours: PropTypes.object.isRequired,
    customClass: PropTypes.string,
    icon: PropTypes.string,
    text: PropTypes.string.isRequired
  };

  render() {
    return (
      <li style={this.props.colours}
          className={`button ${this.props.customClass}`}
          onClick={this.props.clickEvent}>
          {this.props.text}
          {this.props.icon ? <i className={`fa fa-${this.props.icon}
              button-icon`}></i> : null}
      </li>
    );
  }
}

export default Button;
```

Listing 35: Button box component.

11.8.10 Select.js

```
import React, { Component, PropTypes } from 'react';
import capitalise from 'lodash/string/capitalize';
import trim from 'lodash/string/trim';
```

```

class Select extends Component {
  static propTypes = {
    changeEvent: PropTypes.func.isRequired,
    colours: PropTypes.object.isRequired,
    currentlySelected: PropTypes.number,
    direction: PropTypes.oneOf(['left', 'right']),
    indexes: PropTypes.bool,
    options: PropTypes.array.isRequired,
    placeholder: PropTypes.string.isRequired,
    prefix: PropTypes.string,
    size: PropTypes.oneOf(['full', 'half']),
    suffix: PropTypes.string
  };

  render() {
    let innerClass;
    let outerClass;

    if (this.props.size === 'half') {
      outerClass = 'select-half-parent';
      innerClass = 'select-half';
    }

    return (
      <div className={`select-${this.props.direction} ${outerClass}`}>
        <select className={`select-${this.props.direction} ${innerClass
            }`}
                onChange={this.props.changeEvent}
                value={this.props.currentlySelected}
                style={this.props.colours}>

          {this.props.options.map((option, index) =>
            typeof this.props.options[0] !== 'object' ?
              // Used if a simple 1D array is passed.
              <option key={option} value={option}>
                {`${this.props.prefix ? this.props.prefix : ''} ${
                    capitalise(option)}`}
              </option> :
              // Used if a complex array containing objects
              // is passed through.
              <option key={option.id + 1} value={option.id}>
                {`${this.props.indexes ? (index + 1) + '.' : ''}
                  ${option.body}
                  ${this.props.suffix ? this.props.suffix : ''}`
                }
              </option>
          )}
        </select>

        <div className="select-arrow">
          <i className={`${this.props.direction}-arrow`}></i>
        </div>
      </div>
    );
  }
}
```

```

export default Select;
```

Listing 36: Select box component.

11.8.11 Answer.js

```
import React, { Component, PropTypes } from 'react';
import EditableText from 'components/EditableText';

class Answer extends Component {
  static propTypes = {
    answer: PropTypes.string.isRequired,
    correct: PropTypes.bool.isRequired,
    editAnswer: PropTypes.func.isRequired,
    house: PropTypes.string.isRequired,
    id: PropTypes.number.isRequired,
    letter: PropTypes.oneOf(
      ['A', 'B', 'C', 'D']
    ).isRequired,
    markCorrect: PropTypes.func.isRequired,
    questionId: PropTypes.number.isRequired
  };

  editAnswer = (body) => {
    this.props.editAnswer(
      this.props.id,
      body,
      this.props.correct
    );
  };

  render() {
    return (
        <li style={this.props.colours.body} className="answer">
          <span className="letter">
            {this.props.letter}.
          </span>

          <EditableText finishFunction={this.editAnswer}
                        house={this.props.house}
                        inputClass="answer-input"
                        text={this.props.answer}
                        textType="span" />

          <span className="answer-check"
                style={this.props.correct ? this.props.colours.check :
                    null}
                onClick={this.props.markCorrect.bind(
                         this,
                         parseInt(this.props.id),
                         parseInt(this.props.questionId),
                         this.props.answer,
                         !this.props.correct)}>
                <i className="fa fa-check-circle"></i>
          </span>
        </li>
    );
  }
}

export default Answer;
```

Listing 37: Quiz creator answer component.

11.8.12 EditableText.js

```
import React, { Component, PropTypes } from 'react';
import trim from 'lodash/string/trim';

class EditableText extends Component {
  static propTypes = {
    finishFunction: PropTypes.func.isRequired,
    house: PropTypes.string.isRequired,
    icon: PropTypes.string,
    iconClass: PropTypes.string,
    inputClass: PropTypes.string.isRequired,
    text: PropTypes.string.isRequired,
    textType: PropTypes.string.isRequired
  };

  constructor(props) {
    super(props);

    this.state = {
      isEditing: false,
      text: this.props.text
    };
  }

  beginEditing = () => {
    this.setState({ isEditing: true });
  };

  finishEditing = () => {
    this.setState({ isEditing: false });
    // Only update the question if the user
    // has made changes and length > 0.
    if (this.state.text.length && this.state.text !== this.props.text)
        {
      this.props.finishFunction(trim(this.state.text));
    }
  };

  handleChange = (e) => {
    this.setState({ text: e.target.value });
  };

  render() {
    let headingView;

    // Rendered if the user is trying to edit the
    // element. A nice input box linked to the above
    // functions.
    const editView = (
      <input autoFocus={true}
             className={
              `input-editable
               ${this.props.house}-input
               ${this.props.inputClass}`
             }
             onBlur={this.finishEditing}
             onChange={this.handleChange}
             placeholder={this.props.text} />
    );

    // Used if the user is just viewing the text.
    // Changes text type depending on properties
    // passed through.
    switch (this.props.textType) {
      case 'p':
        headingView = (
           <p className={`p-${this.props.house}`}
              onClick={this.beginEditing}>
              {this.props.text}
              {this.props.icon ?
                <i className={`fa fa-${this.props.icon} ${this.props.
                    iconClass}`}></i> :
                null
              }
          </p>
        );
        break;

      case 'span':
        headingView = (
          <span onClick={this.beginEditing}>
            {this.props.text}
            {this.props.icon ?
                <i className={`fa fa-${this.props.icon} ${this.props.
                    iconClass}`}></i> :
                null
              }
          </span>
        );
        break;

      case 'h2':
        headingView = (
          <h2 className={`h2-${this.props.house}`}
              style={this.props.colours.primary}
              onClick={this.beginEditing}>
              {this.props.text}
              {this.props.icon ?
                <i className={`fa fa-${this.props.icon} ${this.props.
                    iconClass}`}></i> :
                null
              }
          </h2>
        );
        break;

      default:
        headingView = <p>Something has gone wrong!</p>;
    }

    return this.state.isEditing ? editView : headingView;
  }
}

export default EditableText;
```

Listing 38: Dual heading / input component.

11.8.13 QuizSettingsPanel.js

```
import React, { Component, PropTypes } from 'react';
import settingsStyle from 'utils/settingsStyle';
import Modal from 'react-modal';
import Button from 'components/Button';

class QuizSettingsPanel extends Component {
  static propTypes = {
    closeSettings: PropTypes.func.isRequired,
    colours: PropTypes.object.isRequired,
    currentSettings: PropTypes.object.isRequired,
    house: PropTypes.string.isRequired,
    settingsAreOpen: PropTypes.bool.isRequired
  };

  saveSettings = () => {
    this.props.saveSettings({
      title: this.refs.title.value,
      startDate: this.refs.startDate.value,
      startTime: this.refs.startTime.value,
      questionLength: this.refs.questionLength.value,
      breakLength: this.refs.breakLength.value
    });

    this.props.closeSettings();
  };

  render() {
    return (
      <Modal isOpen={this.props.settingsAreOpen}
             closeTimeoutMS={150}
             onRequestClose={this.props.closeSettings}
             style={settingsStyle(this.props.colours.settings.
                 borderColour)}>

             <h2 className="centre" style={this.props.colours.text.
                 primary}>Quiz Settings</h2>
             <p className="centre" style={this.props.colours.text.
                 secondary}>Modify the quiz settings in this panel.</p>

             <form>
               <div className="row">
                 <div className="field col-md-12">
```

```
                  <label style={this.props.colours.settings.label}
                      className={`label-${this.props.house}`}>
                     Quiz name:
                  </label>

                  <input className="input-settings"
                         style={this.props.colours.select}
                         placeholder={this.props.currentSettings.title
                             }
                         ref="title"
                         type="text" />
                </div>
              </div>

              <div className="row">
                <div className="field col-md-6">
                  <label style={this.props.colours.settings.label}
                      className={`label-${this.props.house}`}>
                     Start day (the day it will be held):
                  </label>

                  <input className="input-settings"
                         style={this.props.colours.select}
                         ref="startDate"
                         type="date" />
                </div>

                <div className="field col-md-6">
                  <label style={this.props.colours.settings.label}
                      className={`label-${this.props.house}`}>
                         Start time (the time on that day):
                  </label>

                  <input className="input-settings"
                         style={this.props.colours.select}
                         ref="startTime"
                         type="time" />
                </div>
              </div>

              <div className="row">
                <div className="field col-md-6">
                  <label style={this.props.colours.settings.label}
                      className={`label-${this.props.house}`}>
                         Question length (seconds):
                  </label>

                  <input className="input-settings"
                         style={this.props.colours.select}
                         ref="questionLength"
                         placeholder={this.props.currentSettings.
                             questionLength / 1000}
                         type="number"
                         min="5" />
                </div>

                <div className="field col-md-6">
```

```
                    <label style={this.props.colours.settings.label}
                        className={`label-${this.props.house}`}
                           htmlFor="quizBreak">
                           Break length (minutes):
                    </label>

                    <input className="input-settings"
                           style={this.props.colours.select}
                           ref="breakLength"
                           placeholder={this.props.currentSettings.
                               breakLength / 60000}
                           type="number"
                           min="1" />
                  </div>
                </div>

                <div className="row">
                 <div className="field col-md-6 setting-buttons">
                   <Button colours={this.props.colours.button}
                           clickEvent={this.saveSettings}
                           customClass="create-quiz-button"
                           text="Save settings"
                           icon="save"
                           house={this.props.house} />
                 </div>
                 <div className="field col-md-6 setting-buttons">
                   <Button colours={this.props.colours.button}
                           clickEvent={this.props.closeSettings}
                           customClass="create-quiz-button"
                           text="Cancel changes"
                           icon="ban"
                           house={this.props.house} />
                 </div>
                </div>
              </form>
       </Modal>
     );
  }
}

export default QuizSettingsPanel;
```

Listing 39: Quiz settings panel.

11.9 Application Tooling

This subsection lists the tooling and build process that is required for the application to run. Though not technically part of the source, a strong understanding is vital if one is to come to terms with the project.

11.9.1 index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

import { Provider } from 'react-redux';
import { Route, Router } from 'react-router';
import configureStore from 'store/configureStore';
import createBrowserHistory from 'history/lib/createBrowserHistory';
import DevTools from 'containers/DevTools';

import LoginContainer from 'containers/LoginContainer';
import PlayQuizContainer from 'containers/PlayQuizContainer';
import CreateQuizContainer from 'containers/CreateQuizContainer';
import ResultsContainer from 'containers/ResultsContainer';

import '../node_modules/animate.css/animate.min.css';
import '!style!css!sass!styles/porfiry.scss';

const store = configureStore();
const history = createBrowserHistory();

ReactDOM.render(
  <Provider store={store}>
    <div>
      <Router history={history}>
        <Route path="/" component={LoginContainer} />
        <Route path="create" component={CreateQuizContainer} />
        <Route path="play" component={PlayQuizContainer} />
        <Route path="results" component={ResultsContainer} />
      </Router>
      <DevTools />
    </div>
  </Provider>,
  document.getElementById('root')
);
```

Listing 40: Main application entry.

