Final Report for Medical Game

Created by

Ivan Chen
Richard Li
Ben Ramsey

November 7, 2014

Version N/A

Contents

List of Figures

1 Introduction

1.1 Scope

This report is for the project of Software Engineering Group Project 2B in 2014. Our medical team includes Ben Ramsey, Xi Chen and Xianhe Li worked with the Adelaide Health Simulation and Skills Centre under the supervision of Professor Ali Babar during this year. The Client basically requires a mobile game for medical simulation education. Also, an scenario engine is need to be developed for generating different cases for this mobile game. The mobile game is a IOS system based application and it will run on IPHONE and IPAD with latest IOS version.

This document will be structured to have a section describing the functionality of the mobile game and changes have been made compared with the original requirement. Then section design would show the main design choices and options have been considered. And the section methodology describes the technologies and approaches we used to solve the requirements from Client. Afterwards, this document will summary the division of work for each team members and the overall description of the product we have produced. At the end of the document, Appendix section will contain some records of our group meetings and contributions.

1.2 Clinet

The medical team worked with Simon Patten and Halina Tam from Adelaide Health Simulation and Skills Centre. This centre is one of the most high-tech medical teaching facilities in South Australia. They offers practical and high fidelity simulation-based training, education and research opportunities to students and health professionals and equipped with a variety of simulation manikins and a fully integrated video interaction system for medical education and training.

1.3 Problem Description

The director of centre has played a mobile game called Resuscitation!, which is a technical medical simulator game for medical training purpose. The client was satisfied with the accuracy and coverage of medical content in this game. However, due to the lack of gameplay design and challenge design, the game is not good enough to attract medical people to play with. The Simulation Centre want to improve the medical students on dealing with emergency situations that will arise during their career. Thus, they need a mobile game to allow the students some time outside of the Centre to improve their skills with better gameplay and more challenging. Meanwhile, the Client also require a approach to add their own scenarios. This involves the development of a scenario engine to give someone without a programming background the ability to create scenarios with ease which can be download by the mobile application. More details of the requirement will be described in the section 2.

1.4 Solutions

1.4.1 platform and system

The platform we used is IOS system which is required by Client. The mobile game we developed is a menu based system where the players will select various actions in a particular order to save the patient in the scenario.

1.4.2 Medical Game

The first thing that happens in the game is a Handover screen where a nurse or doctor lets the players know all the basic information. Then the player need do actions to save the patient. At the end of the

scenario the player is given the opportunity to reflect on their performance. Actions selected by players could display videos to show how they work in simulation centre.

1.4.3 Scenario Engine

The game is not designed for one special scenario. A scenario engine is developed for medical teachers to add new scenarios by themselves. The scenario engine is a web-based system application that supports a platform to build new scenarios by filling medical data and related videos into a set of web forms. The system would integrate those information with package files which could be downloaded in mobile game. The section 3 would describe the design of the solution in detail.

2 Specification

2.1 Overview

This section will outline the functionalities that the medical mobile game aims to provide and the changes have been made compared with the original requirements.

There are two main goals for the project. The first goal is to develop a game to be run on IOS based system devices which will provide medical students the ability to simulate a medical emergency. The second goal of the project is to develop a scenario engine which will allow someone without a programming background to be able to create scenarios that can be used within the game.

2.2 Requirements

The requirements section describes all of the requirements that focus on how the software functions and some non functional requirements which include usability, reliability and evironmental setup of the application.

2.2.1 Functional Requirements

- Patient Dialogue

  The player must be able to engage in conversation with the patient. This is to be done to assist in the diagnosis of the patient. Using a limited number of dialogue options for this is acceptable.

- Scenario length

  The length of the scenarios need to be limited to both hold the audience's attention, and provide a simulated stress to diagnose the patient.

- Player reflection

  The player must be able to reflect on their actions. The reflection page is to consist of questions of both free text (limited to 150 words) and specific (short/multiple choice) questions. The player should also be given the ability to view and comment on their past reflections, to allow for reflection in their change in thinking.

- Visual Feedback

  Visible feedback including deterioration of the patient needs to be easy to observe. This deterioration needs to be in keeping with the current game state. e.g. if the patient has been given the correct medications, they should not continue to deteriorate.

- Player Summary

  The player must be given feedback in the form of a scenario summary page. This summary page will show the user how they performed, and if there were any issues in their performance, explain what they were and why they were issues.

- Scenarios Download and delete

  Player can download scenarios from a unique website while in the game menu. Also, player can delete any of the scenarios the player has downloaded

- Video Control

  Actions could be displayed by videos and players can skip any of the video sequences.

