UQDraw

PROJECT PROPOSAL REPORT

Prepared for DECO3800 Sprint 0 by Group Orestae

Response To Brief

The Client

University of Queensland eLearning Team.
The eLearning team at the University of Queensland is responsible for the management of a suite of eLearning tools used throughout the university. The other core responsibility of the team, given the dynamic nature of the eLearning field, is to evaluate and integrate new eLearning technologies into their current portfolio. The team is currently looking to have a new eLearning tool developed that will help to facilitate active learning in large classes.

Project Overview

In discussion with the client, it was noted that the eLearning team has previously been involved in the development of a couple of other applications for active learning (UQPoll, and Wordcloud). The types of active learning that these tools allow for is, however, quite restricted. This is where the current tool they are looking to have developed—UQDraw—comes into play. The primary objective of UQDraw is to allow students to submit answers to problems that require more free-form responses. Common examples of such problems include graphing, equations, and diagrams. The secondary objective is to provide a method for lecturers to be able to analyze and improve their teaching methods over time by reviewing responses to these problems provided by the students.

The client has a few broad functional requirements for the application, but intends for the specifics to be decided upon by the development team. These broad functional requirement are:

- Students can submit answers to problems requiring free-form responses such as drawing.
- Lecturers can ask students to solve problems in lecture by 'publishing' them on the screen.
- Lecturers can select a thumbnail of a particular response to fullscreen during the lecture.
- Responses are saved and can be analyzed at a later date by the lecturer.

The one essential non-functional requirement that the client has highlighted comes from their past experiences with the two aforementioned active learning apps. When discussing the outcomes of these prior projects with the client, it became apparent that the largest sticking points with the applications flowed from the fact that many lecturers have poor competence with technology. As a result, it is crucial for UQDraw to be extremely easy to use so that the adoption of the tool is not hampered by this lack of tech savvy on the part of the lecturers.

Lastly, owing to the client being an entity within a university, they have two special requirements:

- The colour scheme and fonts should follow the branding guidelines for UQ.
- The application should use UQ's Single Sign On feature for user authentication.

Our Proposed Solution

Overview

Our solution will be individually tailored to the two primary user groups for UQDraw: lecturers, and students. These two user groups have very different tasks, user characteristics, and will be using different devices when interacting with UQDraw. As a result, the form and functionality of the application that will best meet the needs of each of these distinct user groups will be meaningfully different. In this proposal we will often refer to the lecturer's application and the student's application to reinforce this distinction.

At a high-level, the lecturer's application will have the following behaviour:

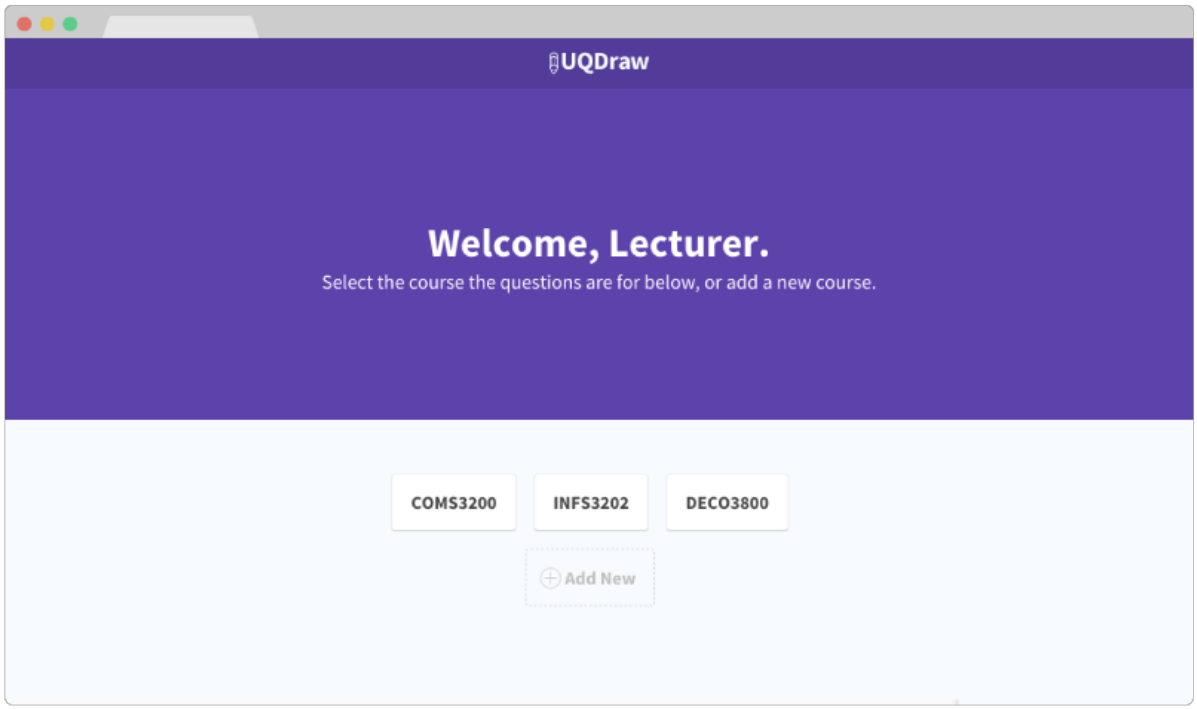
- Allows the lecturer to compose and save question/problems before a lecture.
- Allows the lecture to pose questions/problems to students during a lecture.
- Allows responses to be gathered in real time and displayed on the lecturer's screen.
- Allows particular responses to be selected and full-screened during the presentation.
- Saves all responses, and allows them to be analyzed at a later date.
- Perform optimally on laptops and desktops.

At a high-level, the student's application will have the following behaviour:

- Allows the student to respond to the question posed by the lecturer during the lecture.
- Allows the student to compose a free-form response using graphical tools (predominantly drawing tools)
- Allows a student to submit a response
- Perform optimally on mobile devices with touchscreens (predominantly smartphones and tablets).

Lecturer's Application — A More Detailed Look

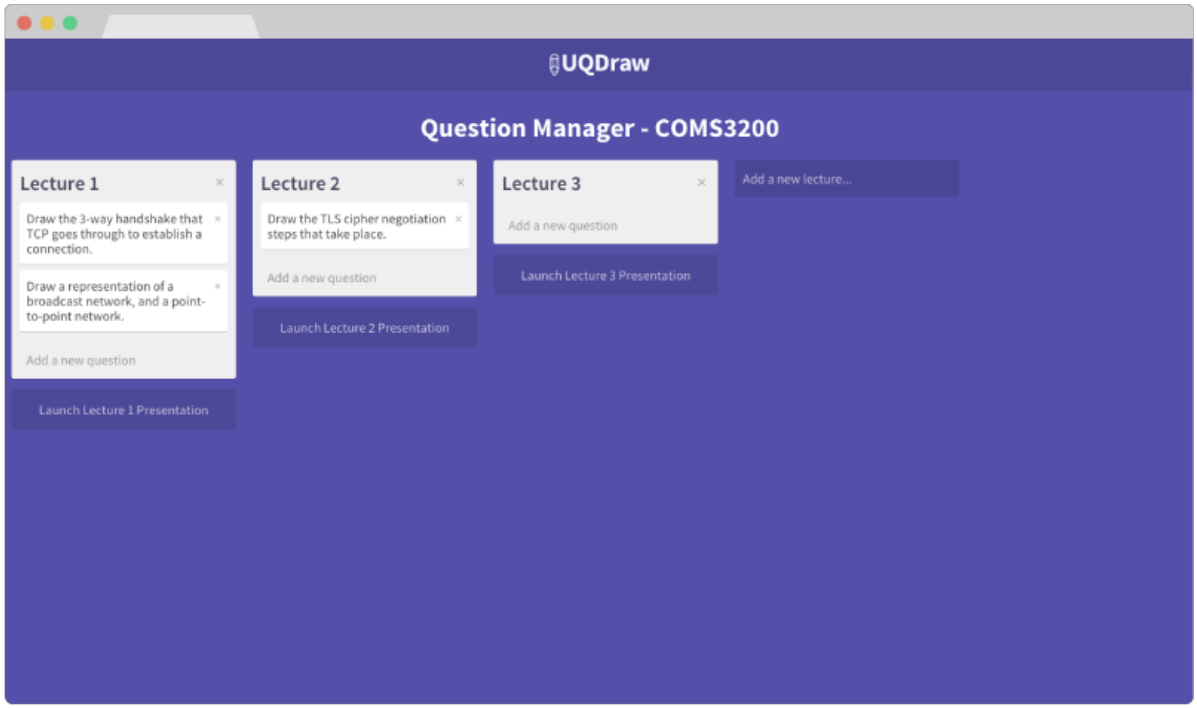
Course List View



Course List View — The view that the lecturer is presented with upon first logging in.