11.9.2 index.html

This file serves as an entry point for the client.

```
<!DOCTYPE html>
<html lang="en" style="height:100%;width:100%">
  <head>
    <meta charset="UTF-8">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black"
        />
    <meta name="viewport" content="width=device-width, initial-scale=1"
        >
    <link rel="stylesheet" href='//fonts.googleapis.com/css?family=
        Dosis:600,700,800'>
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome
        /4.4.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap
        /3.3.5/css/bootstrap.min.css">
    <script src="//cdnjs.cloudflare.com/ajax/libs/fastclick/1.0.6/
        fastclick.min.js"></script>
    <title>Priory School Quiz</title>
  </head>
  <body ontouchstart>
    <div id='root'></div>
    <script type="text/javascript">
      if ('addEventListener' in document) {
        document.addEventListener('DOMContentLoaded', function() {
          FastClick.attach(document.body);
        }, false);
      }
    </script>
    <script src="/static/bundle.js"></script>
  </body>
</html>
```

Listing 41: Main HTML entry.

11.9.3 webpack.config.babel.js

This file compiles all the client dependencies for the application.

```
import path from 'path';
import webpack from 'webpack';

let config = {
  devtool: 'eval',
  entry: [
    'webpack-hot-middleware/client',
    './src/index'
  ],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js',
    publicPath: '/static/'
  },
  resolve: {
    alias: {
      actions: path.resolve('./src/', 'actions'),
      components: path.resolve('./src/', 'components'),
      constants: path.resolve('./src/', 'constants'),
      containers: path.resolve('./src/', 'containers'),
      libs: path.resolve('./src/', 'libs'),
      reducers: path.resolve('./src/', 'reducers'),
      store: path.resolve('./src/', 'store'),
      styles: path.resolve('./src/', 'styles'),
      utils: path.resolve('./src/', 'utils')
    }
  },
  plugins: [
    new webpack.HotModuleReplacementPlugin(),
    new webpack.NoErrorsPlugin()
  ],
  module: {
    preLoaders: [{
      test: /\.js$/,
      loaders: ['eslint-loader'],
      exclude: /node_modules/
    }],
    loaders: [
      {
        test: /\.js$/,
        loaders: ['babel'],
        include: path.join(__dirname, 'src')
      },
      {
        test: /\.css$/,
        loaders: ['style', 'css', 'sass'] }
    ]
  }
};

export default config;
```

Listing 42: Bootstraps the backend.

11.10 Styles

This subsection lists the stylesheets that control the majority of the application's visual aspects.

11.10.1 porfiry.scss

```
@import "helpers/variables";
@import "helpers/mixins";

@import "base/typography";

@import "containers/login";

@import "components/button";
@import "components/select";
@import "components/answer";
@import "components/answer-block";
@import "components/question-timer";
@import "components/input";
@import "components/notifications";
@import "components/countdown";
@import "components/modal";
```

Listing 43: Entry point for styles and for importing the other sheets.

11.10.2 typography.scss

```
h1
{
    font-family: $font-stack;
    font-size: 70px;

    margin-bottom: 5px;
}

h2
{
    font-family: $font-stack;
    font-size: 50px;

    margin-top: 0;
    margin-bottom: 15px;
}

h3
{
    font-family: $font-stack;

    margin-bottom: 0;
}

p
{
    font-family: $font-stack;
    font-size: 26px;
}

.question-title
{
    font-size: 4em;

    margin-top: 5px;
    margin-bottom: 5px;
}

.ReactModal__Content label
{
    font-family: $font-stack;
    font-size: 17px;

    display: block;
    clear: both;

    padding-bottom: 3px;
}

ul
{
    padding: 0;
}

li
{
    list-style-type: none;
}

.quiz-titles
{
    p
    {
        font-size: 20px;
    }
}

.centre
{
    text-align: center;
}

form
{
    width: 515px;
    margin: 0 auto;
}
```

Listing 44: Styling for typographical elements of the application.

11.10.3 answer.scss

```
.answer
{
    font-family: $font-stack;
    font-size: 28px;

    margin-top: 20px;
    margin-bottom: 20px;
    padding: 16px 0 10px 20px;

    text-align: left;

    transition: all .2s ease;

    border: 3px solid;
    border-radius: 4px;

    @include size(100%, 80px);
}

.quiz-answer {
  margin-top: 0;
  margin-bottom: 15px;
}

.letter
{
    padding-right: 15px;
}

.answer-check
{
    font-size: 35px;

    position: relative;
    bottom: 17px;

    float: right;

    height: 76px;
    padding: 12px 25px 0 25px;
}

.answer-edit
{
    font-size: 35px;

    position: absolute;
    position: relative;
    bottom: 3px;

    float: right;

    height: 76px;
}
```

Listing 45: Styling for the answer blocks.

11.10.4 button.scss

```
.button
{
    font-family: $font-stack;
    font-size: 25px;

    display: inline-block;

    padding: 8px;

    cursor: pointer;
    -webkit-transition: all .2s ease;
            transition: all .2s ease;
    text-align: center;

    border: 3px solid;
    border-radius: 4px;

    @include size(243px, 60px);
}


.button:active
{
    -webkit-transform: scale(.6) !important;
            transform: scale(.6) !important;

    opacity: .8;
}


.login-button
{
    margin: 9px 0 0 9px;
}

```

```
.button-group
{
    position: absolute;
    bottom: 0;
}

.create-quiz-button
{
    margin: 10px 10px 0 0;
}

.button-icon
{
    position: relative;
    top: 2px;

    margin-left: 10px;
}

.setting-buttons
{
    font-size: 23px;

    height: 50px;
}

.quiz-title-button {
  font-size: 22px;
  padding-top: 5px;
  margin-right: 10px;

  @include size(150px, 50px);
}

.join-quiz-button {
  margin-left: 130px;
}
```

Listing 46: Styling for the buttons.

11.10.5 input.scss

```
input {
  font-family: $font-stack;
}

.input-editable {
  border: none;
  background-color: rgba(0, 0, 0, 0) !important;
}

.input-settings {
  font-family: $font-stack;
  font-size: 22px;

  border-radius: 4px;
  border: 3px solid #000;
  padding-left: 10px;

  -webkit-appearance: none;

  @include size(100%, 50px);
}

input:focus {
  outline: 0;
}

.question-input {
  font-size: 50px;
  width: 100%;
  margin-top: -20px;
  margin-bottom: -20px;
  padding-left: 0;
}

.answer-input {
  font-size: 28px;
  position: relative;
  bottom: 6px;
  padding-left: 0;
  width: 75%;
}

.fa-pencil {
  margin-left: 15px;
  bottom: 4px;
  position: relative;
}

.question-pencil {
  font-size: 40px;
}

form {
  .field {
    margin-bottom: 15px;
  }
}
```

Listing 47: Styling for the text input elements.

11.10.6 notification.scss

```
.notification {
  font-family: $font-stack;
  font-size: 22px;

  border: 3px solid;
  border-radius: 4px;

  padding-top: 10px;
  padding-left: 15px;

  overflow: hidden;

  position: absolute;

  float: right;
  top: 20px;
  right: 20px;

  @include size(520px, 60px);
}

.notification-success {
  border-color: #0fad1c;
  background-color: rgb(118, 236, 127);
  color: #0fad1c;

  -webkit-box-shadow: 0px 5px 3px 0px rgba(55, 194, 0, 0.18);
  -moz-box-shadow: 0px 5px 3px 0px rgba(55, 194, 0, 0.18);
  box-shadow: 0px 5px 3px 0px rgba(55, 194, 0, 0.18);
}

.notification-danger {
  border-color: #A22929;
  background-color: #ff6f47;
  color: #A22929;

  -webkit-box-shadow: 0px 5px 3px 0px rgba(245, 107, 107, 0.18);;
  -moz-box-shadow: 0px 5px 3px 0px rgba(245, 107, 107, 0.18);;
  box-shadow: 0px 5px 3px 0px rgba(245, 107, 107, 0.18);;
}
```

Listing 48: Styling for the alert notifications.

11.10.7 select.scss

```
select {
  font-family: $font-stack;
  font-size: 22px;

  padding-left: 12px;

  border: 3px solid;
  border-radius: 4px;

  margin: 0px 0 10px 0;

  -webkit-appearance: none;
  -moz-appearance: none;

  padding-top: 0 !important;

  @include size(512px, 55px);

  &.hover:hover {
    transform: scale(1.03);
    -webkit-transform: scale(1.03);
  }

  transition: all 0.2s ease-in-out;
}

.select-arrow {
  margin-left: -35px;
  margin-bottom: 3px;
  display: inline-block;
}

.left-arrow {
  position: relative;
  top: 20px;
}

.right-arrow {
  float: right;
  position: absolute;
  right: 30px;
  top: 35px;
}

.select-arrow i {
  width: 0;
  height: 0;
  border-left: 7px solid transparent;
  border-right: 7px solid transparent;
  display: block;
}

.select-full {
  margin: 0;
  @include size(100%, 60px);
}

.select-half-parent {
  width: 50%;
}

.select-half {
  width: 99%;
}

.select-left {
  float: left;
}

.select-right {
  float: right;
}

.select-group {
  display: inline-block;
  width: 100%;
}
```

Listing 49: Styling for the select elements.

11.10.8 modal.scss

```
.ReactModal__Overlay
{
    overflow-x: hidden;
    overflow-y: auto;

    opacity: 0;
    background-color: rgba(0, 0, 0, .5);

    -webkit-perspective: 600;
            perspective: 600;
}

.ReactModal__Overlay--after-open
{
    transition: opacity 150ms ease-out;

    opacity: 1;
}

.ReactModal__Content
{
    -webkit-transform: scale(.5) rotateX(-30deg);
}

.ReactModal__Content--after-open
{
    transition: all 150ms ease-in;
    -webkit-transform: scale(1) rotateX(0deg);
}

.ReactModal__Overlay--before-close
{
    opacity: 0;
}

.ReactModal__Content--before-close
{
    transition: all 150ms ease-in;
    -webkit-transform: scale(.5) rotateX(30deg);
}

.ReactModal__Content.modal-dialog
{
    border: none;
    background-color: transparent;
}
```

Listing 50: Styling for the modal elements.

11.10.9 login.scss

```
.loginForm
{
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;

  width: 100%;
  height: 400px;
  margin: auto;
}
```

Listing 51: Styling for the login form ensuring it is positioned properly.

11.10.10 variables.scss

```
// Typography variables
$font-stack: Dosis, sans-serif;
```

Listing 52: Variables used throughout the styles.

11.10.11 mixins.scss

```
@mixin size($width, $height: $width) {
  width: $width;
  height: $height;
}

@mixin prefix($map, $vendors: webkit moz ms o) {
  @each $prop, $value in $map {
    @if $vendors {
      @each $vendor in $vendors {
        #{'-' + $vendor + '-' $prop}: #{$value};
      }
    }
    #{$prop}: #{$value};
  }
}
```

Listing 53: Mixins used to apply processes across styles.