- Prescription Chart

  To be able to administer drugs to the patient, the player must first correctly fill out a prescribing chart which then either provides immediate feedback for a wrong drug name/irrelevant drug, or delayed feedback for a non-irrelevant drug but not completely correct. The appearance of the prescribing chart should be in accordance to the standard used in South Australian hospitals. Prescription chart needs to have the player provide dose, route, as well as the drug name.

- Monitor Sound

  During players treat the patient, the sounds of medical monitor will react to the status of the patient by playing sounds of different frequencies according to ISO 60601-1-8.

- Personal Protective Equipment

  The player must have to choose the PPE required for a particular case. This is a major requirement as choosing PPE should be an automatic response.

- Adding Scenarios

  The client needs to have some manner to generate scenarios for the game, without knowledge of programming.

2.2.2 Non Functional Requirements

- Operating System

  This game must be developed to run within the iOS mobile operating system.

- Input Controls

  This game must have all controls to be usable with a touch screen.

- Screen Space

  As the mobile device gives a limited amount of screen space, there must be some way to still show all of the needed data while still being readable to the user.

- Background Running

  When the game is interrupted by any event like Press Home and Phone call, the process running the game should be paused and run in the background of the mobile device. When the player starts the game again, the screen must display a start button and let the player continue the previous session.

- Game Process

  The game process should be executing as expected and should only crash in cases of extreme error.

- Storage Persistence

  When a user has played a scenario the game should keep all of the statistics and reflections that are corresponding to that scenario.

- Input Controls

  This game must have all controls to be usable with a touch screen.

2.3 Changes

- Platform

  Client wish to play the medcial game on multi-platform on different mobile devices. After discussion in the meeting, this requirement was changed to only IOS system based game. At first, we suggest the web based game because diffenent system can access web page so that the multi-platform functionality would be easily solved. However, considering to the application performance and network dependence, this approach was rejected. If we still want implement the multi-platform, the game should be developed on both Android and IOS system. But, the difference in programming language, development kit and code structure lead to a big challenge. Thus, finally, the requirement was changed to developing the game only on IOS based system.

- Scenario length

  The length of each scenario is changed from 15 minutes to 8 minutes. Client thought 15 minutes is so long that not enough challenage for medical students who need to be improved their medical skills under emergency situation.

- Prescription Chart

  At the very begining, the player can choose drugs by a drug list. Then it was replaced by prescription chart which need to be filled in by players. Client states the way of filling names of the drugs is more beneficial for medical students to remember them.

3 Design

3.1 Medical Application

3.1.1 Overview

The Medical Application is an iOS based game that aims to allow medical students and professionals to improve their skills in responding to emergency situations using their medical knowledge. The application is designed to allow the player to perform actions for both diagnosis and treatment of a patient. The application was designed with generic scenarios in mind, so that the scenarios are loaded into the application from files, meaning that the application code is as generic as possible with as little as possible being defined by the code.
Using a MVC structure for the code allows the separation of the different components allowing for changes to have minimal impact on the remainder of the system. We chose to rename the different components as the models of the application are really assets for the application to use. The names chosen for our three high level organisational components are:

- Assets (Model)
- Logic (Controllers)
- Presentation (View)

