Use Cases

The following is a list of the use cases which we have used for planning for our software development.

1. The user wishes to create the simulation and specify general parameters such as the number of devices that will run on the simulation and number of virtual networks.

2. The user wishes to enter and/or edit the number of devices the simulation will be running on.
    a. A selection mechanism needs to be available to allow the user to select the number of devices in the simulation.

3. The user wishes to enter and/or edit the number of virtual networks in the simulation environment. The user may wish to simulate the data replication across several virtual networks. We need to give the user an option to create virtual networks.

4. The user wishes to register a device for a particular virtual simulation session.
    a. A token based system of registration can be used. The virtual simulation is assigned.
    b. A fixed number of tokens corresponding to the number of devices in the simulation environment. Each device needs to be assigned to at least one virtual network set in the simulation and consumes a token provided by the virtual simulation. Once a token is consumed it cannot be used by other devices.

5. The user wishes to deploy an HTML5 application across all devices in our general simulation. Applications may make use of replicated data types.
    a. It would be useful for the user to have some visual representation of how data is being moved through the network for devices in a partition. A small feature should allow the user to simulate deploying the application on all devices.

6. The user wishes to specify which virtual networks can communicate with each other and which ones cannot. E.g. the mobile users at a hiking trail may not be able to talk to the airport mobile users network due to the lack of a network connection between them. The user needs an option list once they have created the device and virtual network set to allocate which virtual networks can communicate with the other.

7. The user wishes to eventually have all virtual networks be able to communicate with the others in order to model the purpose of the simulation - eventual consistency.
    a. Ideally a mechanism would allow the partitions to be capable of communicating with each other by sending replicated data types. In this case the user should be able to visually see consistency across all devices running the application in our system as data becomes available to all devices on the network. All the applications should have consistent data types and states.

8. The user wishes to direct data flow on the HTML5 application across all available virtual networks. The user should be able to test the replication of data types across the application

9. The user wishes to move a device from one virtual network to another, delete a device from a virtual network (bring it offline) or add it to a network.

10. The user creates a virtual network for a device and allows other devices to join this new virtual network. This means the modules for managing virtual networks needs to be decoupled from mobile devices. A mobile device can both manage its own self created virtual network and be managed by an admin portal for the simulation environment.

11. The virtual device user moves or removes virtual devices from its own self-created virtual network. Any virtual networks created by this device can also be destroyed.

12. The user should be able to run some test scripts for testing the simulation system(s) in units and as a whole. These scripts can be used to automatically run

the simulation environment without user input or ensure each unit works as expected.

13. The user checks error tracking logs for the system. Example of logs includes - the server taking long to respond to requests, virtual devices dropping out of virtual network range, networks going down. The logs should include the timestamps .

14. The user checks activity logs for the system. Examples of activity logs includes - device id, date/time stamp, GPS location, network membership, P2P activity.

15. The user deploys the simulation application on several browsers (Chrome, Safari, Opera, Firefox and Internet Explorer). The environment should be consistent in display and functionality across all browsers.

16. The user displays and manipulates the network topology. This includes joining/connecting networks, separating networks, deleting networks, adding networks.

17. The user displays and manipulates the device topology. This includes adding and removing devices from a network.

18. The user selects a device to see pertinent device information including, but not limited to the device type and the token used.

19. The user is able to display the network and device topology from a previous state by searching for or selecting a date/time stamp.

Breakdown of Use Case:

Use Case 16: Display and manipulate the network topology.

Justification for choosing this use case:
This use case is a new feature introduced by the client. It allows the user to see a graphical representation of the networks within a simulation. The user can see how each network is or is not connected and can manipulate these networks to suit their network connectivity needs.

Use Case: Display and manipulate the network topology.
Primary Actor: The manager of the simulation
Scope: The Simulation environment
Description: The user wishes to display the simulation in a graphical format that is interactive, allowing them to effect network change according to their needs.

Preconditions: The server running simulations is running and able to handle requests. While the simulation is running, at least one or more networks and one or more devices must be currently active to effectively display this utility.

Basic Flow:

1. The system provided a network and device topology page, displaying all active networks, devices and any relevant connections between them.
2. The user is able to interact with the current network and connection objects.
3. The user alters the available networks as needed by creating or deleting additional networks.

Alternate Flows:

3: The user does not have the rights to manipulate the topology

3.1 The network topology is available for viewing, though no changes can be made.

4: The user connects the wrong networks

4.1 The user selects the connection between the networks and deletes the connection.
4.2 The user then creates a new connection between the two or more desired networks.

Postcondition:

The network topology is displayed correctly according to the users needs.

Use Case 17: Display and manipulate the device topology.

Justification for choosing this use case:

This use case is a new feature introduced by the client. It allows the user to see a graphical representation of the devices within a simulation. The user can see where each device is located, either in a specific network or in no network at all and can manipulate these devices to suit their device connectivity needs.