12 Program Variables

This section displays the variables used in throughout the program. Due to the functional nature of the language used, variables are used freely, so their usage is rather extensive. There are two types of variables used:

- Block scoped - these variables, defined using the *let* keyword, are similar to standard variables used in most programming languages, and have been used for values that are in some way dynamic or likely to mutate throughout program execution.
- Constants - declared using the *const* keyword, these variables are used for things like modules that have been imported from elsewhere in the codebase, and for variables that read from another source, such as the immutable state store or a *prop* parameter from a parent component.

Each variable listed below is accompanied by its name, its purpose, the data type it holds one of *String*, *Number*, *Object*, *Date*, *Boolean*, *Function*, or *Array*; the module or function it is used in, and whether it was defined using *let* or *const*.

12.1 Main Variables

These variables make up the majority used in the system, and are used for a variety of purposes, the extent to which is described for each one.

Name	**Module**	**Type**	**Stores**	**Purpose**
packet	PlayQuizContainer	const	Object	Stores the answer packet.
questionId	PlayQuizContainer	const	Number	The current question ID.
categoryId	PlayQuizContainer	const	Number	The current category ID.
quizInfo	PlayQuizContainer	let	Object	Info on current quiz.
housePoints	ResultsContainer	const	Object	Total house points earned.
highestScore	ResultsContainer	const	Number	Highest house points won.
winningHouse	ResultsContainer	const	String	The winning house.
notifSend	CreateQuizContainer	const	Object	Notifications library.
categoryBody	CreateQuizContainer	let	String	Body for custom category.
currentBody	CreateQuizContainer	let	String	Current category body.
newId	CreateQuizContainer	let	Number	Latest question ID.
deleteId	CreateQuizContainer	let	Number	ID of question to delete.
deleteIndex	CreateQuizContainer	let	Number	Index of question to delete.
nextQuestionIndex	CreateQuizContainer	let	Number	Index of the question.
nextQuestionId	CreateQuizContainer	let	Number	ID of the next question.
currentlyCorrect	CreateQuizContainer	const	Object	Currently correct answer.
id	CreateQuizContainer	const	Number	ID of question to move to.
finished	CreateQuizContainer	const	Boolean	Whether the quiz is finished.

quiz	CreateQuizContainer	const	Object	Structured quiz for DB.
value	CreateQuizContainer	let	String	Current settings value.
id	CreateQuizContainer	const	Number	ID of the current question.
currentQuestion	CreateQuizContainer	const	Object	Info on the current question.
notificationTheme	CreateQuizContainer	const	Object	Styling for notifications.
houses	LoginContainer	const	Array	The houses in the school.
years	LoginContainer	const	Array	The years in the school.
quiz	LoginContainer	const	Object	Quiz the user has selected.
questionSchema	Question (model)	const	Object	Defines schema for a question.
answerSchema	Answer (model)	const	Object	Defines schema for an answer.
quizSchema	Quiz (model)	const	Object	Defines schema for a quiz.
router	api/index	let	Object	An HTTP router.
quiz	api/routes/quizzes	let	Object	Holds the quiz schema.
store	index	const	Object	Access point for immutable state.
history	index	const	Object	Stores browser history.
socket	quizEvents	const	Object	Links module to socket server.
io	quizSockets	let	Object	Connects socket server to index.
quizIndex	quizSockets	let	String	Address to quiz API.
jobs	quizSockets	let	Object	Holds the scheduled quiz tasks.
currentQuiz	quizSockets	let	Object	Holds the quiz currently running.
quizStatus	quizSockets	let	String	Current status of the quiz.
players	quizSockets	let	Array	Array of all connected players.
answers	quizSockets	let	Object	Sorted hash of all user's answers.
housePoints	quizSockets	let	Object	Current house points total.
houses	quizSockets	const	Array	The houses in the school.
quizStart	quizSockets	const	Date	Start date for the next quiz.
countdownStart	quizSockets	let	Date	Time that countdown wills start.
totalQuizDuration	quizSockets	let	Number	Time in ms of next quiz.
showResults	quizSockets	let	Date	Time that results be shown.
quizFinish	quizSockets	let	Date	Time that quiz will finish.
questionLength	quizSockets	let	Number	Length of question, in ms.
changeQuestion	quizSockets	const	Date	Time that question will change.
countdown	quizSockets	const	Function	SetInterval for question timer.
defaultQuiz	defaultQuiz	const	Object	Default new quiz loaded.
choice	choice	const	Any	Random item from array.
colours	colourScheme	const	Object	Different house colour schemes.
ids	nextBiggest	const	Array	Array of question ids.
sharedConfig	config	const	Object	Config shared between dev and pro.
proConfig	config	let	Object	Production configuration.
devConfig	config	let	Object	Development configuration.
rootReducer	reducers/index	const	Object	Combines all the other reducers.
store	configureStore	const	Object	Store composed with middleware.
nextRootReducer	configureStore	const	Object	The other reducers.

defaultState	userReducer	let	Object	Default house and year for user.
defaultState	quizReducer	let	Object	Initialises the quiz sections.
categories	normaliseQuiz	let	Array	A normalised array of categories.
questions	normaliseQuiz	let	Array	A normalised array of questions.
answers	normaliseQuiz	let	Array	A normalised array of answers.
initialState	answerStatistics	const	Object	Initialises the algorithm state.
newState	answerStatistics	const	Object	New state constructed in process.
BarChart	ResultsChart	const	Object	Charting library.
housePoints	ResultsChart	const	Object	Stores the house points locally.
data	ResultsChart	let	Object	Formats house points for chart.
headingView	EditableText	let	Object	The type of heading tag.
editView	EditableText	const	Object	Editing view for text.
innerClass	Select	let	String	Class applied inside select.
outerClass	Select	let	String	Class applied outside select.
categoryId	CreateQuiz	let	Number	ID of current category.
letters	CreateQuiz	let	Array	Stores A, B, C and D as guides.
categoryName	CreateQuiz	let	String	Name of the current category.
categoryLength	CreateQuiz	let	Number	#questions in the current category.
questionIndex	CreateQuiz	let	Number	Index of the current question.
textColour	LoadQuizPanel	let	Object	Colour of the panel's text.
receivedTime	QuestionTimer	let	Number	The time left until the next ping.
width	QuestionTimer	let	Number	Computed width of question timer.
letters	PlayQuiz	let	Array	Stores A, B, C and D as guides.
quizInfo	PlayQuiz	let	Object	Statistics on the current quiz.
socket	PlayQuizActions	const	Object	Connects module to socket server.
socket	LoginActions	const	Object	Connects module to socket server.
dismissAfter	LoginActions	let	Number	Time in ms until notifications go.
socket	CreatorActions	const	Object	Connects module to socket server.

Table 6: Main variables.

12.2 String Constants

Unlike the previous collection, these variables merely describe the *actions* that are used in the system - every one defines a "thing" that will happen, be that requesting a quiz to load, or deleting an answer. They are all stored in a single file, and referred to throughout the system.

Name	**Module**	**Type**	**Stores**	**Purpose**
CHANGE_COLOURS	actions	const	String	String constant for action.
CHANGE_HOUSE	actions	const	String	String constant for action.
CHANGE_YEAR	actions	const	String	String constant for action.
CHECK_IF_QUIZ_READY	actions	const	String	String constant for action.

QUIZ_IN_PROGRESS	actions	const	String	String constant for action.
QUIZ_IS_SCHEDULED	actions	const	String	String constant for action.
NO_QUIZ_READY	actions	const	String	String constant for action.
DELETE_QUIZ	actions	const	String	String constant for action.
DELETE_QUIZ_SUCCESS	actions	const	String	String constant for action.
DELETE_QUIZ_FAILURE	actions	const	String	String constant for action.
REQUEST_QUIZZES	actions	const	String	String constant for action.
REQUEST_QUIZZES_FAILURE	actions	const	String	String constant for action.
RECEIVE_QUIZZES	actions	const	String	String constant for action.
LOAD_DEFAULT_QUIZ	actions	const	String	String constant for action.
QUIZ_IS_READY	actions	const	String	String constant for action.
UPDATE_ID	actions	const	String	String constant for action.
UPDATE_TITLE	actions	const	String	String constant for action.
UPDATE_START_DATE	actions	const	String	String constant for action.
UPDATE_START_TIME	actions	const	String	String constant for action.
UPDATE_QUESTION_LENGTH	actions	const	String	String constant for action.
UPDATE_BREAK_LENGTH	actions	const	String	String constant for action.
UPDATE_IS_FINISHED	actions	const	String	String constant for action.
UPDATE_ALL_SETTINGS	actions	const	String	String constant for action.
ADD_CATEGORY	actions	const	String	String constant for action.
DELETE_CATEGORY	actions	const	String	String constant for action.
EDIT_CATEGORY	actions	const	String	String constant for action.
DELETE_ALL_CATEGORIES	actions	const	String	String constant for action.
ADD_QUESTION	actions	const	String	String constant for action.
DELETE_QUESTION	actions	const	String	String constant for action.
EDIT_QUESTION	actions	const	String	String constant for action.
DELETE_ALL_QUESTIONS	actions	const	String	String constant for action.
ADD_ANSWER	actions	const	String	String constant for action.
DELETE_ANSWER	actions	const	String	String constant for action.
EDIT_ANSWER	actions	const	String	String constant for action.
DELETE_ALL_ANSWERS	actions	const	String	String constant for action.
UPLOAD_QUIZ	actions	const	String	String constant for action.
JOIN_QUIZ	actions	const	String	String constant for action.
BEGIN_QUIZ	actions	const	String	String constant for action.
SHOW_RESULTS	actions	const	String	String constant for action.
LEAVE_QUIZ	actions	const	String	String constant for action.
DECREMENT_TIME_LEFT	actions	const	String	String constant for action.
SHOW_NEXT_QUESTION	actions	const	String	String constant for action.
MOVE_TO_CATEGORY	actions	const	String	String constant for action.
SELECT_ANSWER	actions	const	String	String constant for action.
RECEIVE_ANSWER	actions	const	String	String constant for action.
RECEIVE_HOUSE_POINTS	actions	const	String	String constant for action.

Table 7: String constant variables.

Part IV

Testing

This initial part of the documentation features the analysis that was performed on the business. It includes the business' background, as well as an in-depth investigation on their current system, featuring document inspections, interviews, and observations. Also included is a problem definition, wherein the broad aims of the project are outlined; this definition also makes reference to the limitations of the solution. Finally, detailed objectives are clearly laid out, providing an overview of exactly what the solution should achieve.

13 Test Strategy

This test strategy details the overall strategy that will be used to test the application. In order to ensure that the system has been tested fully, and in a manner that encompasses and guarantees all aspects of its existance, purpose, and fuctionality, a number of testing methods will be adopted. These will include:

- Unit testing - the testing of each individual "part" of the system, such as this algorithm, or the functionality of this button, in order to detect the extent to which they meet the objectives previously laid out.
- Full system testing - the testing of the entire application at once, from the creation of a quiz through to its completion. This will ensure that the system is fully functional, and can be used in the correct manner.
- Acceptance testing - in this section of the testing, a user will be given access to the application, and be asked to create a quiz and play through it. They will then give their opinion on all aspects of the system, from its design and usability to the degree to which it facilitates the completion of its goals. Through this form of testing, it is possible to discover how well designed and usable the system is.