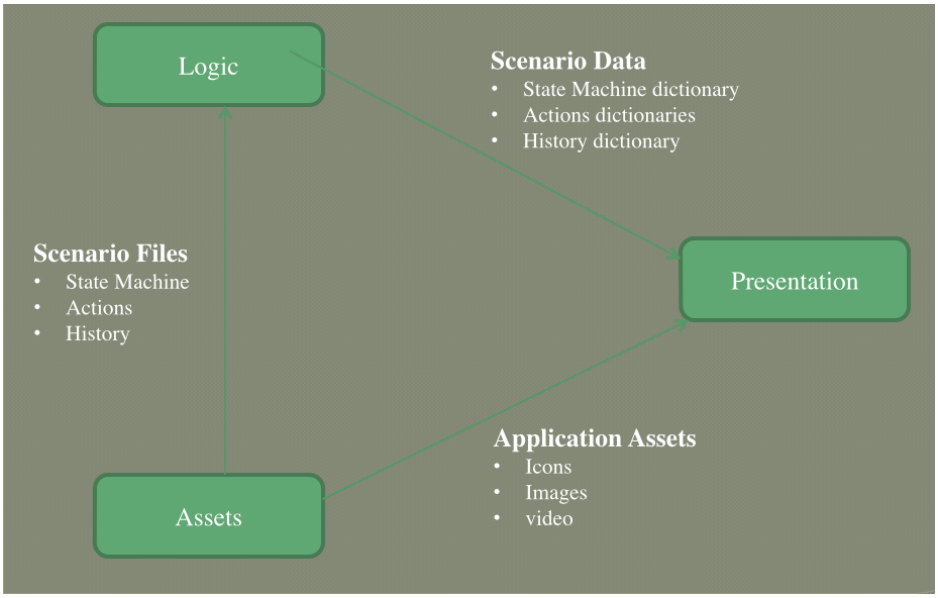

Figure 1: The basic organisational structure of the application

Assets The assets structural component contains scenario information and visual files. This component contains the data files used for the meta-data reading for the scenarios. This includes the drug list, diagnosis lists (investigation, examination, and history), procedure lists, and state machine file. As well as this it contains the video files needed for the visual aspects of the scenario and the scenario meta data which contains links to the files needed for the scenario. This component is used by both the logic and the presentation components. The logic component uses the asset component for the loading of all scenario data into the current game being played. The presentation component uses this to load videos, get icon assets, and other graphical requirements.

Logic The logic structural component contains all of the backend logic of the application which does not have to do with the graphical interface of the application. This includes such things like the state machine, required data structures for scenario information, and scenario metadata. The main task of the logic component is to provide the presentation component with all of the data required for the running of a scenario with a defined interface between the components.

Presentation The responsibility of the presentation structural component is to perform all of the necessary actions to server the graphical interface for the application to the user. The presentation component is made up of three subcomponents: view components, view controllers, and views. The view component subcomponent contains any logic for the custom UI objects created. The view controllers subcomponent contains the logic required to fetch information from the logic component and use it to both get any data required, and perform a user action request.

3.1.2 Design Decisions

Scenario Storage

Description: The scenarios will be stored using plist files which are simple to transfer into the application in the forms of dictionaries. The plist files to be used for the creation of a scenario include:

- Scenario State Definition
- Scenario Data
- Drug List
- History List
- Investigation List
- Examination List
- Procedure List

For extra information about the format of these files see Section 3.1.4

Rationale: The main reason that this choice was made is to encourage reusability, and allow the generic nature of the scenarios be as easy to implement as possible. For a more detailed rationale of this choice see Section 4.2.3

Alternative: Scenario Object Archives.

Objective-C application for iOS:

Description: Building the application using the Objective-C programming language for the iOS mobile platform. The SDK used is iOS 7, which is being built using the XCode IDE.

Rationale: As iOS devices are the most common device under the target demographic of the application the choice of target platform is fairly simple (see Section 4.2.4). In terms of the language choice, this was much more difficult as the target platform had two choices at the start of the project i.e. Objective-C, and Web based languages, but during the year Apple released their new language Swift. We chose Objective-C over the Web based languages due to performance considerations that had to be taken account of. We decided not to migrate to the swift programming language after its release as it is only runnable on iOS 8, which was released onto phones in September meaning that the majority of the target demographic may not have access to that particular version of the SDK. The migration did not happen also due to the amount of time such a task would consume.

Alternative:

- Web-based language e.g. Ruby on Rails, Python Django/Flask
- Swift

Generic State Machine

Description: This part of the code is what allows multiple scenarios to be played without constant modification of the codebase. In contrast to most games where the level (scenario in our case) is only being used by a single screen our application has many screens for each scenario. This means that the state of our scenario needs to be kept track of through screen transitions. In terms of code this means that instead of having a class per level which is a possibility for simpler games, this means that the state machine for the scenario needs to be generic. As described before the scenario is stored in a series of files, this implies that the state machine component needs to read from a file and be able to transition from state to state using the dictionary generated from the plist file. In order for a state transition to occur the view controller would pass a action dictionary to the state machine which will then be used to determine what the next state is if required.

Rationale: This method of using files significantly decreases the complexity of our code base making it easy to test, modify, and extend. Having a single component in charge of the state machine with a well defined interface also gives the ability of having a known structure to interact with the state machine for other components without it being a large, complex task.

Alternative: N/A

Scenario Metadata

Description: This is a file which allows the application to associate certain files with a particular scenario. The files that need to be associated with this can be found in the Scenario Storage design decision section. The definition of this file is that within a plist array object is a series of dictionaries which provide details about the various scenarios that are installed on the device, including their name, handover video that should be played, and the scenario files.

Rationale: This allows for easy finding of files for the scenario currently being played, decreasing the code complexity significantly. This also has the effect of promoting reuse in the different files as it allows for multiple scenarios to reference the same files if required.