Upon first logging in, the lecturer is presented with a simple dashboard. This dashboard lists any courses that the lecturer is currently using UQDraw for. In keeping with the requirement that the application be extremely easy to use, this interface is clean, uncluttered and friendly. There is only a single choice the user needs to make: select a course or add a new one.

Course Questions View



Course questions view allows for managing questions for a particular course.

After the lecturer select a course they are taken to the course questions view. This view allows them to compose questions for upcoming lectures. When new questions are composed they are added to one of the lists that the lecturer has created. This allows the lecturer to easily lay out the questions lecture-by-lecture. The view also allows the questions to be rearranged via drag-and-drop interactions.

On the day of the lecture, the lecturer will navigate to this view and click `Launch Lecture Presentation` beneath the relevant list. This will take the lecturer into the presentation mode for the particular list of questions.

Presentation View

Presentation View — the view shown when a lecturer is presenting question to a class.

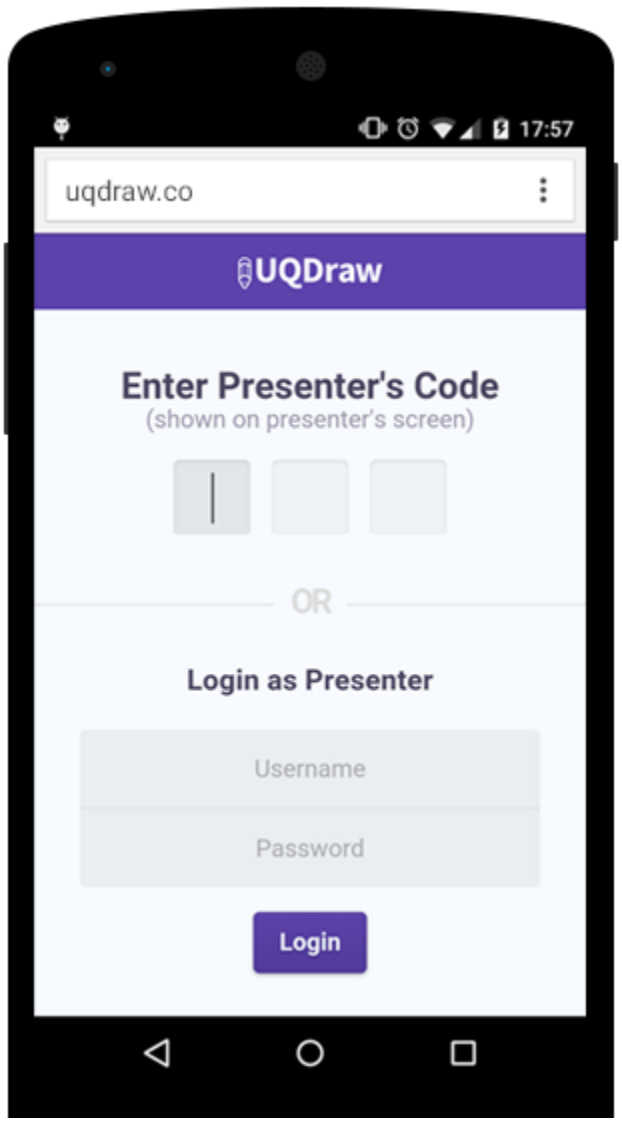
The presentation view is what the lecturer will show to the class in order to pose questions and gather responses. The presentation view includes simple instructions for the students to navigate to the applications URL (uqdraw.co) and to enter the current presenter's code (3FA). This will allow the students to respond to the question on their own devices.

The list of questions for the lecture from the course questions view is available on the right-hand side of the screen. The main panel will present the question selected on the right-hand side.

To start accepting responses the lecturer clicks 'Start Accepting Responses'. This will start a counter and students will now be able to submit responses. As responses are received, the response thumbnails (situated at the bottom of the view) will start becoming populated with the student responses.

Student's Application — A More Detailed Look

The forthcoming few pages will explore the core interfaces of the application that the student user will interact with. The purpose will be to illustrate the general concepts and design ideas to give a sense of what the application looks like from the user's perspective. In line with this, we will refrain from discussing many of the auxiliary interactions that are possible within the interfaces.



Login View

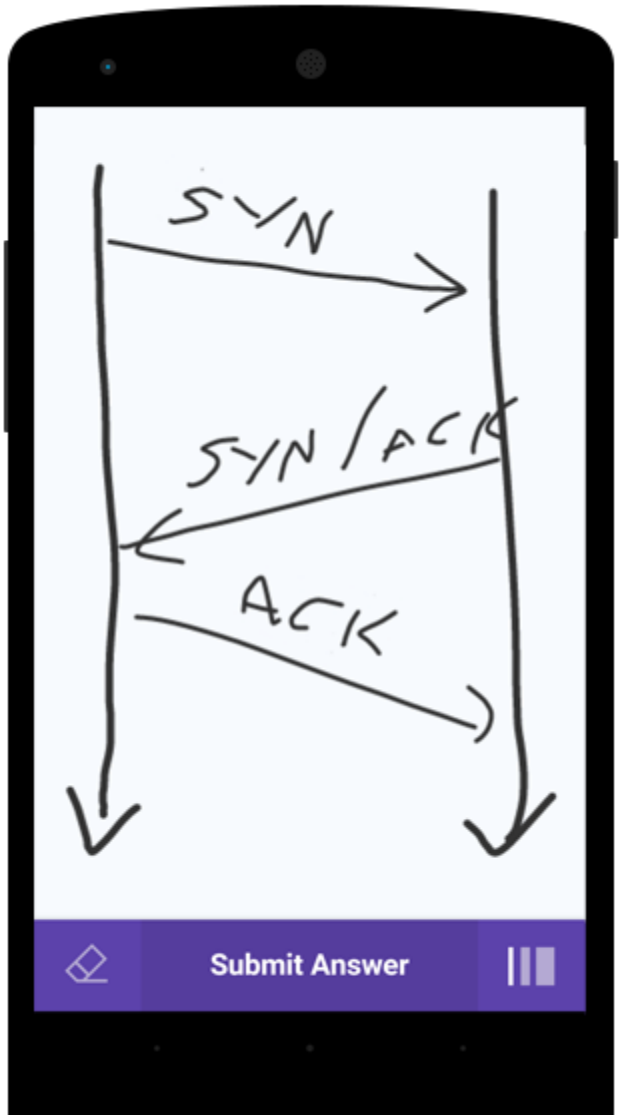
As previously shown, when the lecturer presents a question to the students during a lecture, the presentation screen directs the students to go to the website and then to enter a code. This three symbol code (eg. 3FA) allows the student to easily navigate to the response screen for the current presentation.

Both the domain name and the presenter codes have been consciously designed to minimize the length. This is an important consideration as typing large codes or URLs is highly error prone on smartphone keyboards; leading to plenty of user frustration.

Question View

When the student finishes entering the presenter's code on the previous view they will be taken to the question view for the current presentation. This view will be synchronised with the lecturer's presentation. This means that the question that will get displayed to the student will match the question that the lecturer has started accepting responses to.

The view allows the student to read the task and then tap to begin drawing their response on the canvas that lies beneath the question overlay.



Drawing View

To respond to the task that the lecturer has posed, the student will be able to draw in a free-form manner on a digital canvas. Typically this will be done with a finger, but a stylus input is supported as well.