The most extensive section of testing will be the unit testing, as the system is made up of a variety of different components. The following things will have to be tested to ensure that the system is properly functional:

- When the house is Acton, the interface should be blue.
- When the house is Baxter, the interface should be orange.
- When the house is Clive, the interface should be turquoise.
- When the house is Darwin, the interface should be purple.

- When the house is Houseman, the interface should be red.
- When the house is Webb, the interface should be yellow.
- In the quiz creation section of the program, the ability to do the following will be tested:
  - Add a question to the quiz.
  - Delete a question from the quiz.
  - Edit a question's title.
  - Add a new category to the quiz.
  - Rename a category in the quiz.
  - Edit an answer's body.
  - Set one of the answer's as correct.
  - Save the quiz to the server.
- In the quiz playing section of the quiz, the ability to do the following will be tested:
  - Move to the correct question on time, according to when the quiz started and the length of each question.
  - End the quiz at the correct time, after all the questions have been finished.
- Ensure that it is possible to edit a quiz that has already been saved.
- Ensure that quizzes can be deleted.
- Ensure that the correct buttons are shown on the login screen depending on whether a quiz has been scheduled in the next 20 minutes or is in progress.

Certain tests - specifically, testing the API and the scoring algorithms - are less suited to the screenshot based approach that will be taken with the majority of the application's components. Because of their nature, it makes little sense to provide a series of screenshots demonstrating a connection to the server. These would be easy to fake, difficult to capture, and would provide very little information of actual use. Additionally, the screenshots above regarding the saving and loading of quizzes demonstrates this perfectly well. Instead, automated unit tests, of the kind that would be used in industry, have been written. These provide a guarantee that the code functions properly on an algorithmic level, and, as such, act as perfect proof.

14 Test Plan

This section contains detailed test plans for each part of the system. Each individual test plan includes the part of the system that is being tested, the steps that should be performed, the data to be entered and the expected outcome.

14.1 Navigation Test Plan

In order to ensure that staff and pupils can navigate around the different sections of the application easily, the navigation put in place must be tested to ensure that it works correctly. The following aspects of navigation will be tested:

- Link to quiz creator should take them to the quiz creator.
- Link to quiz player should take them to the quiz player.
- Link out of quiz creator to login screen should take them to the login screen.
- Button to settings dialog should load the settings dialog.
- Select boxes in quiz creator takes users to the correct question / category.
- Button to load quiz dialog should load the load quiz dialog.
- Edit button on load quiz dialog should load the correct quiz.

14.2 Login Screen Testing

This subsection shows the test data that will be used to test the login screen. It should be noted that the login screen serves as a "portal" to other aspects of the system, displaying a number of different options based on context. The test data below will include full details of how such contextual situations should be replicated.

14.2.1 Connect to State Store

1. Open the developer tools, navigate to LoginContainer, and check that the store has been passed in the props list.

   Expected result: Connection to state store should be open.

14.2.2 Colours

This section demonstrates how to test that the system is properly able to theme itself depending on the colour's selected.

1. Select “Acton” in the house box.

   Expected result: Interface should turn blue.

2. Select “Baxter” in the house box.

   Expected result: Interface should turn orange.

3. Select “Clive” in the house box.

   Expected result: Interface should turn green.

4. Select “Darwin” in the house box.

   Expected result: Interface should turn purple.

5. Select “Houseman” in the house box.

   Expected result: Interface should turn red.

6. Select “Webb” in the house box.

   Expected result: Interface should turn yellow.

14.2.3 Load Quizzes

1. Create and save a quiz with the title “Testing Quiz”. (*Typical*)

   Expected result: One quiz with the title “Testing Quiz” should be shown in box.

2. Create and save five quizzes with the title “Testing Quiz”. (*Extreme*)

   Expected result: Five quizzes with the title “Testing Quiz” should be shown in box.

3. Turn off the internet and press the load quiz button. (*Erroneous*)

   Expected result: Box should contain message “Quizzes couldn’t be loaded”.

4. Do not save any quizzes. (*Null*)

   Expected result: Box should contain message “No quizzes available”.

14.2.4 Delete Quizzes

1. Create and save a quiz with the title “Testing Quiz”. (*Typical*)
2. Press the delete quiz button.

   Expected result: Quiz should be removed from the list, and should no longer be in database.

14.2.5 Show Correct Menu Buttons

1. Schedule a quiz for 20 minutes time. (*Typical*)

   Expected result: Join quiz button should be shown.

2. Schedule a quiz for 2 minutes time. (*Typical*)

   Expected result: Join quiz button should be shown.

3. Schedule a quiz for 1 months time. (*Typical*)

   Expected result: Create and load quiz buttons should be shown.

4. Do not schedule any quizzes. (*Null*)

   Expected result: Create and load quiz buttons should be shown.

14.3 Quiz Creator Test Plan

This subsection outlines the data that will be used to test the functionality of the quiz creator.

14.3.1 Connect to State Store

1. Open the developer tools, navigate to CreateQuizContainer, and check that the store has been passed in the props list.

   Expected result: Connection to state store should be open.

14.3.2 Add Question

1. Press the add question button to create question 02.

   Expected result: New question should be added.

14.3.3 Edit Question

1. Click question title
2. Type in “Who is the Prime Minister?”

   Expected result: Question title should now be “Who is the Prime Minister?”

14.3.4 Delete Question

1. Press the add question button to create question 02.
2. Press the delete question button to delete question 02.

   Expected result: Question 02 should be deleted.

14.3.5 Add Category

1. Press the add category button, and enter it “Literature”.

   Expected result: New category “Literature” should be created, with a default question ready.

14.3.6 Delete Category

1. Press the add category button, and call it “Literature”.
2. Press the delete category button.

   Expected result: Category “Literature” should be deleted along with all questions inside.

14.3.7 Rename Category

1. Press the add category button, and call it “Literature”.

2. Press the rename category button and type in “History”.

   Expected result: Category “Literature” should be renamed to “History”.

14.3.8 Edit Answer

1. Click answer 01
2. Type in “David Cameron”

   Expected result: Answer body should be “David Cameron‘”

14.3.9 Mark Answer as Correct

1. Press the second check mark on Answer 2.

   Expected result: Answer 3 should no longer be marked as correct; answer 2 should be.

14.3.10 Save Quiz

1. Create the following quiz:

```
{
  title: 'Summer Term House Quiz',
  start: new Date(),
  questionIntervals: 10000,
  realtimeGraphics: true,
  intervalLength: 300000,
  categories: [
    {
      name: 'History',
      questions: [
        {
          title: 'Who killed JFK?',
          answers: [
            { text: 'Bill Osmond', correct: false },
            { text: 'Lee Harvey Oswald', correct: true },
            { text: 'Henry Kissinger', correct: false }
          ]
        },
        {
          title: 'Who was the British Prime Minister in 1916?',
          answers: [
            { text: 'David Lloyd George', correct: true },
            { text: 'Lee Harvey Oswald', correct: false },
            { text: 'Lord Liverpool', correct: false }
          ]
        }
      ]
    },
    {
      name: 'Sport',
      questions: [
        {
          title: 'Who won the 1966 world cup?',
          answers: [
            { text: 'England', correct: true },
            { text: 'Wales', correct: false },
            { text: 'Scotland', correct: false },
            { text: 'Uzbekistan', correct: false }
          ]
        },
        {
          title: 'Which city hosted the 1924 Summer Olympics?',
          answers: [
            { text: 'Brussels', correct: false },
            { text: 'Orelans', correct: false },
            { text: 'Madrid', correct: false },
            { text: 'Paris', correct: true }
          ]
        }
      ]
    },
    {
      name: 'Literature',
      questions: [
        {
          title: 'Who serves as the main character in "Crime and Punishment"?',
          answers: [
            { text: 'Raskolnikov', correct: true },
            { text: 'Razumikhin', correct: false },
            { text: 'Porifry Petrovich', correct: false },
            { text: 'Svidrigailov', correct: false }
          ]
        },
        {
          title: 'Which literary genre does "Ulysses" belong to?',
          answers: [
            { text: 'Modernist', correct: true },
            { text: 'Post-modernist', correct: false },
            { text: 'Romance', correct: false },
            { text: 'Thriller', correct: false }
          ]
        }
      ]
```

```
        }
    ]
}
```

2. Press the save quiz button.

   Expected result: quiz should be saved to database and green success notification should appear.

14.4 Play Quiz Testing

This subsection demonstrates the test data that should be used to test the main quiz playing interface, as well as the steps that should be performed.

14.4.1 Connect to State Store

1. Open the developer tools, navigate to PlayQuizContainer, and check that the store has been passed in the props list.

   Expected result: Connection to state store should be open.

14.4.2 Move Questions on Time

1. Create and save a quiz with the title "Testing Quiz", and the following questions:
   (a) Who is the French Premier?
   (b) Who won the 1960 World Cup?
   (c) What is the capital of Iceland?

   Set it for two minutes time, and join the quiz.
2. When the quiz is in progress, the questions should navigate automatically in this order:
   (a) 0-10 seconds - *Who is the French Premier?*
   (b) 10-20 seconds - *Who won the 1960 World Cup?*
   (c) 20-30 seconds - *What is the capital of Iceland?*

14.4.3 End Quiz at Correct Time

1. Create and save a quiz with the title "Testing Quiz", and the single default question. Set it for two minutes time, and join the quiz.

2. After 15 seconds, the quiz should end, and the login page should be shown, displaying the create and load quiz buttons should be shown.

14.5 Results Screen

This section displays the tests that will be used to ensure the full functionality of the results screen.

14.5.1 Connect to State Store

1. Open the developer tools, navigate to ResultsContainer, and check that the store has been passed in the props list.

   Expected result: Connection to state store should be open.

15 Test Runs

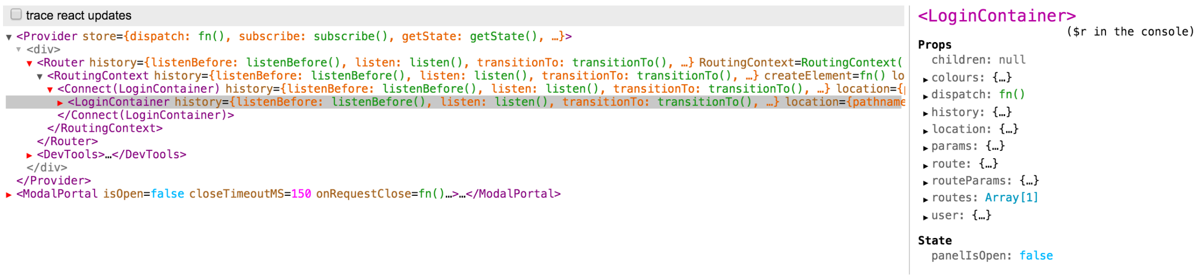
These are the actual test runs, indicating whether or not a test has successfully completed. For each test, the unit test code is included, followed by a screenshot of the test outcome. If a test is unsuccessful, the changes made to the code will be shown, followed by another screenshot of the test outcome.

15.1 Login Test Runs

This subsection contains the test runs for the login screen, using as a reference the login screen test plan available at 13.1.