Alternative: Enforced file naming convention e.g. <Scenario_Name>_<File_Type>.plist

3.1.3 Component Decomposition

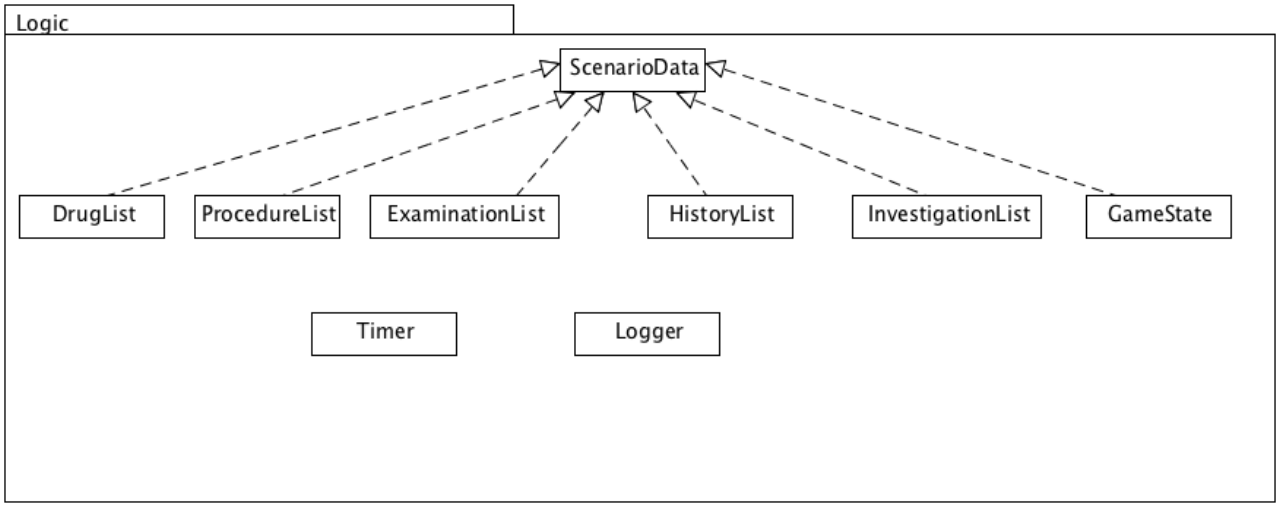

Figure 2: Components of the Logic Organisational Component

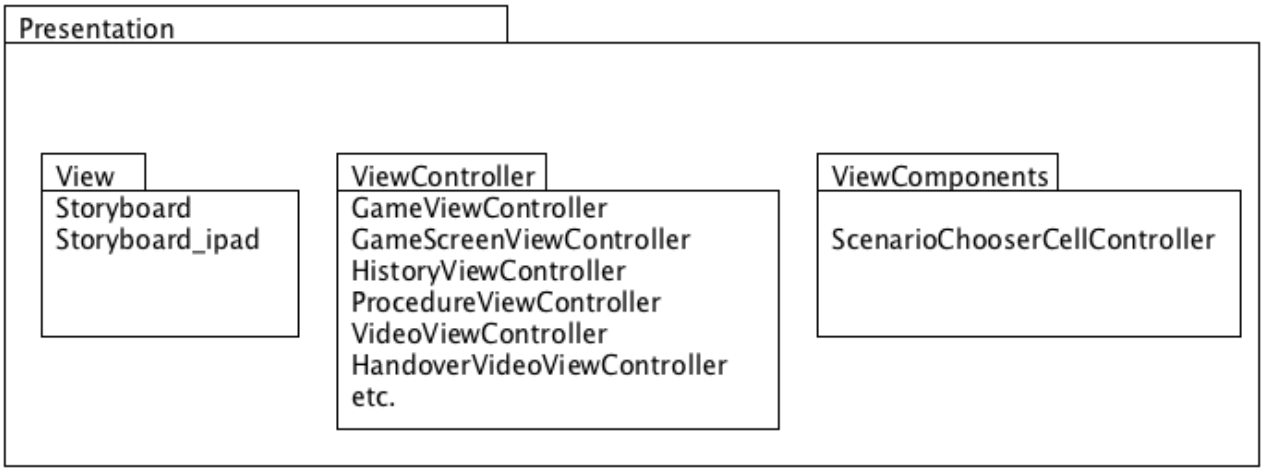

Figure 3: Components of the Presentation Organisational Component

While there are a large number of components within these two organisational components we will be focusing on the high importance ones. These are:

Game State This component manages all of the state machine details including reading in the state machine file to build the state dictionary, performing the necessary actions when passed in an action dictionary, and state transitions.