To keep things simple and straight-forward we have opted for an interface with a minimal set of design tools. Specifically, the user is given three pen widths that can be cycled through, and a rubber. We are of the belief that the initial toolset should provide the minimum necessary functionality. Over time the usage patterns and desires of the users can be used to add new tools if there proves to be sufficient benefits to be gained through the added complexity.

One of the main constraints with effectively being able to draw on a mobile device is screen real estate. To optimize this, the student will also be able to work with the canvas in fullscreen mode.

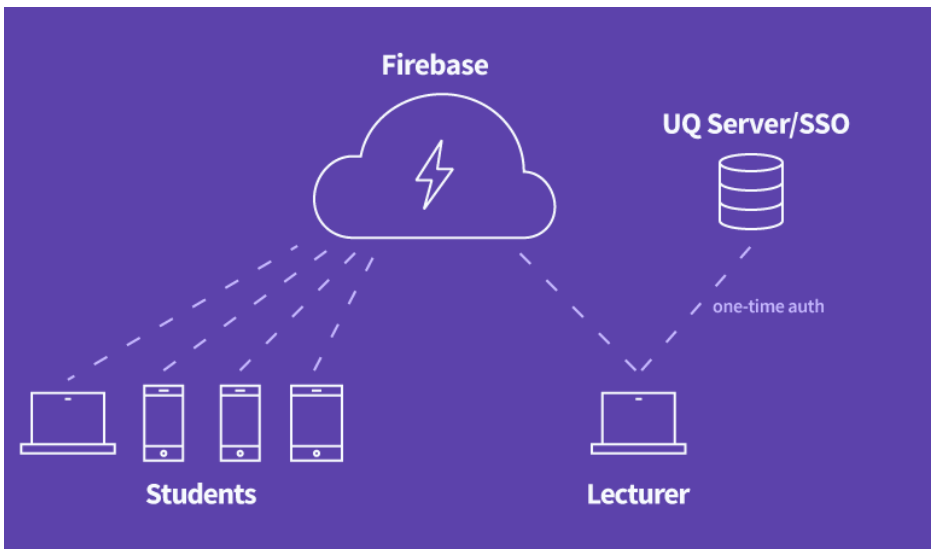
Upon completing their submission, the user can tap ‘Submit Answer’ to submit their response to the lecturer.

Technical Choices

As mentioned, one of the key qualities highlighted by the client during discussion was that the app needed to be easy to use. One of the most important facets in making something easy to use, is to make it easy to access. To this end, we believe building a **native web application** is the best fit for this project. The web application runtime works on all common devices, and the application can be run without having to go through any download and installation process. This will remove any barriers to getting started, both for students and for lecturers. As an additional benefit, this also lowers the development and maintenance costs for the project as bespoke applications need not be made for all major mobile operating systems.

One usability concern, however, that is often present with web apps, but rarely with native mobile applications, is that of interface responsiveness. Web apps built in a traditional manner where large changes in state require loading a new page feel extremely unresponsive and laggy to a user. This problem is especially pronounced on mobile devices due to the increased network latency experienced by these devices. To surmount this problem we will be building the application using **React**. React is a JavaScript library for building highly-performant web application interfaces, and removes the need to perform any page reloads. This framework is also not limited to rendering to the browser, so can be used to power native mobile applications. This would allow for native mobile apps of UQDraw to be deployed in the future if the need arises, without rewriting core application logic from scratch.

The clients for both the lecturer and student will be backed by a real-time datastore called **Firebase**. This is a hosted data-store designed specifically for building web and mobile apps that have real-time communication requirements. With our proposed application, there will be real-time synchronization between the student and lecturer apps during the presentation, which is why this technology is a good fit for the project. This also eliminates the need for writing and maintaining backend server code for performing real-time sychronization between thousands of devices, and allows for seamlessly scaling up and down in response to demand, thus eliminating all scalability and performance concerns that would crop up with a traditional on-premise solution.



Overview of system architecture

Work Breakdown

The project has been broken down into six phases: Initialization, Planning, Design, Development, Testing and Deployment phases. This Work Breakdown is used to assist the Project Manager in ensuring the team is on schedule and having a bird's eye view of the project would also help the Project Manager in planning efficiently for the coming weeks.

A Man-hour system is used to calculate the hours. Take the Duration(hrs) of the task and multiply it by the number of people needed for the task.

For example, a job requires 2 programmers to put in 3 hrs each.

2 programmers x 3 hrs = 6 man hours

The *Resource* system that we used is very specific in identifying which member is required for the task, this will help the Project Manager in assigning tasks appropriately. This also allows the team to work concurrently; more efficiently.

Additionally, we do not want to be paying the whole team to come up with a task that can be completed by an individual. This is also a realistic approach when we have to estimate the costs of 'paying the salaries' to the different individuals.

Tasks	Duration (hrs)	Man-hours	Resource	Start Date	End Date
Phase 1: Project Initialization	**16**	**66**		**10-Mar-15**	**20-Mar-15**
Group Meetings	2	12	All members	10-Mar-15	20-Mar-15
Project Charter	6	6	Project Manager	10-Mar-15	16-Mar-15
Role Assignment	1	6	All members	10-Mar-15	13-Mar-15
Meet up with Client	1	6	All members	10-Mar-15	13-Mar-15
Approval from Client					
Brainstorm	3	18	All members	16-Mar-15	20-Mar-15
Document and Finalize Plan	3	18	All members	16-Mar-15	20-Mar-15
Phase 2: Planning/Proposing	**184**	**524**		**23-Mar-15**	**5-Jun-15**
Group Meetings	20	120	All members	23-Mar-15	1-Jun-15
Initial Proposal	10	10	Project Manager	23-Mar-15	27-Mar-15
Task Allocation	1	6	All members	23-Mar-15	23-Mar-15
Sprint Zero	20	35		23-May-15	13-Apr-15

Response to Brief	5	5	Project Manager		
Work Breakdown	5	20	Programmer x 4		
Quote	5	5	Project Manager		
Risk Analysis	5	5	Project Manager		
System Architecture	5	20	Programmer x 4	14-Apr-15	20-Apr-15
Storyboards	10	10	Graphic Designer	20-Apr-15	27-Apr-15
Prototype	70	190		27-Apr-15	15-May-15
Prototype for Lecturers' Application	35	95		27-Apr-15	8-May-15
Initial Proposal	5	5	Graphic Designer		
Wireframes	5	5	Graphic Designer		
API Gathering	5	5	Programmer		
Build Hi-Fi Prototype	20	80	Programmer x 4		
Prototype for Students' Application	35	95		4-May-15	15-May-15
Sketches	5	5	Graphic Designer		
Wireframes	5	5	Graphic Designer		
API Gathering	5	5	Programmer		
Build Hi-Fi Prototype	20	80	Programmer x 4		
Prototype Testing	30	90		11-May-15	25-May-15
Test cases	10	30	Programmer x 2 Graphic Designer	11-May-15	19-May-15
For Lecturers	5	15			
For Students	5	15			
Functional testing	10	30	Programmer x 2 Graphic Designer	14-May-15	20-May-15
For Lecturers	5	15			
For Students	5	15			
User testing	10	30	Programmer x 2 Graphic Designer	20-May-15	25-May-15
For Lecturers	5	15			
For Students	5	15			
Improvement of Prototype	10	20	Programmer x 2	25-May-15	1-Jun-15
Documentation	5	5	Project Manager	25-May-15	4-Jun-15