15.1.1 Connects to State Store

This test ensures that the component can connect to the state store. As the



```
▼ <Provider store={dispatch: fn(), subscribe: subscribe(), getState: getState(), …}>
  ▼ <div>
    ▼ <Router history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} RoutingContext=RoutingContext(
      ▼ <RoutingContext history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} createElement=fn() lo
        ▼ <Connect(LoginContainer) history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} location={
          ▶ <LoginContainer history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} location={pathname
          </Connect(LoginContainer)>
        </RoutingContext>
      </Router>
    ▶ <DevTools>…</DevTools>
    </div>
  </Provider>
▶ <ModalPortal isOpen=false closeTimeoutMS=150 onRequestClose=fn()…>…</ModalPortal>
```

```
<LoginContainer>
                    ($r in the console)
Props
  children: null
▶ colours: {…}
▶ dispatch: fn()
▶ history: {…}
▶ location: {…}
▶ params: {…}
▶ route: {…}
▶ routeParams: {…}
▶ routes: Array[1]
▶ user: {…}
State
  panelIsOpen: false
```

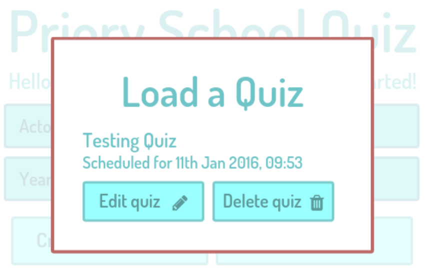
Figure 3: Props view of LoginContainer, showing the state store being passed.

screenshot demonstrates, the state has been passed to the component correctly, meaning the connection succeeded. *Success.*

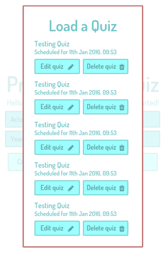
15.1.2 Load Quizzes

This test ensures that a list of all available quizzes can be searched for and found on the login screen, ready to either edit or delete.

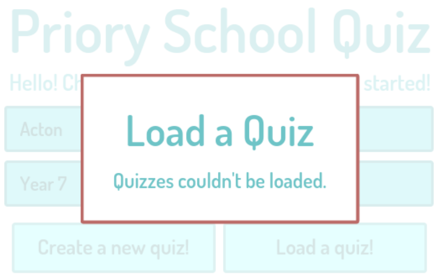
As can be seen, all the tests have worked successfully: when there is only a single quiz in the database, only a single quiz is shown; when there are five, all of these are listed; when there is no connection, and the system is unable to perform an API call, the correct error message is shown; and when there are simply no quizzes available, this too is properly relayed to the user. *Success.*



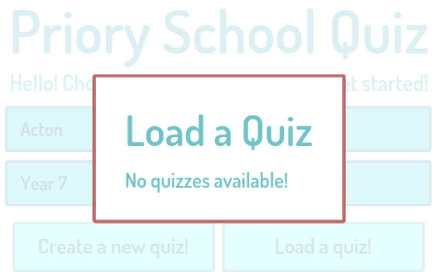
(a) Single quiz (typical)



(b) Five quizzes (extreme)



(c) No connection (erroneous)

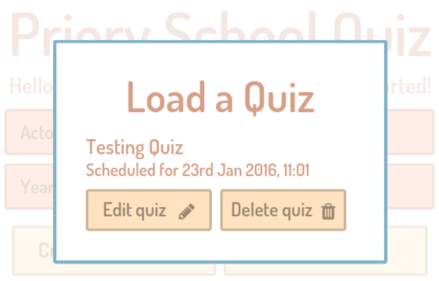


(d) No quizzes (null)

Figure 5: The load quiz dialog.

15.1.3 Delete Quizzes

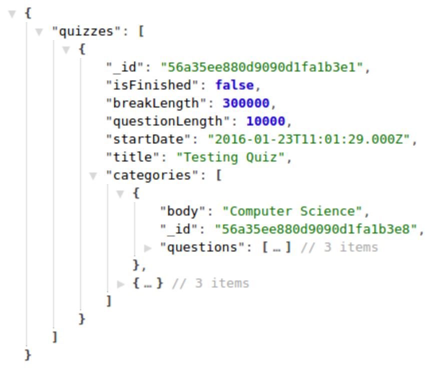
This test ensures that the user is able to delete a quiz from the database, using the button in the load quiz dialog.



Load a Quiz

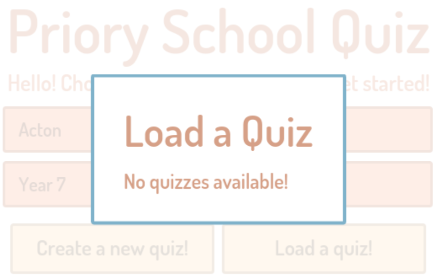
Testing Quiz
Scheduled for 23rd Jan 2016, 11:01

(a) Before (panel)



```
{
  "quizzes": [
    {
      "_id": "56a35ee880d9090d1fa1b3e1",
      "isFinished": false,
      "breakLength": 300000,
      "questionLength": 10000,
      "startDate": "2016-01-23T11:01:29.000Z",
      "title": "Testing Quiz",
      "categories": [
        {
          "body": "Computer Science",
          "_id": "56a35ee880d9090d1fa1b3e8",
          "questions": [ … ] // 3 items
        },
        { … } // 3 items
      ]
    }
  ]
}
```

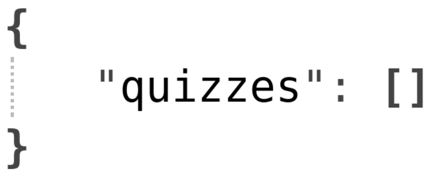
(b) Before (database)



Load a Quiz

No quizzes available!

(c) After (panel)



```
{
    "quizzes": []
}
```

(d) After (database)

Figure 7: Deleting a quiz.

As indicated by the *after* images, the quiz is correctly deleted from the database after the delete button is pressed. When the panel is loaded, it calls the API, which in turn calls the database, so the two will always reflect each other. As the quiz is no longer listed in either, the test has passed. *Success.*

15.1.4 Show Correct Menu Buttons

This test ensures that the correct buttons are shown on the login screen depending on the whether or not a quiz has been scheduled. Due to the time based nature of this test, it is impractical to take screenshots; therefore, a table has been included for a trusted individual to sign.

Test	Expected Result	Works?	Signature
Schedule a quiz for 20 minutes time.	Join quiz button should be shown.	Yes.	
Schedule a quiz for 2 minutes time.	Join quiz button should be shown.	Yes.	
Schedule a quiz for 1 months time.	Create and load quiz buttons should be shown.	Yes.	
Do not schedule any quizzes.	Create and load quiz buttons should be shown.	Yes.	

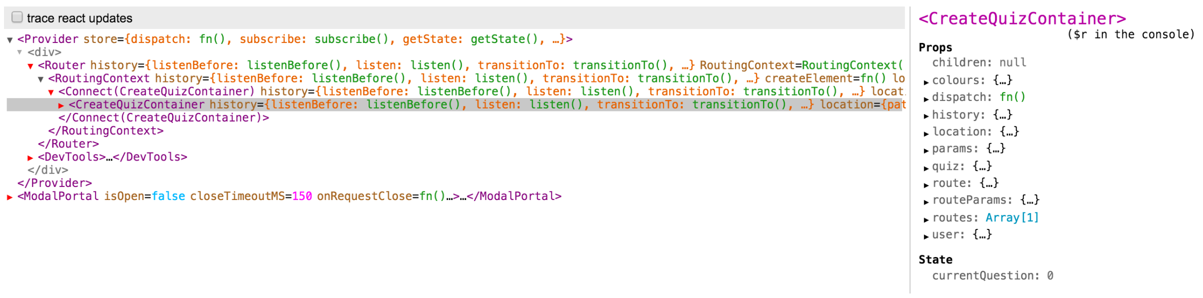
Table 8: Menu button combinations.

15.2 Create Quiz Test Runs

This section contains the test runs performed on the quiz creator.

15.2.1 Connects to State Store

This test ensures that the component can connect to the state store. As the



```
▼ <Provider store={dispatch: fn(), subscribe: subscribe(), getState: getState(), …}>
  ▼ <div>
    ▼ <Router history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} RoutingContext=RoutingContext(
      ▼ <RoutingContext history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} createElement=fn() lo
        ▼ <Connect(CreateQuizContainer) history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} locat
          ▶ <CreateQuizContainer history={listenBefore: listenBefore(), listen: listen(), transitionTo: transitionTo(), …} location={pat
          </Connect(CreateQuizContainer)>
        </RoutingContext>
      </Router>
    ▶ <DevTools>…</DevTools>
    </div>
  </Provider>
▶ <ModalPortal isOpen=false closeTimeoutMS=150 onRequestClose=fn()…>…</ModalPortal>
```

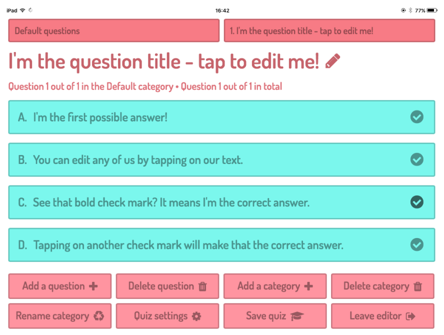
```
<CreateQuizContainer>
                    ($r in the console)
Props
  children: null
▶ colours: {…}
▶ dispatch: fn()
▶ history: {…}
▶ location: {…}
▶ params: {…}
▶ quiz: {…}
▶ route: {…}
▶ routeParams: {…}
▶ routes: Array[1]
▶ user: {…}

State
  currentQuestion: 0
```

Figure 8: Props view of CreateQuizContainer, showing the state store being passed.

screenshot demonstrates, the state has been passed to the component correctly, meaning the connection succeeded. *Success.*

15.2.2 Add Question

This ensures that the user is able to succesfully add a new question to the current quiz when the add question button is pressed.
As the two pictures indicate, after pressing the add question button, a new question was added to the system, and this was reflectd in the label unde the question title. *Success.*



iPad 16:42 77%

Default questions

1. I'm the question title - tap to edit me!

I'm the question title - tap to edit me!

Question 1 out of 1 in the Default category • Question 1 out of 1 in total

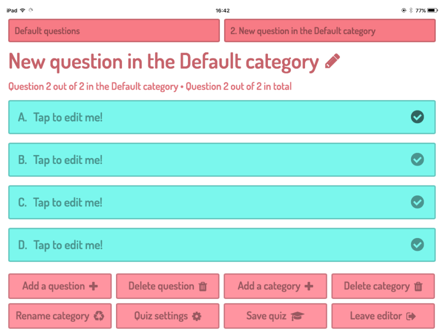
A. I'm the first possible answer!

B. You can edit any of us by tapping on our text.

C. See that bold check mark? It means I'm the correct answer.

D. Tapping on another check mark will make that the correct answer.

(a) Before



iPad 16:42 77%

Default questions

2. New question in the Default category

New question in the Default category

Question 2 out of 2 in the Default category • Question 2 out of 2 in total

A. Tap to edit me!

B. Tap to edit me!

C. Tap to edit me!

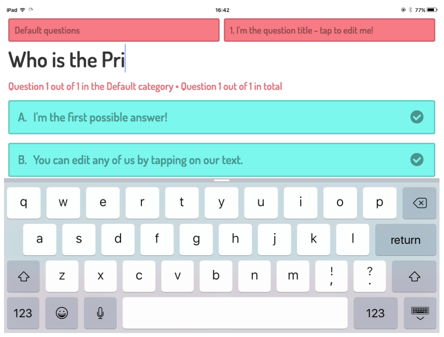
D. Tap to edit me!