Use Case: Display and manipulate the device topology
Primary Actor: The manager of the simulation
Scope: The Simulation environment
Description: The user wishes to display the simulation in a graphical format that is interactive, allowing them to change a devices location according to their needs.

Preconditions: The server running simulations is running and able to handle requests. While the simulation is running, at least one or more networks and one or more devices must be currently active to effectively display this utility.

Basic Flow:

1. The system provided a network and device topology page, displaying all active networks, devices and any relevant connections between them.
2. The user is able to interact with the current device objects.

3. The user clicks and drags the available devices their desired destination as required.

Alternate Flows:

5: The user does not have the rights to manipulate the topology

5.1 The network topology is available for viewing, though no changes can be made.

6: The user clicks and drags the current device to the wrong network

6.1 The user selects and drags the device to the desired network.

Postcondition:

The network topology is displayed correctly according to the users needs, with the proper devices shown within the desired networks.

Use Case 18: Display device information.

<u>Justification for choosing this use case:</u>

This use case is a new feature introduced by the client. It allows the user to view more detailed information on the device being selected.

Use Case: Display device information
Primary Actor: The manager of the simulation
Scope: The Simulation environment
Description: The user wishes to display the currently selected device's information, up to but not limited to it's device type and token information.

Preconditions: The server running simulations is running and able to handle requests. While the simulation is running, at least one or more networks and one or more devices must be currently active to effectively display this utility.

Basic Flow:

1. The system provided a network and device topology page, displaying all active networks, devices and any relevant connections between them.
2. The user is able to interact with the current device objects.
3. The user clicks one of the devices.
4. The desired information dealing specifically with that device is displayed on screen.
5. When the users curiosity is satisfied, the user closes the device information window, returning to the network and device topology screen.

Alternate Flows:

7: The user does not have the rights to view the details on the device in question

7.1 The network topology is available for viewing, though no device information is available.

Postcondition:
The device information is displayed correctly, and upon closing the device information window, the network and device topology screen is correctly displayed.

Use Case 19: Display network and device topography from a previous state.

Justification for choosing this use case:
This use case is a new feature introduced by the client. It allows the user to recall a previous state of the topology, showing a graphical representation of the network from the past.

Use Case: Display network and device topography from a previous state
Primary Actor: The manager of the simulation
Scope: The Simulation environment
Description: The user wishes to recall a previously saved network topology state view it on screen.

Preconditions: The server running simulations is running and able to handle requests. While the simulation is running, at least one or more networks and one or more devices must have been created and manipulated to create the required log files and the changes to show the difference between past and present states.

Basic Flow:

1. The system has logged the creation of and changes to the currently active network topology.
2. The user recalls the saved topology by selecting the <<insert button name here>> and entering the date/time information for the desired state.
3. The user selects OK.
4. The previously saved topology is displayed for the user on screen.
5. When the users curiosity is satisfied, the user closes the window, returning to the current network topology screen.

Alternate Flows:

8: The user does not have the rights to recall previous network topology states

8.1 The <<insert button name here>> for displaying the window that allows for date/time searching is not displayed.

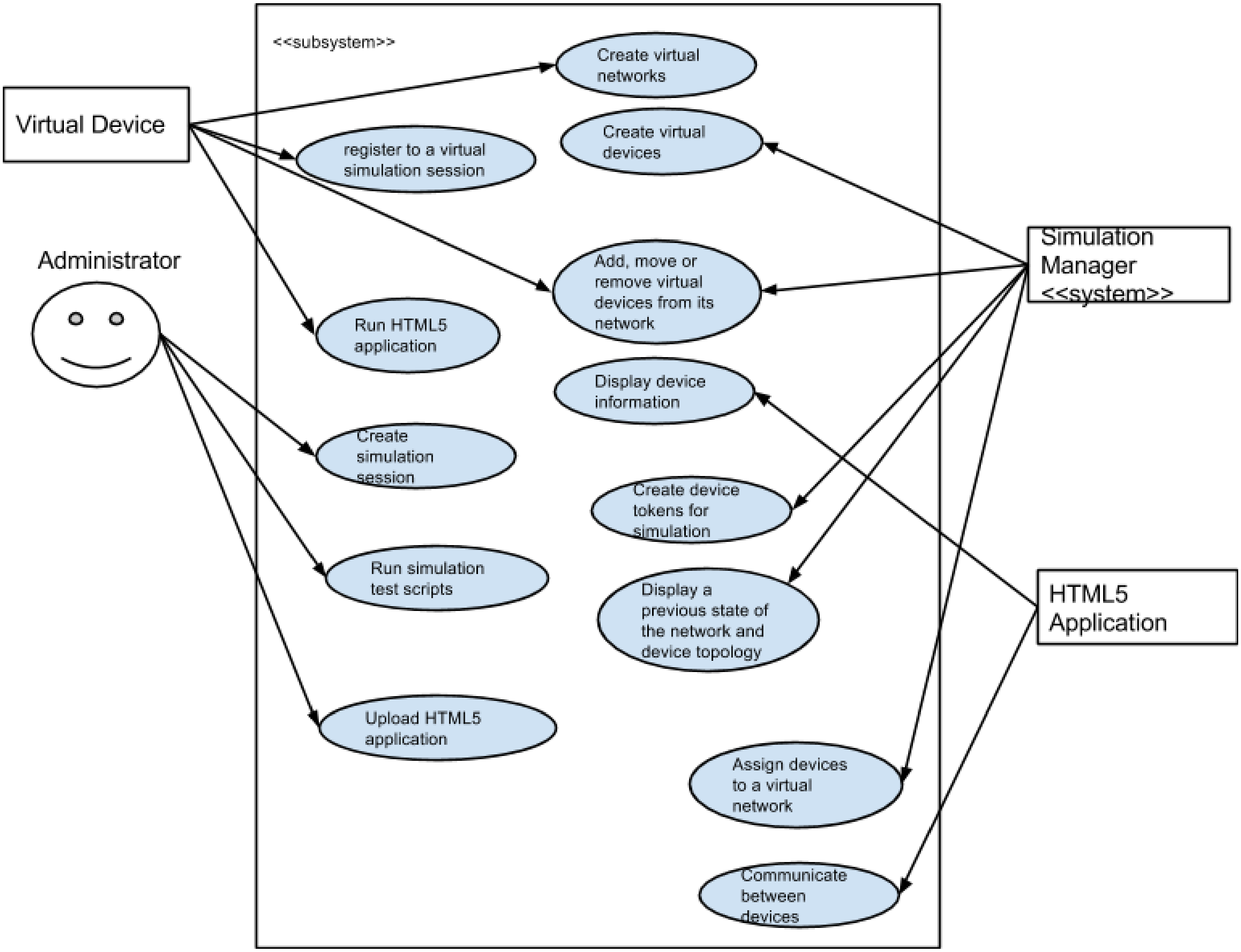
9: The user enters the wrong date/time stamp.

9.1: The user closes the window displaying the incorrect results.
9.2: The user then re-enters the desired date/time stamp and acquires the desired results.

10: The user enters the right date/time stamp, but sees no previously saved state of the topography.
10.1: The user notifies the administrator of the deficiency and repeats the process later.

Postcondition:
The device information is displayed correctly, and upon closing the device information window, the network and device topology screen is correctly displayed.

Use Case Diagram


Virtual Device
Administrator
<<subsystem>>
Create virtual networks
register to a virtual simulation session
Create virtual devices
Add, move or remove virtual devices from its network
Run HTML5 application
Display device information
Create simulation session
Create device tokens for simulation
Run simulation test scripts
Display a previous state of the network and device topology
Upload HTML5 application
Assign devices to a virtual network
Communicate between devices
Simulation Manager <<system>>
HTML5 Application

Traceability Matrix

Feature											
Use Cases	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9	Feature 10	Feature 11
Use case 1	✳		✳								
Use case 2	✳			✳							
Use case 3	✳		✳								
Use case 4	✳	✳									
Use case 5	✳				✳						
Use case 6	✳		✳								
Use case 7	✳		✳			✳					
Use case 8	✳			✳		✳					
Use case 9	✳			✳							
Use	✳		✳	✳							

case 10											
Use case 11	✳		✳	✳							
Use case 12	✳					✳	✳	✳			
Use case 13	✳									✳	
Use case 14	✳										✳
Use case 15	✳								✳		

Features List

1. Process incoming page requests and provide correct responses
2. Possesses a robust token system in order to selectively allow devices to participate in a simulation
3. Token propagation to different users via Email
4. Create and manage multiple virtual networks with:
    a. Ability to add, move, or remove any virtual devices from any networks
    b. Ability to allow or deny communication between members of different networks
5. Create a virtual mobile device in the browser which:
    a. Can register itself with a network simulation session
    b. Can connect and disconnect from pre-existing virtual networks within its simulation
    c. Can create, manage, and destroy virtual networks which it has created
    d. Can run HTML5 applications which can communicate with other virtual devices running the same application

6. Register HTML5 applications with a simulation session which any participating virtual device may run
7. Test the performance of the HTML5 applications
8. Automatic execution of test scripts
9. Track and manage multiple network simulation sessions
10. Compatible with all browsers
11. Token propagation via multiple methods, which the administrator may select from.
12. Error logging for the simulation environment available to the moderator
13. Have activity logging for the HTML5 application, tracking user login date, time, location and network activity.

Feature List Justification

We present justification behind the ordering of the features in the feature list. This includes explanations of why these features are risky and their applications to the three Q's or architecture.