Client Meeting	3	18	All members	5-Jun-15	5-Jun-15
Phase 3: Design	**19**	**54**		**3-Aug-15**	**12-Aug-15**
Group Meetings	2	12	All members	3-Aug-15	12-Aug-15
UI Design	10	10	Graphic Designer	3-Aug-15	12-Aug-15
UI Design for Lecturers' Application	5	5			
UI Design for Students' Application	5	5			
Database Design	5	20	Programmer x 4	3-Aug-15	11-Aug-15
Client Meeting	2	12	All members	12-Aug-15	12-Aug-15
Phase 4: Development	**142**	**417**		**13-Aug-15**	**18-Sep-15**
Group Meetings	10	60	All members	13-Aug-15	18-Sep-15
Architecture	10	10	Programmer x 2	13-Aug-15	21-Aug-15
Architecture for Lecturers' Application	5	5			
Architecture for Students' Application	5	5			
UI Development	45	85		13-Aug-15	21-Aug-15
Visual Representation	5	5	Graphic Designer		
Model Implementation	20	40	Programmer x 2		
Model Integration	20	40	Programmer x 2		
Firebase & React.js Research	10	20	Programmer x 2	17-Aug-15	21-Aug-15
Database Implementation	10	20	Programmer x 2	17-Aug-15	21-Aug-15
Implementation of General Functions	50	190		24-Aug-15	11-Sep-15
For Lecturers	50	100		24-Aug-15	11-Sep-15
User Authentication	10	20	Programmer x 2		
User Settings	10	20	Programmer x 2		
Composition/Saving Questions	5	10	Programmer x 2		
Active Question Display Page	10	20	Programmer x 2		
Student Response Analyzer	10	20	Programmer x 2		
Saving of Students'	5	10	Programmer x 2		

Answers					
For Students	45	90		24-Aug-15	11-Sep-15
Code Verification	10	20	Programmer x 2		
Question Retrieval	10	20	Programmer x 2		
Generation of Canvas and Drawing Tools	15	30	Programmer x 2		
Uploading of Drawings to Database	10	20	Programmer x 2		
Functional Integration	5	20	Programmer x 4	14-Sep-15	17-Sep-15
Client Meeting	2	12	All members	18-Sep-15	18-Sep-15
Phase 5: Testing and Debugging	**67**	**172**		**21-Sep-15**	**16-Oct-15**
Group Meetings	5	30	All members	21-Sep-15	16-Oct-15
Test cases	10	20		21-Sep-15	25-Sep-15
For Lecturers	5	10	Programmer x 2		
For Students	5	10	Programmer x 2		
Functional testing	10	20		21-Sep-15	25-Sep-15
For Lecturers	5	10	Programmer x 2		
For Students	5	10	Programmer x 2		
User testing	10	20		25-Sep-15	12-Oct-15
For Lecturers	5	10	Programmer x 2		
For Students	5	10	Programmer x 2		
Result Analysis	5	5	Programmer x 1	13-Oct-15	13-Oct-15
Debugging	15	45	Programmer x 3	25-Sep-15	15-Oct-15
Implementation of Solutions	10	20		12-Oct-15	15-Oct-15
For Lecturers	5	10	Programmer x 2		
For Students	5	10	Programmer x 2		
Client Meeting	2	12	All members	16-Oct-15	16-Oct-15
Phase 6: Deployment	**41**	**131**		**19-Oct-15**	**30-Oct-15**
Group Meetings	5	30	All Members	19-Oct-15	30-Oct-15
Deploying the Solution	10	40	Programmer x 4	19-Oct-15	21-Oct-15
Stabilizing the Deployment	5	20	Programmer x 4	21-Oct-15	22-Oct-15

Transferring Ownership to Operations	5	5	Project Manager	23-Oct-15	23-Oct-15
Documentation	12	12	Project Manager	26-Oct-15	29-Oct-15
User Guide	3	3			
FAQs	3	3			
Final Evaluation	3	3			
Grand Summary	3	3			
Client Meeting	4	24	All Members	30-Oct-15	30-Oct-15

Refer to the appendix for Gantt charts that provide a visual timeline of the work breakdown and its sequencing.

Quote: Cost Estimate

Here are the Costs Estimate for the salaries of all the team members. The team consists of six members. A Project Manager, a Graphic Designer and four Programmers. By doing some research, we came up with a hourly rate payout for each of the three roles. The links below show where we got these figures from.

Project Manager - $55/hr
http://www.payscale.com/research/AU/Job=Project_Manager,_(Unspecified_Type_%2F_General)/Salary

Graphic Designer - $40/hr
http://www.payscale.com/research/AU/Job=Graphic_Designer/Salary

Software Programmer - $45/hr
http://www.payscale.com/research/AU/Job=Software_Engineer_%2F_Developer_%2F_Programmer/Salary

Using the data from *Work Breakdown* and the figures above, this is the breakdown of the Costs Estimate for the whole duration of this project.

Core Project Costs

Phase	Duration (hrs)	Man-hours	Resource	Work Hours	Rate (p/h)	Cost	Phase Cost
Project Initialization	16	66	Project Manager	16	$55.00	$880.00	**$3,080.00**
			Graphic Designer	10	$40.00	$400.00	
			Programmer (4)	40	$45.00	$1,800.00	
Planning	184	524	Project Manager	54	$55.00	$2,970.00	**$23,700.00**
			Graphic Designer	84	$40.00	$3,360.00	
			Programmer (4)	386	$45.00	$17,370.00	
Design	19	54	Project Manager	4	$55.00	$220.00	**$2,400.00**
			Graphic Designer	14	$40.00	$560.00	
			Programmer (4)	36	$45.00	$1,620.00	
Development	97	417	Project Manager	12	$55.00	$660.00	**$18,640.00**
			Graphic Designer	13	$40.00	$520.00	
			Programmer (4)	388	$45.00	$17,460.00	
Testing and	67	172	Project Manager	7	$55.00	$385.00	**$7,775.00**

Debugging			Graphic Designer	7	$40.00	$280.00	
			Programmer (4)	158	$45.00	$7,110.00	
Deployment	41	131	Project Manager	26	$55.00	$1,430.00	**$6,110.00**
			Graphic Designer	9	$40.00	$360.00	
			Programmer (4)	96	$45.00	$4,320.00	
						TOTAL COST ($)	**$61,705.00**

Supplementary Costs

Real-time Datastore

As outlined previously, the project will use a real-time datastore product called Firebase. Firebase offers a multi-tiered pricing structure (firebase.com/pricing). The plan that best meets the demands for the application is the **Candle** plan. This plan is $49 per month and offers the following quotas:

- 200 Connections
- 20 GB Data Transfer
- 3 GB Data Storage
- 10 GB Hosting Storage
- 1 TB Hosting Transfer

Image Storage

The user-submitted response images will be stored using Amazon Simple Storage Server (S3). Amazon S3 provides cost-effective and highly-scalable object storage. The storage rates for the local Sydney data centre is $0.0330 per GB per month (aws.amazon.com/s3/pricing/). The images generated from the application are typically around 50kb in size. This means that roughly 20,000 user responses will cost $0.033 per month. We will assume 1,000,000 user responses as a very generous upper-bound. This gives us a monthly storage cost of $1.65. There is also a charge for GET/PUT/POST requests. Assuming 100,000 of these requests per month, this will cost $0.044 per month.

Domain Name

The domain name for the application (**uqdraw.co**) costs $12 per annum with the domain registrar being GoDaddy (godaddy.com/tlds/co-domain).

Web Server

The application's architecture has no inherent requirement for a traditional web server setup. The only reason why a web server enters the picture is due to limitations of the UQ Single Sign On service. Other related limitations of this service also mean that the web server must be running on UQ infrastructure behind their firewall. As a result, there will be no appreciable cost input for this

server as it will be running on shared infrastructure where the capex has already been outlayed. The server will also consume negligible resources as it only needs to serve a single page to lecturer's when they are logging in. The application itself can easily be served from S3 as there is no server-side logic.