(b) After

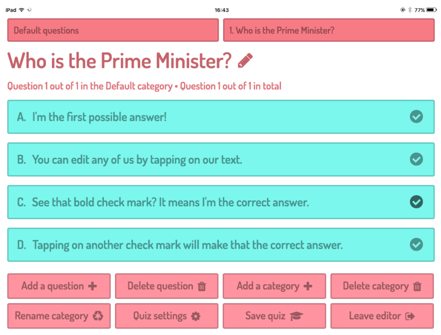
Figure 9: Adding a question to the quiz.

15.2.3 Edit Question

This ensures that the user is able to succesfully edit a question in the current quiz.


Default questions
1. I'm the question title - tap to edit me!
Who is the Pri
Question 1 out of 1 in the Default category • Question 1 out of 1 in total
A. I'm the first possible answer!
B. You can edit any of us by tapping on our text.

(a) During


Default questions
1. Who is the Prime Minister?
Who is the Prime Minister?
Question 1 out of 1 in the Default category • Question 1 out of 1 in total
A. I'm the first possible answer!
B. You can edit any of us by tapping on our text.
C. See that bold check mark? It means I'm the correct answer.
D. Tapping on another check mark will make that the correct answer.
Add a question
Delete question
Add a category
Delete category
Rename category
Quiz settings
Save quiz
Leave editor

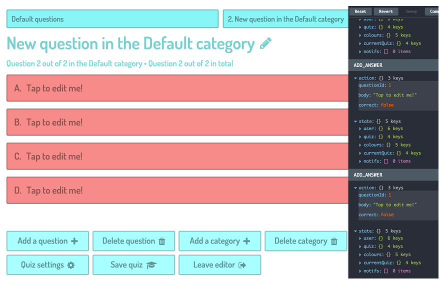
(b) After

Figure 10: Editing a question to the quiz.

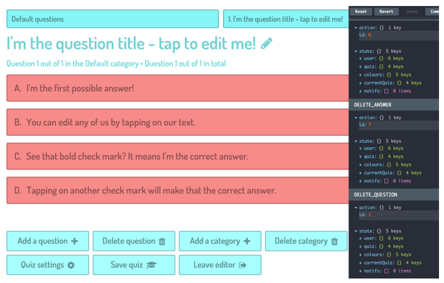
15.2.4 Delete Question

This ensures that the user is able to succesfully delete their questions from the current quiz.

Note: *In order to prove that the question has actually been deleted, the screenshot captures the developer tools used in the creation of the application; an action showing that the question has been deleted is clearly visible.*



(a) Before


Default questions
1. I'm the question title - tap to edit me!
I'm the question title - tap to edit me!
Question 1 out of 1 in the Default category • Question 1 out of 1 in total
A. I'm the first possible answer!
B. You can edit any of us by tapping on our text.
C. See that bold check mark? It means I'm the correct answer.
D. Tapping on another check mark will make that the correct answer.
Add a question
Delete question
Add a category
Delete category
Quiz settings
Save quiz
Leave editor
DELETE_ANSWER
DELETE_QUESTION

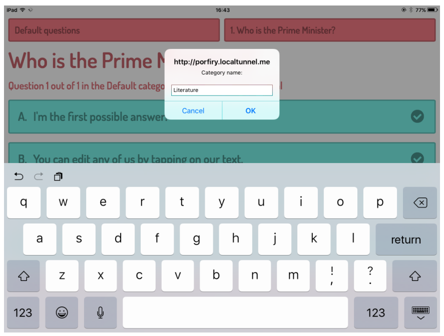
(b) After

Figure 11: Delete a question in the quiz.

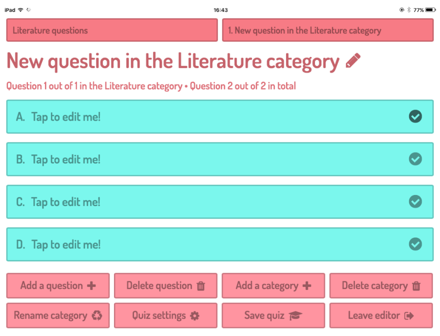
The second screenshot shows that a delete question action was passed to the quiz reducer, resulting in the question being successfully deleted; this is also reflected in the labels. *Success.*

15.2.5 Add Category

This ensures that the user is able to succesfully add a custom category to the current quiz. As expected, a dialog to enter a category name is shown when


Default questions
1. Who is the Prime Minister?
Who is the Prime M
Question 1 out of 1 in the Default catego
http://porfiry.localtunnel.me
Category name:
Literature
Cancel
OK
A. I'm the first possible answer
B. You can edit any of us by tapping on our text.
return

(a) During



(b) After

Figure 12: Adding a category to the quiz.

the add category button is pressed, and the new category is added successfully. *Success.*

15.2.6 Delete Category

This ensures that the user is able to succesfully delete their custom categories in the current quiz. As expected, a dialog to enter a category name is shown when

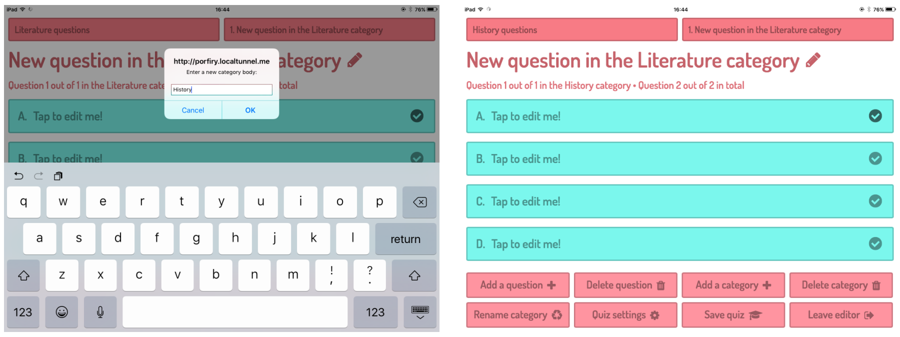
(a) Before (b) After

Figure 13: Deleting a category from the quiz.

the add category button is pressed, and the new category is added successfully. *Success.*

15.2.7 Rename Category

This ensures that the user is able to succesfully rename their custom categories in the current quiz. As expected, a dialog to enter the new category name is shown when the rename category button is pressed, and category is renamed successfully. *Success.*


Literature questions

1. New question in the Literature category

New question in the Literature category

Question 1 out of 1 in the Literature category

http://porfiry.localtunnel.me

Enter a new category body:

History

A. Tap to edit me!

B. Tap to edit me!

History questions

1. New question in the Literature category

New question in the Literature category

Question 1 out of 1 in the History category • Question 2 out of 2 in total

A. Tap to edit me!

B. Tap to edit me!

C. Tap to edit me!

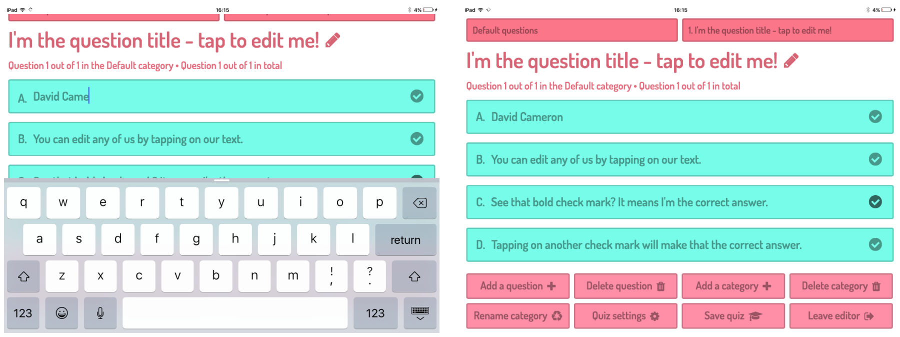
D. Tap to edit me!

(a) During (b) After

Figure 14: Renaming a category in the quiz.

15.2.8 Edit Answer

This test ensures that the user is able to edit the body of an answer in the current quiz.


I'm the question title - tap to edit me!

Question 1 out of 1 in the Default category • Question 1 out of 1 in total

A. David Came

B. You can edit any of us by tapping on our text.

Default questions

1. I'm the question title - tap to edit me!

I'm the question title - tap to edit me!

Question 1 out of 1 in the Default category • Question 1 out of 1 in total

A. David Cameron

B. You can edit any of us by tapping on our text.

C. See that bold check mark? It means I'm the correct answer.

D. Tapping on another check mark will make that the correct answer.

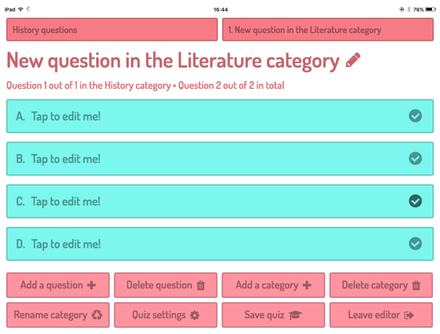
(a) During (b) After

Figure 15: Marking an answer in as correct.

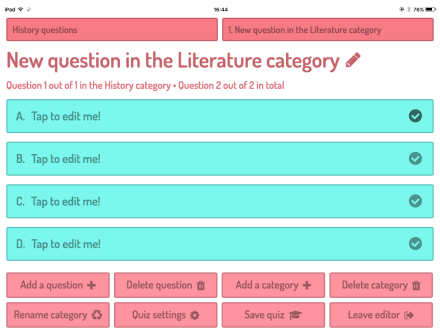
As expected, the application allowed for the answer body to be edited, and then persisted this change after leaving the edit mode. *Success.*

15.2.9 Mark Answer as Correct

This ensures that the user is able to succesfully mark an answer as correct in the current quiz.
As expected, the correct mark moved from the third to the first question, meaning that it was marked as correct. *Success.*


History questions
1. New question in the Literature category
New question in the Literature category
Question 1 out of 1 in the History category • Question 2 out of 2 in total
A. Tap to edit me!
B. Tap to edit me!
C. Tap to edit me!
D. Tap to edit me!

(a) During


History questions
1. New question in the Literature category
New question in the Literature category
Question 1 out of 1 in the History category • Question 2 out of 2 in total
A. Tap to edit me!
B. Tap to edit me!
C. Tap to edit me!
D. Tap to edit me!

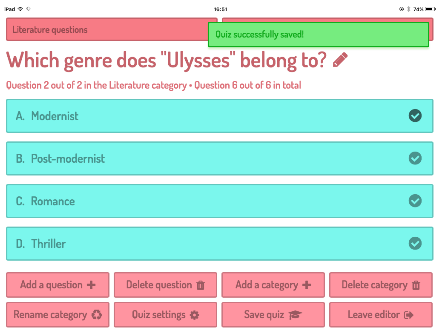
(b) After

Figure 16: Marking an answer in as correct.

15.2.10 Save Quiz

This test ensures that the user is able to save their quizzes to the database.