1. Server features such as handling incoming page requests and providing correct responses is a very risky feature to our architecture. It is the foundation of our program, as it handles channeling data to and from users, therefore it is essential to our system. Only a single member of our team is familiar with node-js which we will be using to implement the server, therefore we are unsure of how to implement it. Finally, we are not entirely sure what the server should do and how to handle it. Therefore this satisfies the three Q's of architecture, and this feature is very risky and should be handled early by architecture.
2. Having a robust token system is critical to our system, it allows for devices to access a simulation, and without it none of the other required features for the system would be possible to implement. Therefore this is an essential feature to our system. This is a challenging feature, as actual devices as well as simulated devices must be able to have tokens, the tokens must be unique, and can only be used once. Therefore, we do not entirely understand what this means and how this should be implemented. Thus, this feature satisfies the 3 Q's of architecture and is at high risk to our design, so we should tackle it with architecture as early as possible.
3. It is necessary for our tokens to be propagated to devices which should be added to our system. The most straight-forward, sought after, and secure way we believe to do this is with email. As token propagation is required for any device to join a network, it is essential to our system. We understand this feature, but are unsure of entirely how to implement this feature. Therefore this satisfies 2 out of 3 of the three Q's of architecture, but as it is such a central feature to our system, we place it high on our features list.

4. Create and manage multiple virtual networks. This is a very essential feature to our system, as it is the part of the core of our program and most of the other features would be nonexistent or useless without it. We believe that we do not understand this feature in its entirety. We do not know how the networks will have to behave in the future, and what sort of communication will be required. As we do not entirely understand this feature, we do not understand how to implement it. Therefore this feature satisfies the 3 Q's of architecture and is very risky. Thus, we handle it with architecture early on and place it high on the priority of the feature list.
5. Create virtual mobile devices and allow connection of real mobile devices to system. This is essential to our system, but we believe we understand how to handle the implementation and understand how it should work. This satisfies one of the three Q's of architecture, and therefore is not as risky as the ones above but should be still handled by architecture and as early as possible, as it is one of the most essential parts of our system.
6. Test the performance of given HTML5 applications. This is essential to our system as it is in the given vision statement of the project. We are unsure of what it means, as we do not know how the system should interact with HTML5 applications and what should be done. As well, we do not know how to implement this, as we do not know how these applications are to be sent or run. Therefore, this satisfies the three Q's of architecture, and this feature is very risky and should be handled early by architecture.
7. Automatic execution of test scripts. This is essential to our system as it is given in the vision statement of the project. We are not exactly sure what this means, as we do not know what scripts the customer would want run and how to execute them. Therefore we are also unsure of how to implement. This satisfies the three Q's of architecture and is therefore a very risky feature and should be handled by architecture early on.
8. Register HTML5 applications with a simulation session which any participating virtual device may run. This feature is essential to our system as it is stated in the vision document of the project and therefore must be included in the program. We are unsure of exactly how to implement this and how this should be done. Therefore this satisfies the three Q's or architecture, and is quite risky. Therefore we handle this with architecture.
9. Track and manage multiple network simulations is essential to our system, as it is one of the required features to the system, although we do not believe that it is of utmost importance and therefore it is significantly low on the features list. We believe that we understand what this concept means, but we are unsure of how this should be handled and how to implement it, especially along the lines of how various simulations should be accessed. Therefore it satisfies two of the 3 Q's of architecture, and should be handled by architecture and design. For this feature,

our belief is that it would be simplest to have a website which allows you to select the simulation you wish to connect to and then route you to that specific simulation.

10. Token propagation via multiple methods this is not essential to our system, although it would provide added flexibility and allow certain users not willing to supply their email addresses, or without an email address to access our system. It would as well allow for administrators to handle different kinds of simulations with more or less security (for instance, a user being assigned a token when accessing the website is much less secure than email authentication). We are unsure of how to handle this, but we understand the feature. Therefore this satisfies 1 of the three Q’s of architecture, and so is low on our list of features, although we still believe it is an important feature.
11. Compatible with all browsers. This feature is not very risky. It is not “essential to our system”, we know how to handle it, and we know how to implement it. Therefore it is not pertinent to include architecture for this feature.
12. Error logging for the simulation does not satisfy any of the 3 Q’s of architecture, so it is low on the features list. This feature will allow for much simpler debugging and programming of our system and therefore would streamline the design process. Therefore, we believe that this is a necessary feature to our system.
13. Activity logging for the system would allow the administrator and possibly users to view activity of HTML devices and the simulation. This does not satisfy any of the 3 Q’s of architecture, so it is at the bottom of the features list. We believe that this feature would be very useful for understanding the effect that certain HTML5 applications and scripts have on the simulation and provide us with much more in depth information, which could be very useful for tracking the effects of HTML5 applications.