List Components These components e.g. history list, contain a dictionary data structure which is built from a file from these classes, and they also allow querying by the presentation components related to the particular component to get the list of possible actions.

Timer This component is what is used to simulate the pressure of the scenario, this is done by having a timer count down every second with actions being performed taking off time on top of this.

HandoverVideoViewController This component is in charge of the playing of the handover video which is how the scenario starts.

GameViewController This component is not a view controller associated with a particular screen, rather it is a super class which provides methods to the majority of the other view controllers, mainly the ability to start and stop videos from playing.

GameScreenViewController This component controls the parent views actions which the components such as HistoryView, ProcedureView are embedded into and controlled from. It also provides the ability to programmatically switch which view controller is being used in its container, deciding what screen is being seen to simulate a tabbed view.

3.1.4 Data Format

The data format of each of the required files needs to be decided with care and precision as it impacts greatly on how the application logic will be written and the complexity of every action with the risk that if this is not done correctly the performance will be significantly impacted in a negative way.

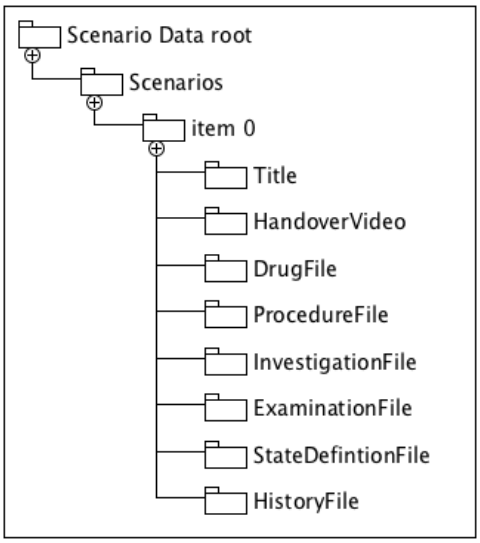

Figure 4: Data Format for Scenario List File

In Figure 4 we can see that the structure is already in the form of a dictionary which has a root with only an array as its children. This array contains the metadata for every scenario that is installed into the system. Apart from the title key, all of the other values associated with the keys for item 0 is a reference to a file contained within the application which should be associated with this particular scenario. As can be seen from Figure 2 this needs to be highly tested as most other components within the logic organisation component is dependent on this being correct, and thus the entire application needs this to be completely tested.

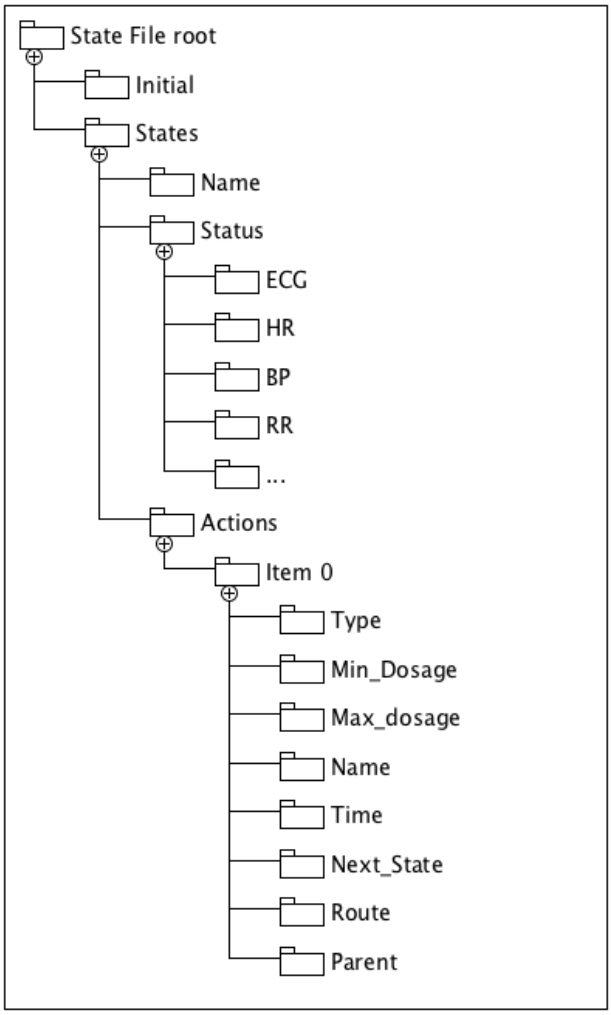

Figure 5: Data format for the State Definition File

Initial The initial state for the application, this name must reference a state defined in the states array with the value of initial being matched to the name of the state.

