During development, each iteration began with writing requirements and a testing plan for that iteration. After it was complete, any changes to the requirements during implementation were noted and the testing plan was executed. The following contains the documentation written for each iteration, restructured to be more concise and consistent, changes from the original notes are:

- The testing plan tables include results, this column was not completed until the end of the iteration.
- Tests where there were no results due to requirement changes during implementation have been removed from the test tables.

1 Iteration 1 - Presenter System

1.1 Requirements

R1.1 Create new Plans.

R1.2 Create new Motes within a Plan.

R1.3 Delete Plans/Motes.

R1.4 Edit Plans/Motes.

R1.5 Output JSON for Motes.

R1.6 Authorise presenter users with establishments own authentication system.

R1.7 Ensure only presenter who created Plan can access/edit it.

R1.8 Push content to distribution server.

1.2 Requirement changes during implementation

R1.6 Decision was made not to use establishments own authentication system, opting to use own account system.

R1.8 Decision was made to push content to Redis pub/sub server, rather than directly to distribution server.

1.3 Testing

Test Ref.	Req.	Method of Testing	Expected Result	Result
T1.1	R1.1	Click 'New plan' button, complete form, click 'create'.	New Plan available in Plan list, new Plan entry in database.	Pass
T1.2	R1.2	Select plan, click 'New mote' button, complete mote form, click 'create'	New Mote available in Plan's Mote list, new Mote entry in database.	Pass
T1.3	R1.3	Click 'delete' button next to Plan/Mote	Plan/Mote removed from list, entry removed from database	Pass
T1.4	R1.4	Click 'edit' button next to Plan/Mote	Same form as used with create action should appear prepopulated with existing values. Saving should modify existing Plan/Mote, not create a new one	Failed, difficulties with creating form for specific Motes. Implementation delayed until Iteration 5.
T1.5	R1.5	Use testing to call JSON output function on Mote	Function returns valid JSON containing all attributes for Mote	Pass
T1.6	R1.6	Use login form to login with demonstration account.	Login successful	Pass
T1.7.1	R1.7	Use login form to login with demonstration account	Ensure only plans created by demonstration account are visible	Pass
T1.7.2	R1.7	Use logout button to logout of account	Ensure no plans are visible	Pass
T1.8	R1.8	Click 'Push' button next to Mote	Use Redis console to check whether content is being received	Pass

2 Iteration 2 - Distribution Server

2.1 Requirements

R2.1 Receive messages published on Redis pub/sub channel.

R2.2 Use Redis to cache content.

R2.3 Facilitate resource serving for client.

R2.4 Implement socket.io to broadcast content to listening clients.

2.2 Testing

Test Ref.	Req.	Method of Testing	Expected Result	Result
T2.1	R2.1	Push mote from presenter system	View console log for distribution server, ensure mote appears.	Pass
T2.2	R2.2	Push mote from presenter system	Use Redis console to check whether content has been stored	Pass
T2.3	R2.3	Attempt to access test client	Test client should load	Pass
T2.4	R2.4	Open test client, push mote from presenter system	Test client should show broadcasted content	Pass

3 Iteration 3 - Client System

3.1 Requirements

R3.1 Enable users to authorise themselves for access to plans.

R3.2 Use socket.io to listen to content from distribution server.

R3.3 Display content based on template fetched from server.

R3.4 Navigate pushed content history.

R3.5 Prevent unauthorised users from accessing plans.

3.2 Requirement changes during implementation

R3.1 Decision was made to not require authorisation, but a plan-specific access code. Requirement changed to: Enable users to enter plan access code to subscribe to plan.

R3.4 Requirement delayed until Iteration 6 as client development overran schedule.

R3.5 Due to change in R3.1, this requirement was changed to: Ensure users only receive content for the plan they are subscribed to.

3.3 Testing

Test Ref.	Req.	Method of Testing	Expected Result	Result
T3.1	R3.1	Enter access code for plan	Latest plan content appears	Pass
T2.2	R3.2	Open client, subscribe to plan, push content from presenter	Content should appear in client	Pass
T2.3	R3.3	Open client, subscribe to plan, push content from presenter, try with different types of content	Content types should show with their own defined template	Pass
T2.5.1	R3.5	Open client, do not subscribe to plan, push content from presenter	No content should be displayed	Pass
T2.5.2	R3.5	Open client, subscribe to plan A, push content from plan B	Content should not change	Pass

4 Iteration 4 - Interaction

4.1 Requirements

R4.1 Allow client to return responses to distribution server.

R4.2 Support different structure of response for every mote type.

R4.3 Enable the presenter interface to listen for new responses.

R4.4 Support rendering of responses in the presenter interface.

4.2 Testing

Test Ref.	Req.	Method of Testing	Expected Result	Result
T4.1	R4.1	Respond to a pushed mote from the client system	Check for response in presenter interface	Pass
T4.2	R4.2	Try to respond to different types of pushed mote	Each returns different data to the presenter interface	Pass
T4.3	R4.3	Respond to a pushed mote from the client system	Check for response in presenter interface	Pass
T4.4	R4.4	Respond to a pushed mote from the client system	Response is rendered as specified in the mote type definition	Pass

5 Iteration 5 - Additional Features

5.1 Requirements

R5.1 Create additional mote types

R5.2 Create ajax-style interactions for content creation process

R5.3 Remove any device-specific code.

R5.4 Develop CSS for iPad-size display.

R5.5 Implement edit forms for Motes/Plans

5.2 Requirement changes during implementation

R5.2 Ajax-style interactions were not required, requirement changed to: Streamline content creation process.

R5.3 No device-specific code was found.

5.3 Testing

Test Ref.	Req.	Method of Testing	Expected Result	Result
T5.1.1	R5.1	Create motes of new mote types in presenter system	New motes created	Pass
T5.1.2	R5.1	Push motes of new mote type	New motes appear on client system	Pass
T5.1.3	R5.1	Respond to motes of new mote type	Responses displayed as expected on presenter interface	Pass
T5.2	R5.2	Present presenter interface to other users, instruct them to create plans and motes	Users able to easily create new content	Pass
T5.4	R5.4	Open client on iPad	Ensure all elements fit	Pass
T5.5	R5.5	Attempt to edit motes of varying types	Edit form, pre-populated with existing mote values appears, submitting form edits existing mote.	Pass

6 Iteration 6 - Polish System

6.1 Requirements

R6.1 Create additional CSS files to support other screen sizes.

R6.2 Streamline client user interface.

R6.3 Develop further content types, particularly for demonstration of other uses of the system.

6.2 Requirement changes during implementation

As features in this iteration were optional, it was considered a priority at this point to refactor code and improve the user experience. As such, the above requirements were not implemented, but the duration of the iteration was used to tidy the entire system to make it presentable.