Ongoing Maintenance Costs

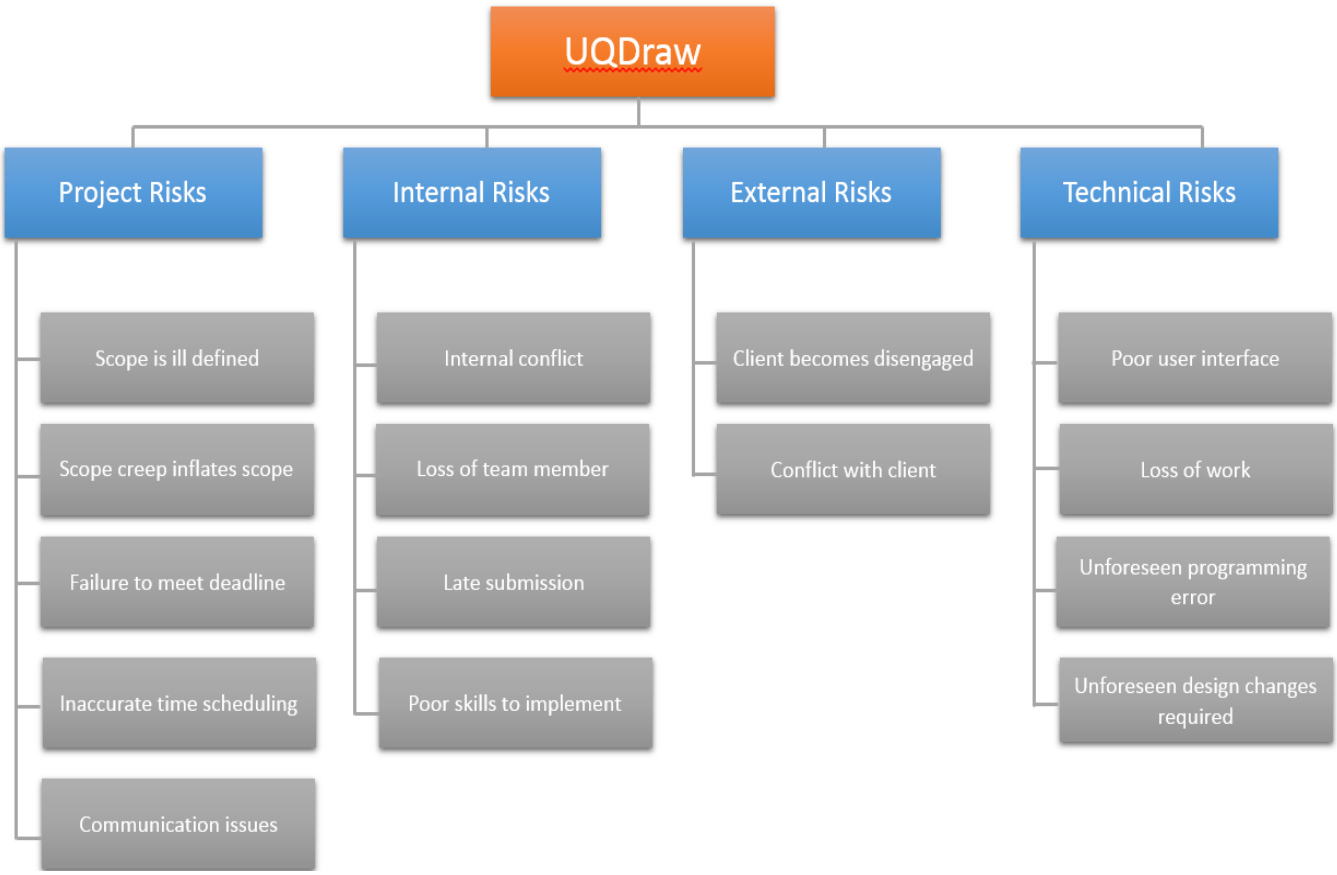
Ongoing maintenance costs are expected to be very low as a result of the chosen architecture. Sharding of data stores to ensure redundancy, patching the application web-server with new security updates, scaling-out of server resources as bottlenecks are hit, and many more common maintenance tasks are eliminated, more or less. There will be maintenance costs for servers on the UQ cloud, but these will be performed on the private cloud itself and thus are not an additional overhead imposed by the application. The main source of ongoing maintenance costs will be software bugs that are related to the plethora of operating systems and web browsers that will be using the application. There will inevitably be issues that crop up (especially as new operating system version are released in coming years) that will need to be fixed. These will be minor patches to the existing application however, so it is expected that 8 hours per month of developer time should be sufficient to handle these issues as they arise. This will roughly equate to $360 a month in developer maintenance fees at a going rate of $45/hour. Notably, this degree of maintenance is not an absolute requirement as many of the bugs will be related to non-mainstream browsers and operating systems. Many application you commonly use (eg. all banks, all existing UQ services) are littered with these types of bugs which are never fixed.

Summary of Supplementary Costs

Service	Cost Per Month	Cost Per Annum
Firebase (Real-time Datastore)	$49.00	$588.00
Amazon Simple Storage Server (Image Storage)	$1.70	$20.40
GoDaddy (Domain Name)	$1.00	$12.00
Ongoing Software Bug Fixes	$360.00	$4320.00
	TOTAL	**$4940.40**

Risk Identification & Mitigation

The main risks of this project have been identified and categorised (shown below).



All identified risks have been broken down into four parts:
Project risks: the risks happen in the project
Internal risks: the risks may happen inside the team
External risks: the risks happen between the client and the team
Technical risks: the risks happen during the development
Besides, the risk assessment and response methodology outlining the likelihood, description, consequences and mitigation strategies provided for avoidance and contingency planning has been outlined on the following table.

Likelihood	Description	Consequences	Mitigation Strategy
Part 1: Project Risks			
Risk: Scope is ill-defined			

Low	The risk of an error or omission in scope definition.	This could lead to a project that the client is not satisfied with.	Analyse project in team meeting before any work is done.
Risk: Scope creep inflates scope			
Low	Uncontrolled changes and continuous growth of scope. Client may increase scope of project during developing.	This may lead to spend extra time on new features.	Accurately dene project scope, and develop a clear work breakdown structure.
Risk: Failure to meet deadline			
Moderate	It may happen if the team does not set clear deadlines, team does not communicate often and other uncontrolled problems.	Significant delays in project.	Set internal deadlines, and review with team members regularly.
Risk: Inaccurate time scheduling			
Moderate	An unclear work breakdown or increasing the scope of the project.	Delays, failure to meet deadlines, and increased workload.	Regularly review current work, update and improve schedule constantly.
Risk: Communication issues			
Low	Often absent from the meeting or lack of communication with other team members.	There are significant delays in the project, and poor results.	Discuss goals in team meetings often. And spend extra time discussing problems with each team member.
Part 2: Internal Risks			
Risk: Internal conflict			
Moderate	An unequal distribution of workload or there are different views on the same issue.	Deadlines not met and poor quality of work.	Regular team meetings where team members have the opportunity to solve problems.
Risk: Loss of team member			

Very Low	Members may leave the team if they think the team is uncommunicative or does not have potential. Or some uncontrolled reasons.	the rest of members would need to carry more tasks in the project.	Back up works with team members. In the event a team member leaves, redistribute tasks and contact course coordinator as soon as possible.
	Risk: Late submission of personal task		
Low	It may happen if the team does not set clear deadlines. Or other team members are busy with other things before deadline.	Deadlines not met and poor quality of work.	The leader should supervise each member, ensure tasks are distributed appropriately and remind them to upload their work by the deadline.
	Risk: Poor skills to implement task		
Moderate	Most of members may have similar skills, so it may lead to poor skill to finish specific tasks.	This problem may result in an unequal distribution of workload to those with very specific skillsets, and might result in poor project implementation.	The availability of teaching staff, client assistance and various online materials should minimize the risk.
Part 3: External Risks			
	Risk: Client becomes disengaged		
Very Low	Lack of communication with client.	Lack of feedback from client on design choice.	Talk to client often, show work completed through blog posts to keep client interested.
	Risk: Conflict with client		
Low	Client is not satisfied with results or client often increases scope of project.	Deadlines not met and poor quality of work.	Communicate with client often and get feedback from client regularly. If it is happened, the team need to confirm the requirements from the client carefully again and solve the problem immediately .