Name An easy to understand name for the state

Status A dictionary where the keys are medical statistics of the patient, and the values are what those statistics are in the current state this allows the ability to show the difference between states of the scenario.

Actions An array of important actions to the state machine which triggers a state transition.

Type The type of action that is being performed which is either procedure, or drug as those actions are the ones which trigger a state transition.

Min/Max Dosage For the drug type action, what is the range of values that the dosage can be which will trigger the action.

Action Name The name of the action, which is used as a search key

Time Amount of time the action takes to complete

Next State The state to transition to if all is well

Route The route of drug application to the patient.

Parent The parent action if there is one. (this is rarely used)

3.1.5 User Interface Design

The user interface design of the IOS based game focus on the player's experience and interaction. The goal of user interfaece design is to make the player's interaction as simple and efficient possible. Also, the game should be integrated with medcial element which is better for medical students to play.

storyboard for different devices As the result of the IOS system, the game should work well on both iphone and ipad which have different size and resolution of the screen. Then, we need to design two interface version(storyboard) for both iphone and ipad. Thus, any images including backgrounds, buttons and simulation monitor need to be considered the resolution. Also, the postion of any view and button is easily built by storyboard which are dragged by cursor instead of code. So the we can not use the same storyboard for iphone and ipad. Everything need to be modified two times instead of just copy and paste.

medical element Because the IOS game is developed for medical students, more medical element should be integrated into the user interface. Background image, action buttons and simulation monitor are designed with medcial visual effect and medical students could easily know the function for each action button and the current status of patient by looking the at images.

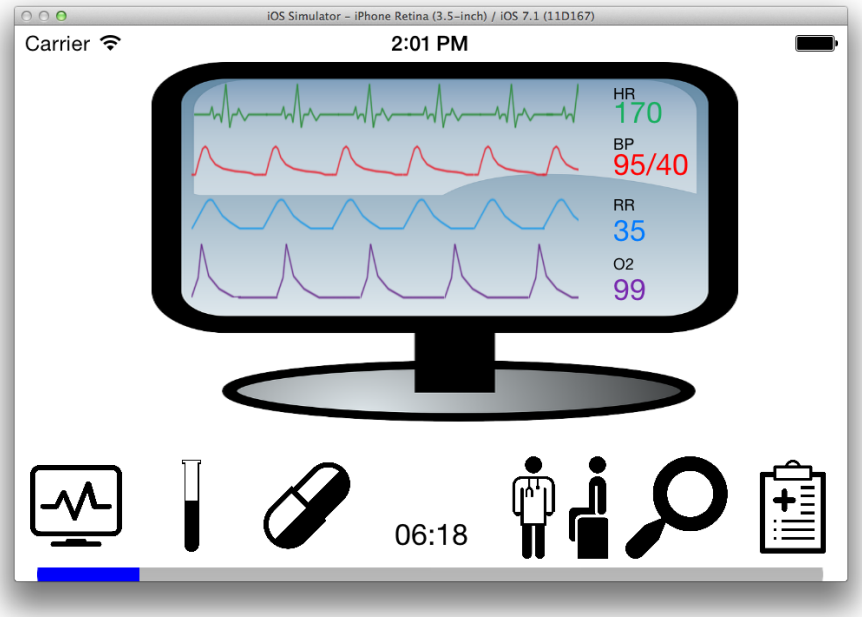

Figure 6: medical element monitor and buttons

3.2 Scenario Engine

3.2.1 Overview

Scenario Engine is a web-based server that aims to help game administrators to create new game scenario for download without any software background knowledge. The Scenario Engine is designed and implemented with a Model-View-Controller pattern using Ruby on Rails framework. It contains:

Scenario Generator Scenario Generator is a web-based service that generates game scenario by gathering game data from administrators via various forms. It contains different types of forms that require administrators to insert medical information and videos, which will be used to generate game scenarios for users to download. Also, all data inserted will be saved in the Scenario Engine, so that it can be reused in later scenario generations.

Scenario Downloader Scenario Downloader is a web page that contains all scenarios that are published by administrators. It can be accessed by users via the Game on their iOS devices. The users can view the listed scenarios and download them into the Game directory to play.

3.2.2 Main Design Decisions

Ruby on Rails

Description: The Scenario Engine is implemented using Ruby on Rails, which is a MVC framework of Ruby used for developing Web-based software. The version we used is 4.1.5, which is the latest stable version we have got from the developers.

Rational: Ruby on Rails is a rapid web development tool that allows us to model out website features quickly. It has very useful and powerful testing tool, so that we can test our codes easily. Also, Ruby on Rails has a large and helpful community that we can get help from when we have some issues.