(a) During


Literature questions
Quiz successfully saved!
Which genre does "Ulysses" belong to?
Question 2 out of 2 in the Literature category • Question 6 out of 6 in total
A. Modernist
B. Post-modernist
C. Romance
D. Thriller

(b) After

Figure 17: Saving a quiz to the database.

As expected, the correct mark moved from the third to the first question, meaning that it was marked as correct. *Success.*

15.3 Play Quiz Test Runs

15.3.1 Connects to State Store

This test ensures that the component can connect to the state store.

Expected result	**Works**	**Signature**
Displays ”Who is the French Premier?” at 0-10 seconds	Yes	
Displays ”Who won the 1960 World Cup?” at 10-20 seconds	Yes	
Displays ”What is the capital of Iceland?” at 20-30 seconds	Yes	

Table 9: My caption

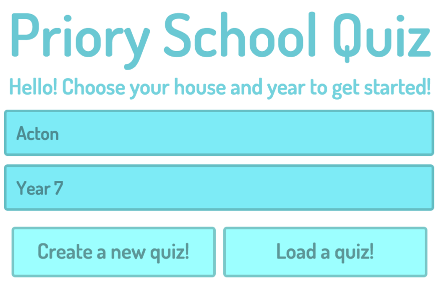
15.3.2 Move Questions on Time

This test ensures that the quiz moves to the correct questions at the correct time, using the test quiz specified in the test plan. Due to the difficulty of taking valid screenshots of this process, a table is included below with room for a teacher to sign as confirmation that this functionality is working.

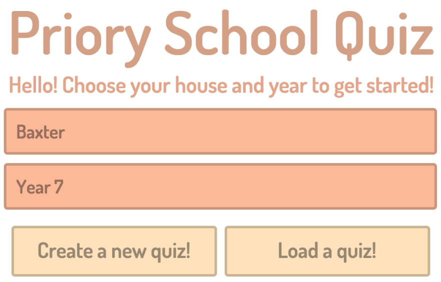
As can be witnessed, the quiz moves to the correct questions at the correct time, meaning that the test has passed. *Success.*

15.4 Colour Test Runs

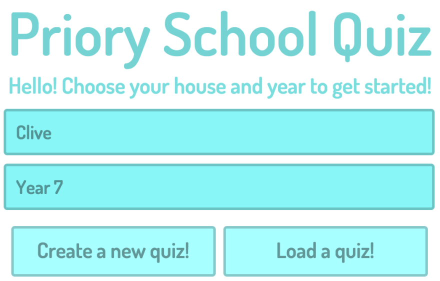
Depending on which house the user belongs to, the system should theme itself in that house's colours.


Priory School Quiz
Hello! Choose your house and year to get started!
Acton
Year 7

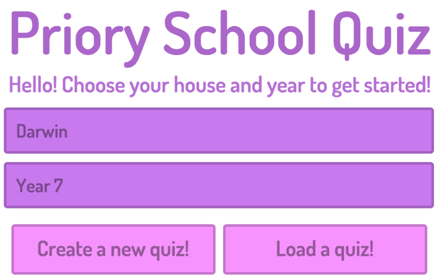
(a) Acton


Priory School Quiz
Hello! Choose your house and year to get started!
Baxter
Year 7

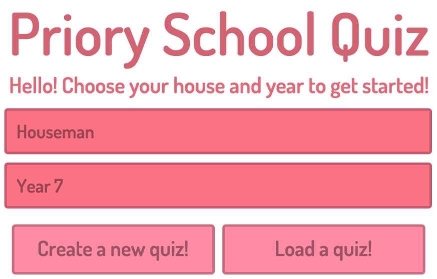
(b) Baxter


Priory School Quiz
Hello! Choose your house and year to get started!
Clive
Year 7

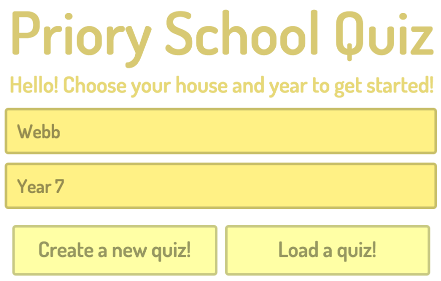
(c) Clive


Priory School Quiz
Hello! Choose your house and year to get started!
Darwin
Year 7

(d) Darwin


Priory School Quiz
Hello! Choose your house and year to get started!
Houseman
Year 7

(e) Houseman


Priory School Quiz
Hello! Choose your house and year to get started!
Webb
Year 7

(f) Webb

Figure 20: The different house themes.

As the above images demonstrate, the system correctly themed itself in different colours depending on which houses were chosen. Acton was blue, Baxter orange, Clive a shade of turquiouse, Darwin was purple, Houseman red, and Webb was yellow; all according to the test plan. *Success.*

15.5 Results Test Runs

This section tests that the results screen works correctly.

15.5.1 Connects to State Store

This test ensures that the component can connect to the state store.

15.5.2 Display Correct Results

This test ensures that the results screen shows the correct results.

15.6 API Test Runs

These are the tests for the restful API. The API acts as a means of communicating with the database. In order to do so, certain requests - either GET, POST, PUT or DELETE - are sent to an address; in this case, */api/quizzes*. The API then looks at the request type that was sent, along with any other data that came with it, and performs the appropriate operation, be that showing a list of all quizzes, showing a specific quiz, or updating a quiz.

15.6.1 api.spec.js

```
import chai from 'chai';
import mongoose from 'mongoose';
import request from 'supertest';
import sampleQuiz from './helpers/sampleQuiz';

describe('Restful API', () => {
  // Establish a connection to the testing database.
  before(done => {
    mongoose.connect('mongodb://localhost:27017/porfiry');
    done();
  });

  const url = 'http://localhost:5000';

  describe('GET /quizzes', () => {
    it('should respond with JSON', done => {
      request(url)
        .get('/api/quizzes')
        .set('Accept', 'application/json')
        .expect('Content-Type', /json/)
        .expect(200, done);
    });
  });

  describe('POST /quizzes', () => {
    it('should save a quiz to the database', done => {
      request(url)
        .post('/api/quizzes')
        .send(sampleQuiz)
        .expect('Content-Type', /json/)
        .expect(200, done);
    });
  });

  describe('GET /quizzes/:id', () => {
    it('should return a specific quiz from the database', done => {
      request(url)
        .get('/api/quizzes/563a78030474e73d275408d7')
        .set('Accept', 'application/json')
        .expect('Content-Type', /json/)
        .expect(200, done);
    });
  });

  describe('PUT /quizzes/:id', () => {
    it('should edit the properties of a quiz', done => {
      request(url)
        .put('/api/quizzes/563a78030474e73d275408d7')
        .send({ title: 'Test Quiz' })
        .set('Accept', 'application/json')
        .expect('Content-Type', /json/)
        .expect(200)
        .end((err, res) => {
          if (err) return done(err);

          let quiz = JSON.parse(res.text).quiz;

          if (quiz.title === 'Test Quiz') {
            done();
          } else {
            done(err);
          }
        });
    });
  });
});
```

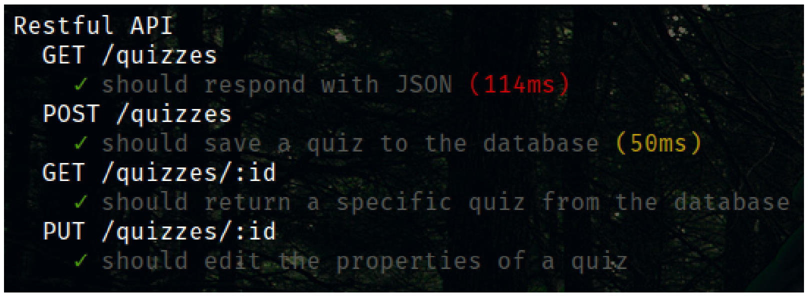
Listing 54: Unit tests for the API.

As can be seen by the little green ticks, all the different HTTP methods passed, proving that the API is working correctly, and thereby allowing requests to be sent from throughout the application. *Success.*

16 Acceptance Testing

The system was passed to a friend for testing, with the request that he note down sections that he both liked and disliked. He had the following positive comments about the application:

- The different colour schemes are effective, and are a useful differentiator



```
Restful API
  GET /quizzes
    ✓ should respond with JSON (114ms)
  POST /quizzes
    ✓ should save a quiz to the database (50ms)
  GET /quizzes/:id
    ✓ should return a specific quiz from the database
  PUT /quizzes/:id
    ✓ should edit the properties of a quiz
```

Figure 21: Test outcome for api.spec.js 1

between the houses. They also give the program a nice, fun feel, which fits in with the audience.

- It's very easy to make a quiz, as all the buttons are clear, with all the functions labelled. I think it would be suitable for someone who isn't very skilled with technology.
- I really like the bouncing animation on the login screen.
- The timer in the play quiz screen makes it really exciting to play on; there's a lot of tension.

He also had the following, less positive, notes:

- At first I didn't know how to change the question title. You could make this a bit clearer by adding a box around it, or something.
- When there are a lot of people on the quiz, the program slows down quite a lot, and the timer bar is very jittery.
- It's a shame that there aren't any live visualisations to see what other people are answering.

As can be gleaned from the notes above, the tester's overall opinion of the system was strong, especially in the quiz creator interface. This is not surprising, as a large amount of time was spent making this work properly. The quiz playing interface lacks the same amount of polish, and this was picked up by the tester, though he did comment that it was sufficiently tence and exciting due to the inclusion of the live countdown timer.

Part V

Evaluation

This section contains an extensive evaluation of different parts of the application, and the processes that went into its development. A number of areas have been evaluated, including the test results. In order to qualify the testing process as acceptable, it must be evaluated to ensure that the majority of the testing was successful; focus will be placed on each of the individual areas of testing that were performed, namely: the unit tests for each individual component, the acceptance testing, and the full system testing. By evaluating all these areas, it will be possible to say whether the test strategy laid out was a success, and the degree to which this is true.

Also evaluated is whether the system meets the original objectives, those objectives outlined at the beginning of the project, before development began, and that took into account the research performed in the analysis section. For the system to considered a success, it must meet all or most of these objectives; this section of the evaluation attempts to determine whether or not this is the case.

The evaluation will also take into account the evaluation criteria - these criteria are additional to the original objectives, and lay out several supplementary aspects that the system should meet to be viewed in a wholly positive light; it also takes into account areas like the length of time spent on actually developing the system. This section of the evaluation therefore evaluations these critera.

Potential future improvements will also be explored. As with any system, there are aspects of it that could be improved. This section outlines several of these, taking into account whether or not they are feasable, and gives methods that could be used to implement them within a reasonable time frame.

17 Evaluation of Original Objectives

Provide staff with an attractive user interface with which they can create, update and delete quizzes. In this respect, the system can be considered a success. Provision has been made for unlimited to questions to be created; the user can press the *Add Question* button as many times as they like. As specified, each of these questions has four answers, only one of which can be set as correct. The ability to define categories was also implemented, allowing the quesions to be split up into natural groups. The settings panel allows for the staff producing the quiz to define a common question length, thereby setting the duration in which students can answer the question before the quiz moves on. This same panel also allows a date and time to be set, at which point the quiz will be scheduled. This schedule works, as shown in the tests.