Part 4: Technical Risks			
Risk: Poor user interface			
Moderate	Users are not satisfied with design, or application is required a lot of learning time.	This will make the users abandon the application and try to look for a better one.	Heavy prototyping will be done before development work on the user interface is started. The prototype will be demonstrated to the client and tested on various users.
Risk: Loss of work			
Very Low	Delete\overwrite works by mistake.	Unnecessary time spent on recovering works, or re-do these works again. This may lead to deadlines not met.	Use version control systems (such as git), share works with team. And back up works regularly with team members.
Risk: Unforeseen programming error			
Very High	There are some unknown or unexpected errors in developing application.	Deadlines not met, time wasted.	Test often, rely on shared experience of entire team. Or consult with course coordinator on how to solve.
Risk: Unforeseen design changes required			
High	Sometimes, the client will request changes to the project during development. The team will be asked to fulfill these requests and as a result, may need to redo certain aspects of the application.	Additional work in redesigning the solution. And this may lead to sacrifice the quality of the final product.	The team will be in constant communication with the client in order to resolve the issue. Besides, the team needs to present their application and the functions they made to the client every week to make sure the client is satisfied with the product.

Appendix

Figure 1 - Initial mockup of the Lecturer's welcome screen.

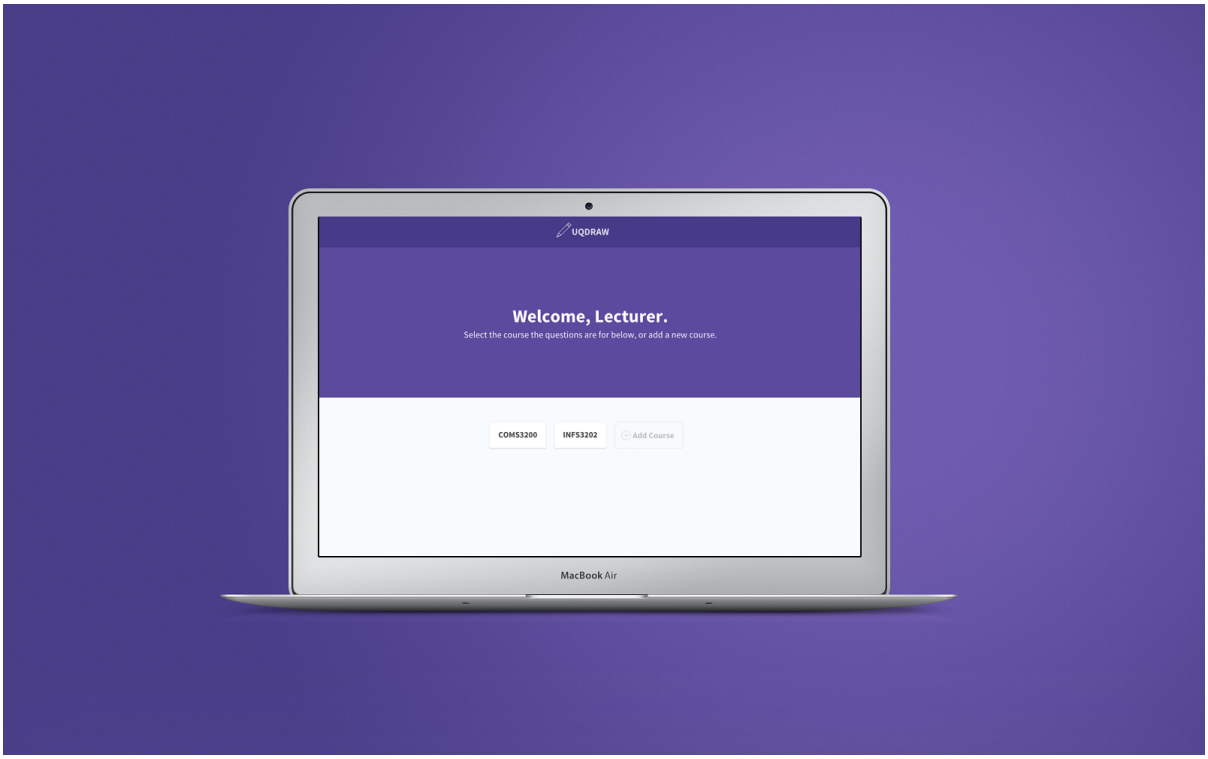


Figure 2 - Initial mockup of the Lecturer's question manager view.

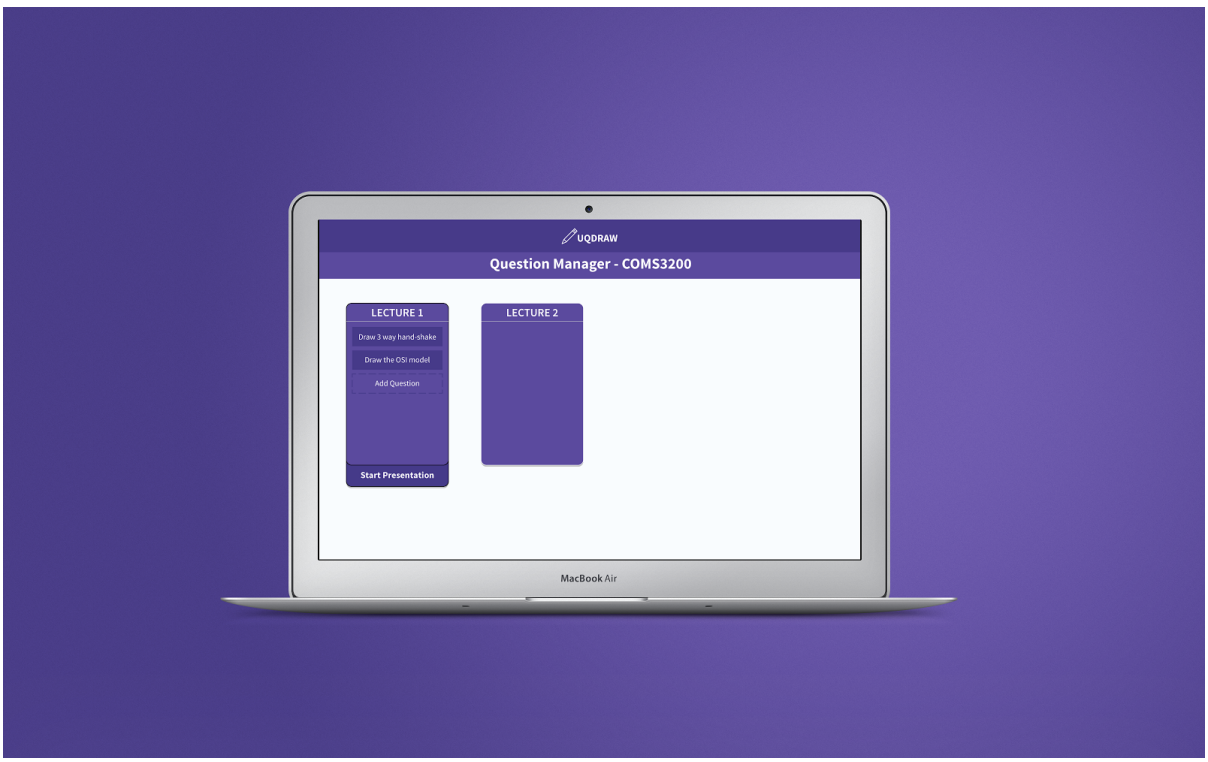


Figure 3 - Initial mockups of the Student's landing page and response drawing view.