Alternative: PHP, Python

Model-View-Controller

Description: The Scenario Engine is implemented with a MVC architectural pattern, which contains scenario controller, scenario view and scenario model.

Rational: Using MVC pattern can make Scenario Engine more structural and flexible. Also, developers can maintain codes in better ways using mvc. MVC pattern can reduce complexity of codes so that the codes can be more suitable for future implementation.

Alternative:

Gathering data using html forms

Description: The Scenario Generator contains list of various html forms used for generating scenarios. The html forms require the basic information of scenarios, such as names, medical data, time taken and video.

Rational: Using html forms to gathering data from admin can make this process easier to admin, since it requires no software knowledge to insert medical data into forms. Also, html forms provide different types of input fields that can generate scenario with more clear structure.

Alternative: Uploading related medical files, such as csv, excel, text and zip files, that contains scenario information.

Models with many-to-many relationships

Description: Models, such as scenarios, drugs, procedures investigations and examinations, are built with many-to-many relationships. Therefore, the existing data editing and deleting in the controller will be only change the relevant relationship tables instead of edit the target models.

Rational: It is required that some models can be reused in the later scenario generations. In order to achieve that, using many-to-many relationship can make models independent, which means that deleting data in scenario only means deleting the relationship between two models instead of deleting model itself.

Alternative: Model with one-to-many relationships, which means that each scenario related information, such as drugs, only belongs to one scenario and cannot be reused by other scenarios.

Auto-completing form based on the existing information

Description: A jQuery and JavaScript function that provides auto-completing and auto-filling feature is added into the scenario forms in Scenario Generator. This helps admin to reuse the existing scenario data in Scenario Engine.

Rational: Auto-completing drop-down list and auto-filing form can provide a better user experience, which can make the scenario generation with form easily to use.

Alternative: A html list that contains all existing scenario data in Scenario Engine.

3.2.3 Component Decomposition

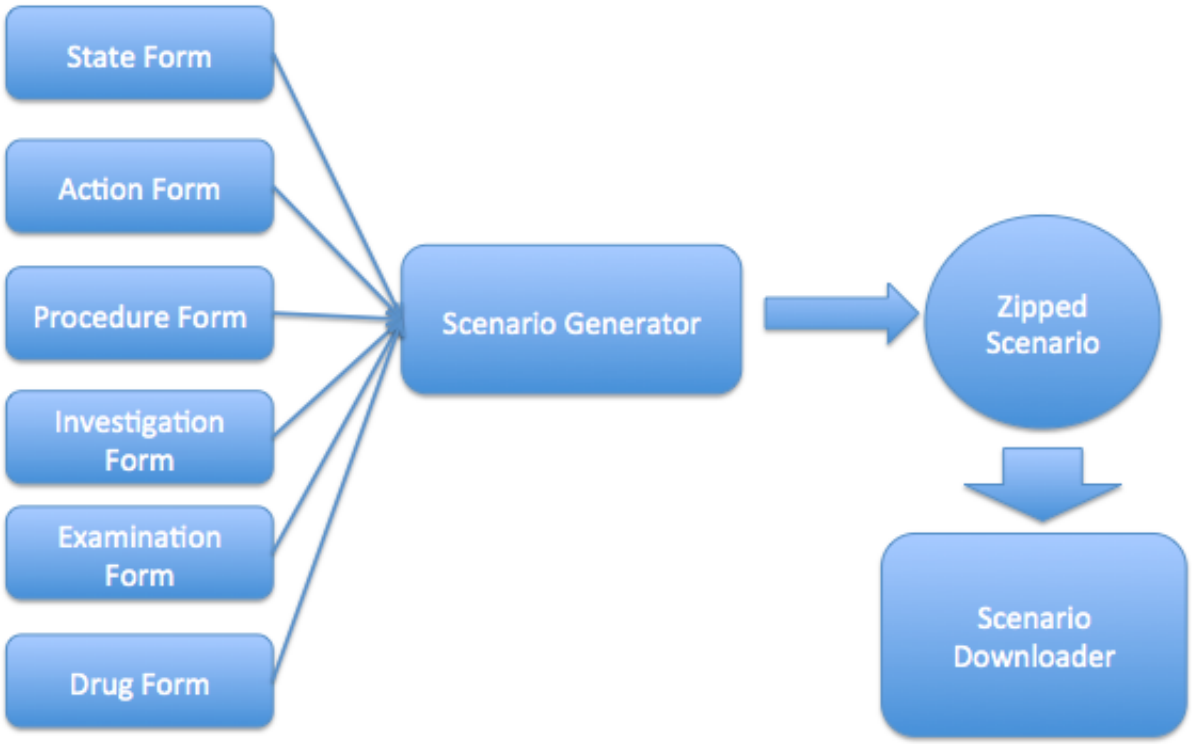

Figure 7: Component Decomposition

Main Page: This page lists all the scenario created by administrators. Admin can create a new scenario and view/delete/publish existing scenario.