Allow students of the school to answer the quiz in real time and compete against one another. By and large, the objectives in this section were met. Using a simulator that spawns n number of connections (effectively students), it was possible to check whether the quiz supports up to 1000 students at the same time without failing; the simulator proved that this is the case, and, indeed, that the system is able to support ten times that amount with no failings. Another objective was that the system should begin at the specified time, no matter how many students are connected. Due to the way the two major parts of the system, the server and the client, were written (they are decoupled, neither relying on the other), this objective was met: the system will simply move forward with each question, paying no regard to who is connected. It was decided that, rather than have a five minute grace period to allow those who were late to still take part in the quiz, the system would instead allow students to join at any time, and would simply take them to the same point in the quiz as everyone else. This promotes a "hive" mentality, whereby the importance of each individual node (student) is reduced; this reduces the chance of the final scores being innaccurate if the quiz is disrupted in any way. As implied above, the system was designed so that every connected client should always be at the same point in the quiz: in the same category, on the same question, with the same time remaining. This "on-rails" approach means that the integrity of the scores can be guaranteed, and that teachers need simply sit back and watch as the system takes over. Again, this aspect of the system was a success, the only small issue being that the timer bar jumps around a little at first when joining a quiz mid-way through a question. The interface for the quiz also meets the objectives: it contains all the required elements, including the current category name, question body, and the four answers; as well as a live timer bar that displays how much time is left to answer the question. The system also reports the final results at the end of the quiz, displaying the winning house and the points earned by each house, in the form of a bar chart.

Display a real time visualisation of how other participants in the quiz are answering. Due to time constraints and technical difficulties, this was the only aspect of the system that was not implemented. The difficulty lay mostly in making the process efficient enough for real-world use: in order for the visualisation to work, every time someone chose an answer, every single connectec client would have to be notified of this. The real-time infrastructure makes use of the websocket protocol, so the issue was not in the size of the data being sent and received; rather, its quantity. In a quiz containing ten questions, each 10 seconds long, and with 1000 participants, 10,000 packets of data would be received by every client, with 1000 spread across ten seconds; and, more likely, half of these packets would be received across a two second period, as the majority of students choose an answer. For every one of these packets, the visualisation would have to be updated; therefore, within a 1-2 second period, the client would be updating 500 times in a single second. This is obviously an issue, and would result in so much lag that the system would be unusable. Time constraints unfortunately prevented alternative methods being explored.

Work effectively across a range of different devices and display types. The system was designed and developed for the web, so this objective was easy to meet. The application will run effectively on all modern browsers (IE >= 10, all up to date WebKit browsers), with no missing functionality. The server aspect of the system, controlling the real-time infrastructure, can be deployed to any system for which Node binaries are available; effectively any system one could care to use. The client system need not be overly powerful; the quiz ran smoothly even on a first generation iPad Mini, with only 512mb of RAM. The school will use the system mainly on iPads, so a resolution of 1024x768 produces the best results. However, effort has been made to optimise the system for different screen sizes, and it will work with full functionality even on much smaller or larger screen sizes.

Theme itself to match the house colours of the logged in user. This was again a success. As can be seen in the myriad screenshots of the application, the colours change correctly, according to the house of the user.

Speaking holistically, it would be remiss to state that the system did not meet the original objectives. Though it was disappointing that the live visualisations were not completed in time, practically every other aspect of the system was, and to a very high quality.

18 Evaluation of Evaluation Criteria

Performance of the system should be considered. The system should be able to cope with at least 1000 connections at any one time, all communicating with each other. The system easily met this criteria, at least on the server side. Due to the impracticality of testing the application on 1000 individual iPads, a small additional program was written that spawned socket nodes, each one simulating an iPad / other device. With the simulator generating answer packets at a rate of roughly 300/s, roughly what could be expected from a live quiz, the system was easily able to cope with the stress, generating no lag or latency in working out the updated scores.

Staff and students should find the user interface simple and easy to use. A deal of effort was put into making the system as user friendly as possible. A consistent font, Dosis, is used throughout the entire application, and this fits well with its general light, colourful nature. Bright colours are used all the way throughout, making it stand in contrast to the usually dull and drab systems that fill the market. Additionally, subtle animations are scattered throughout, making the system more approachable to those not used to it. To further aid in this, the default quiz first displayed in the creator section acts as a tutorial, instructing users how to use the system. The actual interface was also designed with simplicity in mind: it consists mainly of a series of buttons, with labels like “Add a question” that make their action very obvious.

The application code should be as high quality as possible, making use of functional paradigms where possible. A great deal of effort was spent making the code achieve this goal. One of the key tenets that has been followed is immutable state. In a program or language that makes use of global mutable state, the code can very quickly become unmanageable. Functions from any part of the application can mutate the state, making it difficult to keep track of current values at any one time. This becomes an especially big issue when using a Virtual DOM on the client side (needed to handle the large number of updates resulting from the number of connected sockets), as unnecessary renders may be invoked, causing large performance issues. Instead, a single immutable state tree has been used, from which "smart" components can read from, and then pass down data to purely functional components. Many of these definitions are by definition pure: given the same parameters (passed by-value as properties from parent components), they will always return the same result. The state tree can only be updated by dispatching "actions" - pure packets of data that describe what has happened. For example, when a new question is generated, the following structure is passed to the state reducer:

```
{ type: 'ADD_QUESTION', body: action.body }
```

. The reducer then returns a new copy of itself, with the new question appended to an array; in this way, immutability is achieved.

Stability is an important factor, and it will also be considered. If the system is constantly prone to crashing, it could not be called stable Again, the system meets this criteria. Extensive testing has been performed, ensuring that all edge cases that might make the system crash have been accounted for. As noted above, the application is able to cope with a high degree of traffic without crashing. The system has been writted in such a way that it is fully automated. It will simply take the quiz passed to it, schedule jobs corresponding to the answer times, and then let them run through. In this way, maintenance is also lowered: the system can just be left running, and it will not run into problems. The environment in which it is intended, the school, is very controlled, making it even less likely that it will meet with an exception it does not know how to handle.

The cost of the system to run should also be taken into account Though it is difficult to give a precise cost for the running of the system, due to fluctuations in hosting costs, effort has been made to make it as performant as possible. All of the stress testing for the backend was performed on a comparatively weak machine, compared to even the cheapest servers, and performance was more than acceptable. As such, the school need only rent the smallest machine, which, as was stated in the interview section, will cost around £0.33 per annum; taking this into account, this criteria is an easy success.

Overall, it would be correct to say that the evaluation criteria were met, and this part of the system was another success.

19 Future Improvements

There are a tremendous number of ways in which the system can be improved. Perhaps the most obvious is fully implementing the live visualisation system, whereby the current status of each house and form is relayed in real time to each individual user. As set out in the evaluation above, there are a number of factors that make this relatively difficult to achieve, but it is definitely possible.

One possible feature could be the ability to see if a form or house has consistently won over several quizzes. This way, a running tally could be kept of the best form or house, increasing competitiveness between houses, one of the system's inital goals. This could be taken further with a streak system: for every n number of quizzes the house has won, they could receive n extra house points. Such as system would also allow individual forms that have done especially well to be reported. Of course, such an approach could make it difficult for other houses to catch up and beat the house with the highest streak: by winning just a few quizzes at first, their score would eventually snowball, placing them at an advantage during subsequent quizzes.

To solve this, a "bonus" system could be implemented. Every so often, special events, such as a knockout question between just two individual students, could be held, worth a large number of house points. These could take place at random intervals, perhaps between questions or categories,

A further change that might be considered is rewriting the application's structure and backend. The system currently utilises a fairly loose structure, with little coordination between files and directories. It would be beneficial to reimplement the backend in a language that enforces such a structure. A potential candidate for this might be Erlang, a language well suited for building highly concurrent, real-time applications, much like the quiz system. Roughly 40% of the world's telephone and SMS communications pass through Erlang powererd servers, giving it a well deserved reputation of stability. The langauge also more rigidily enforces a functional style of programming. Though a, largely successful, drive to stick to the paradigm was a key criteria in the current version of the system, there are still certain areas where the code more resembles that seen in an imperative or object-oriented system, or where the code could simply be refactored to be much simpler. Using a more functional language would help improve this.

Part VI

User Documentation

This section outlines how to install and make use of the application.

20 Installation

This section details how to install the application. The process is rather specialised, and the precise details vary depending on which cloud provider is being used to host the system, but by and large there is a process that can be followed.

Note: *Installation of the system is only possible on Unix based operating systems, such as a Linux distribution or OSX.*

20.1 Cloning the Repository

In order to get the system on your server, the git repository needs to be cloned. This process is simple: navigate to the folder in which the system will reside (the home directory is recommended), and run the following command:

```
git clone http://github.com/deerob4/porfiry
```

You will be asked for credentials; these will be supplied with the system.

20.2 Installing Dependencies

The next stage is to install the application's external dependencies. Without these, the system will not be able to run. Ensure that Node is installed on the server; if not, install it using the appropriate package manager. The minimum version of Node required is 4.2.3. To install the dependencies, run the following command:

```
npm install
```

«««< HEAD This will create an *npm_modules* directory in the root of the system's folder, which will contain all of the dependencies. The system will pull from this directory. ======= This will create an *npm_modules* directory in the root of the system's folder, which will contain all of the dependencies. The system will pull from this directory, so it is important that it is not moved. »»»> 8f16a150217d5effa08f86d621639ca80c0922fd

21 Usage

This section displays how to use the quiz system for it's intended purpose. The different parts of the system have been split up into different subsections for convenience.

21.1 Creating a New Quiz

Creating a new quiz is a very simple process. Select your house and year from the drop down lists on the menu, and then press the "Create a new quiz" button.

21.2 Editing a Quiz

The interface for adding questions and answers to a quiz is similar to other quiz creation systems you may have used in the past. When a quiz is first created, there is one question, with four answers - all of these can be edited.

21.2.1 Editing a Question

To edit a question's title, simply click or tap the current title, and type in your changes.

21.2.2 Editing an Answer

To edit a question's answer, simply click or tap the current answer, and type in your changes.

21.2.3 Marking an Answer as Correct

To mark an answer as correct, tap the tick icon at the end of the answer. The answer currently marked as correct will no longer be so.

21.2.4 Adding a Question

To add a new question, simply press the "Add Question" button.

21.3 Scheduling a Quiz

To schedule a quiz, perform the following steps:

1. Tap the “Quiz settings” button.
2. Enter the date and time you wish the quiz to begin.
3. Tap the “Save settings” button to exit the settings panel.
4. Tap the “Save quiz” button.

Your quiz has now been scheduled, and will take place at the set time.

21.4 Loading a Quiz

If you wish to load a quiz that has been created earlier, for further editing, press the “Load quiz” button on the main menu, and locate the quiz you wish to edit by its title and schedule date. From there, press the “Edit quiz” button, and you will be taken to the familiar quiz editing interface.

21.5 Deleting a Quiz

Deleting a quiz is also very simple. Press the “Load quiz” button on the main menu, and locate the quiz you wish to edit by its title and schedule date. From there, press the “Delete quiz” button, and the quiz will be deleted.

Note: *If the quiz has been scheduled to play, deleting the quiz will cancel the schedule, and the quiz will no longer take place.*