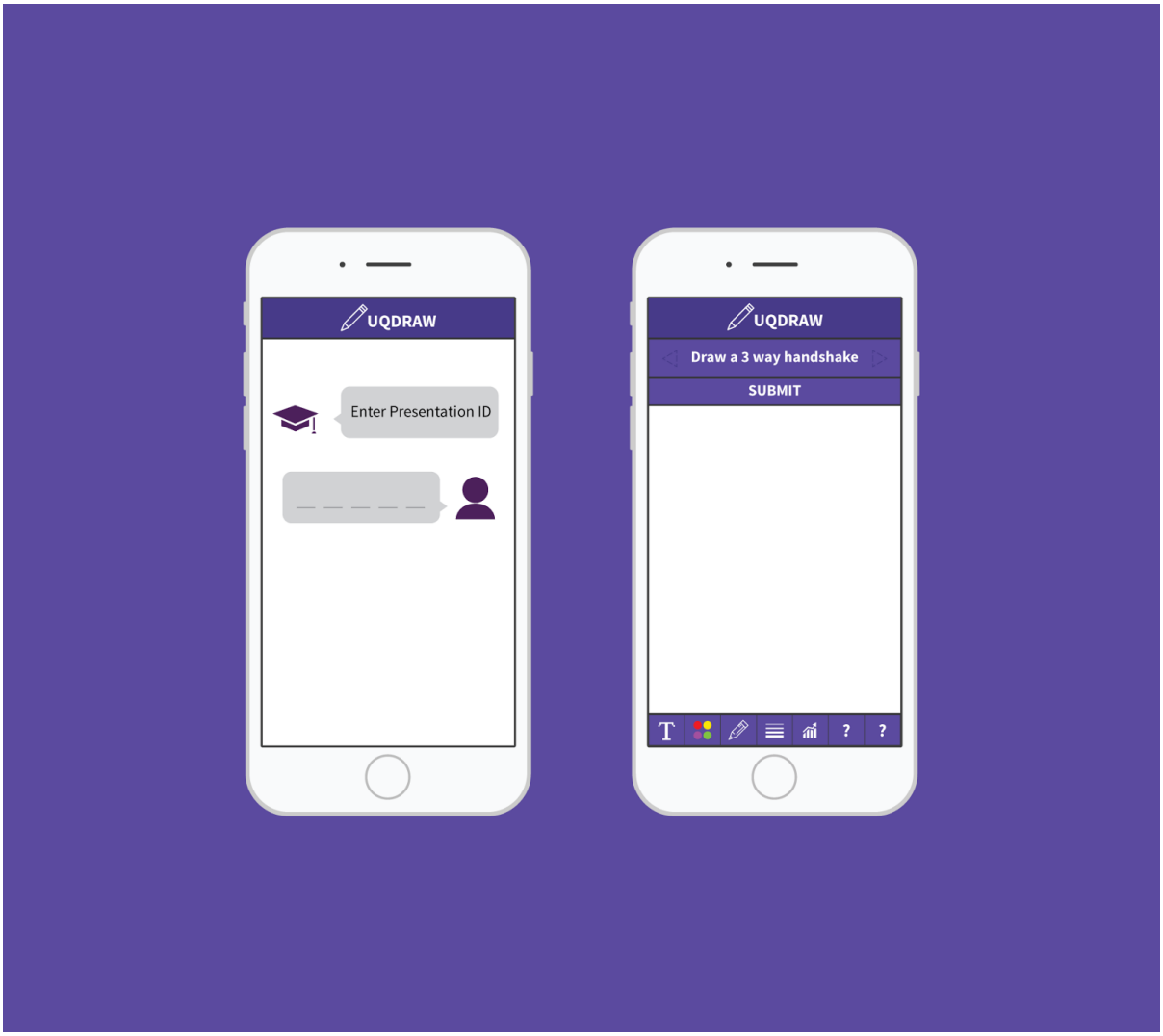
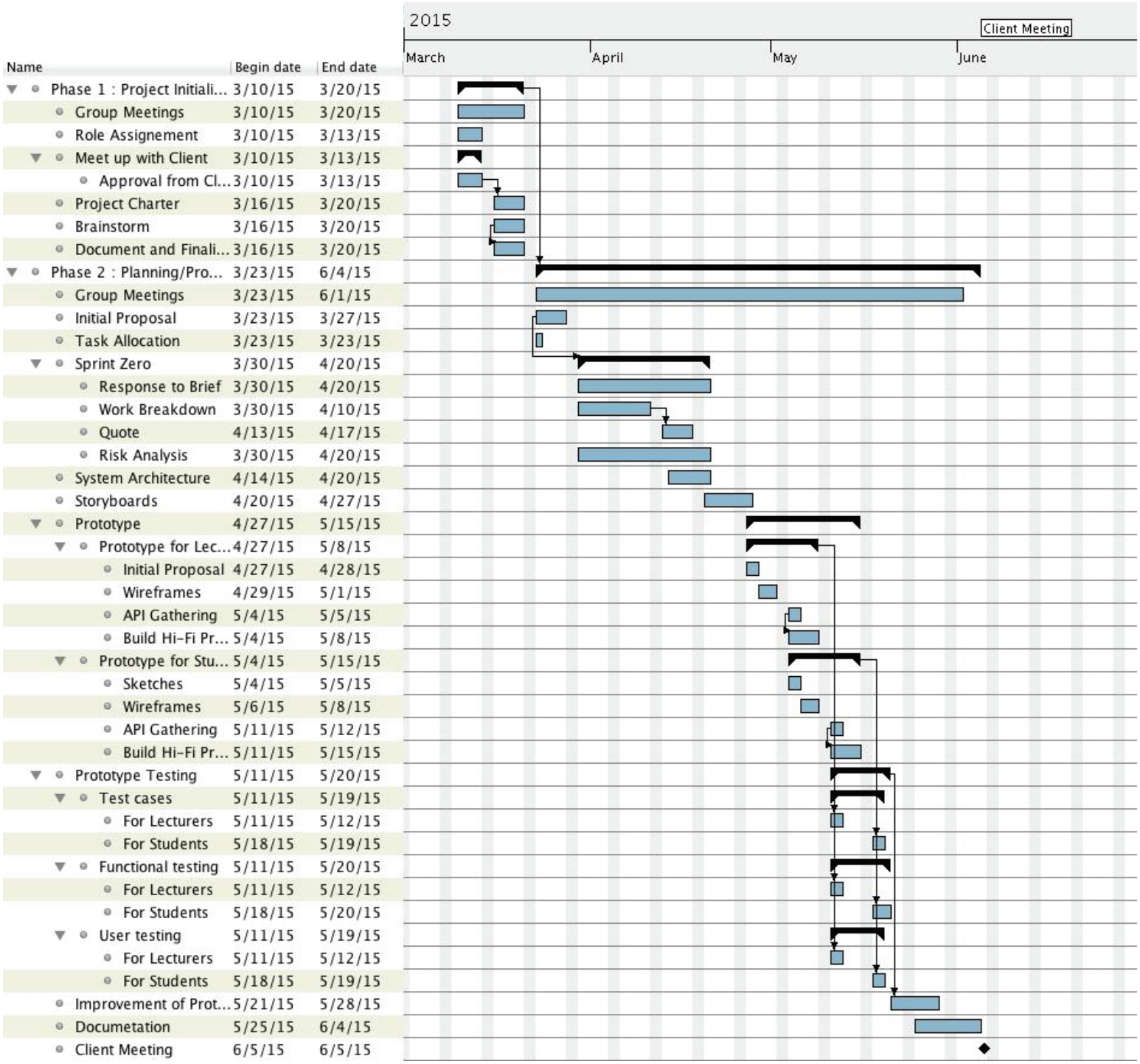
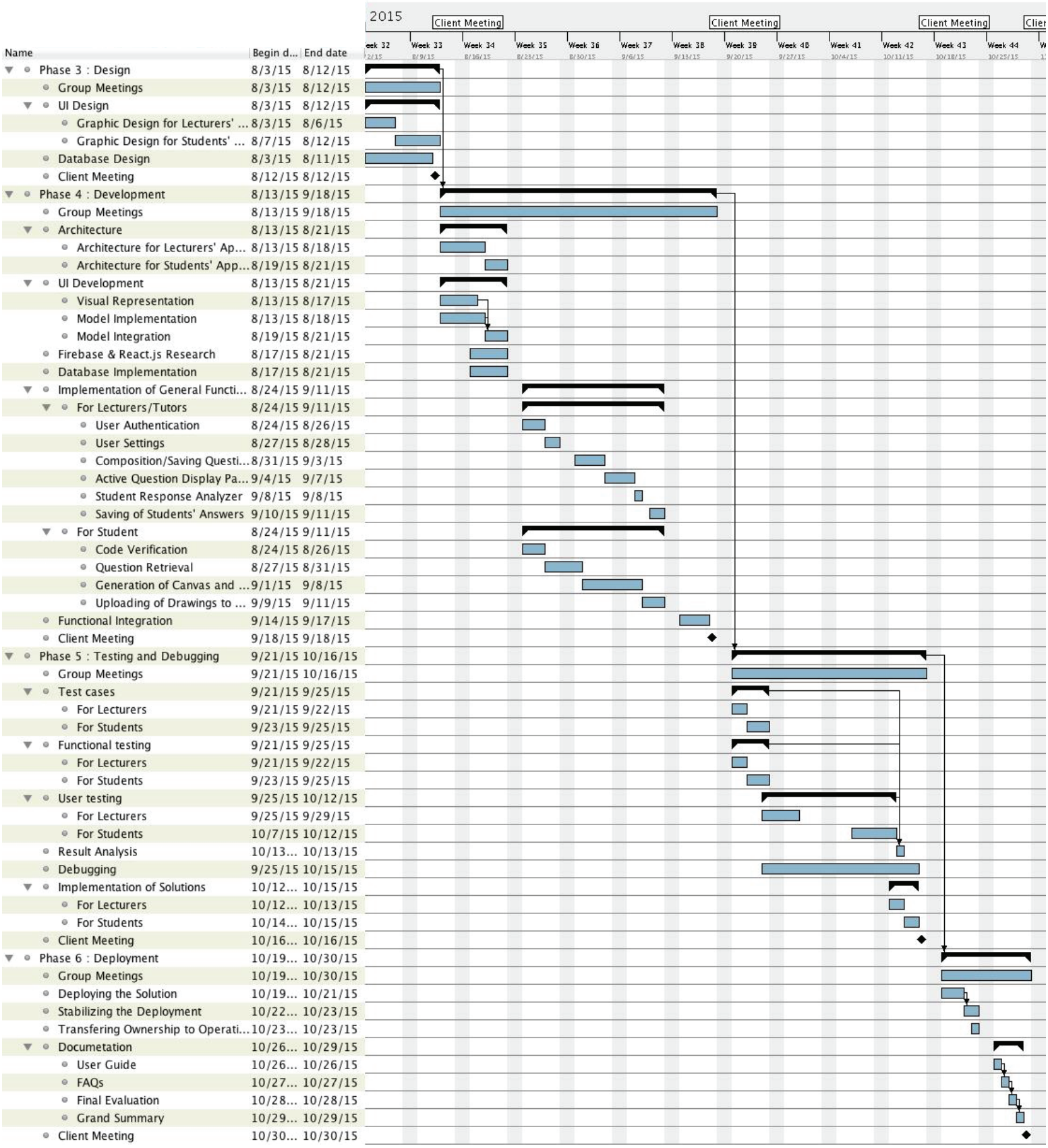


Figure 4. Gantt chart for period 1 March 2015 - 30 June 2015



Name	Begin date	End date
Phase 1 : Project Initiali...	3/10/15	3/20/15
Group Meetings	3/10/15	3/20/15
Role Assignement	3/10/15	3/13/15
Meet up with Client	3/10/15	3/13/15
Approval from Cl...	3/10/15	3/13/15
Project Charter	3/16/15	3/20/15
Brainstorm	3/16/15	3/20/15
Document and Finali...	3/16/15	3/20/15
Phase 2 : Planning/Pro...	3/23/15	6/4/15
Group Meetings	3/23/15	6/1/15
Initial Proposal	3/23/15	3/27/15
Task Allocation	3/23/15	3/23/15
Sprint Zero	3/30/15	4/20/15
Response to Brief	3/30/15	4/20/15
Work Breakdown	3/30/15	4/10/15
Quote	4/13/15	4/17/15
Risk Analysis	3/30/15	4/20/15
System Architecture	4/14/15	4/20/15
Storyboards	4/20/15	4/27/15
Prototype	4/27/15	5/15/15
Prototype for Lec...	4/27/15	5/8/15
Initial Proposal	4/27/15	4/28/15
Wireframes	4/29/15	5/1/15
API Gathering	5/4/15	5/5/15
Build Hi-Fi Pr...	5/4/15	5/8/15
Prototype for Stu...	5/4/15	5/15/15
Sketches	5/4/15	5/5/15
Wireframes	5/6/15	5/8/15
API Gathering	5/11/15	5/12/15
Build Hi-Fi Pr...	5/11/15	5/15/15
Prototype Testing	5/11/15	5/20/15
Test cases	5/11/15	5/19/15
For Lecturers	5/11/15	5/12/15
For Students	5/18/15	5/19/15
Functional testing	5/11/15	5/20/15
For Lecturers	5/11/15	5/12/15
For Students	5/18/15	5/20/15
User testing	5/11/15	5/19/15
For Lecturers	5/11/15	5/12/15
For Students	5/18/15	5/19/15
Improvement of Prot...	5/21/15	5/28/15
Documetation	5/25/15	6/4/15
Client Meeting	6/5/15	6/5/15

Figure 5. Gantt chart for period 1 August 2015 - 30 October 2015



Name	Begin d...	End date
Phase 3 : Design	8/3/15	8/12/15
Group Meetings	8/3/15	8/12/15
UI Design	8/3/15	8/12/15
Graphic Design for Lecturers' ...	8/3/15	8/6/15
Graphic Design for Students' ...	8/7/15	8/12/15
Database Design	8/3/15	8/11/15
Client Meeting	8/12/15	8/12/15
Phase 4 : Development	8/13/15	9/18/15
Group Meetings	8/13/15	9/18/15
Architecture	8/13/15	8/21/15
Architecture for Lecturers' Ap...	8/13/15	8/18/15
Architecture for Students' App...	8/19/15	8/21/15
UI Development	8/13/15	8/21/15
Visual Representation	8/13/15	8/17/15
Model Implementation	8/13/15	8/18/15
Model Integration	8/19/15	8/21/15
Firebase & React.js Research	8/17/15	8/21/15
Database Implementation	8/17/15	8/21/15
Implementation of General Functi...	8/24/15	9/11/15
For Lecturers/Tutors	8/24/15	9/11/15
User Authentication	8/24/15	8/26/15
User Settings	8/27/15	8/28/15
Composition/Saving Questi...	8/31/15	9/3/15
Active Question Display Pa...	9/4/15	9/7/15
Student Response Analyzer	9/8/15	9/8/15
Saving of Students' Answers	9/10/15	9/11/15
For Student	8/24/15	9/11/15
Code Verification	8/24/15	8/26/15
Question Retrieval	8/27/15	8/31/15
Generation of Canvas and ...	9/1/15	9/8/15
Uploading of Drawings to ...	9/9/15	9/11/15
Functional Integration	9/14/15	9/17/15
Client Meeting	9/18/15	9/18/15
Phase 5 : Testing and Debugging	9/21/15	10/16/15
Group Meetings	9/21/15	10/16/15
Test cases	9/21/15	9/25/15
For Lecturers	9/21/15	9/22/15
For Students	9/23/15	9/25/15
Functional testing	9/21/15	9/25/15
For Lecturers	9/21/15	9/22/15
For Students	9/23/15	9/25/15
User testing	9/25/15	10/12/15
For Lecturers	9/25/15	9/29/15
For Students	10/7/15	10/12/15
Result Analysis	10/13...	10/13/15
Debugging	9/25/15	10/15/15
Implementation of Solutions	10/12...	10/15/15
For Lecturers	10/12...	10/13/15
For Students	10/14...	10/15/15
Client Meeting	10/16...	10/16/15
Phase 6 : Deployment	10/19...	10/30/15
Group Meetings	10/19...	10/30/15
Deploying the Solution	10/19...	10/21/15
Stabilizing the Deployment	10/22...	10/23/15
Transfering Ownership to Operati...	10/23...	10/23/15
Documetation	10/26...	10/29/15
User Guide	10/26...	10/26/15
FAQs	10/27...	10/27/15
Final Evaluation	10/28...	10/28/15
Grand Summary	10/29...	10/29/15
Client Meeting	10/30...	10/30/15