Scenario Display: This page displays all the information in the selected scenario. Admin can click name to view information in the section.

General Form: This page contains list of forms that can help admin to edit the general information of scenario, such as scenario name and length.

State Form: This page contains list of forms that can help admin to edit the state information of scenario, such as state message and medical data in the state.

Action Form: This page contains list of forms that can help admin to edit the action information of states, such as action type and action name.

Procedure Form: This page contains list of forms that can help admin to edit the procedure information of scenario, such as procedure name, procedure time and the video for the procedure.

Investigation Form: This page contains list of forms that can help admin to edit the investigation information of scenario, such as investigation name, investigation time and the video for the investigation.

Examination Form: This page contains list of forms that can help admin to edit the examination information of scenario, such as examination name, examination time and the video for the examination.

Drug Form: This page contains list of forms that can help admin to edit the drug information of scenario, such as drug name, drug time, drug dosages, drug route and the video for the examination.

Download Page: This page contains list of scenario that published by admin, so that users can download scenario by visiting this page via the game on their device.

3.2.4 Architectural Design

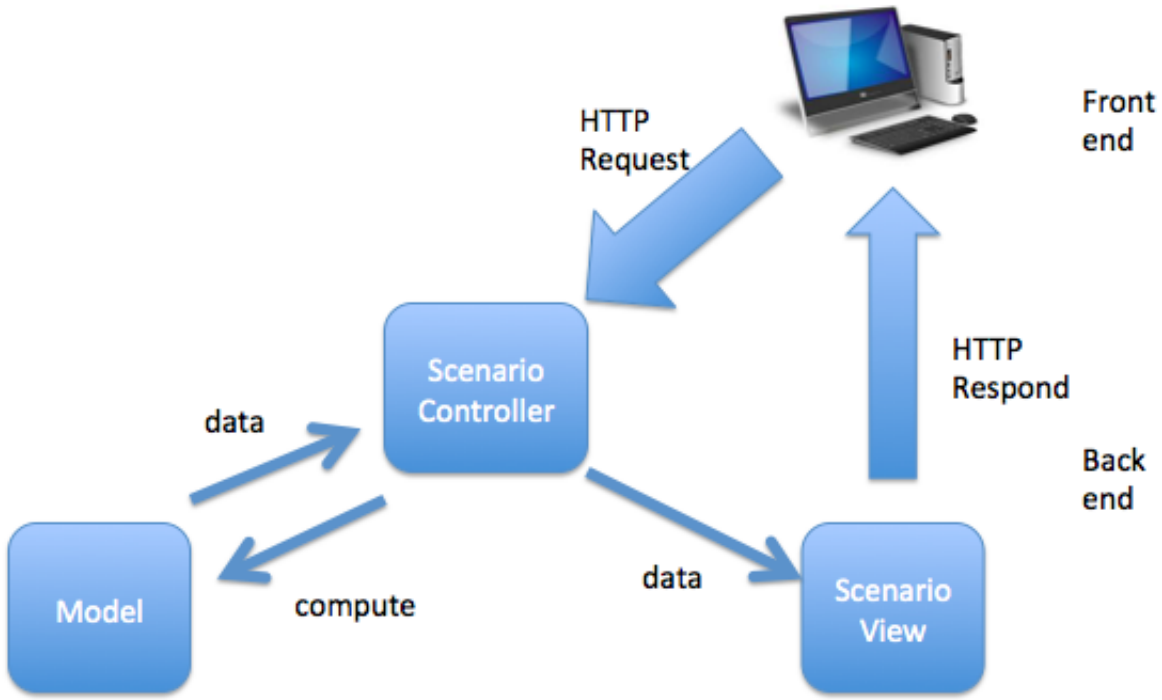

Figure 8: Model-View-Controller Structure

Scenario Controller Scenario Controller controls communications between Scenario View and Model. It can:

- gathering variables from Scenario Models
- receiving parameters from Scenario Views
- computing data received
- passing variable to Scenario Views to display
- passing data to Scenario Model to modify data in database

Scenario View Scenario View can access variable from related Scenario Controller and display them on the html page. It also can receive input from users and pass them as parameters using HTTP POST requires.

Scenario Model Scenario Model gathers scenario data from database and generate scenario model based on the relationship implemented in the model. It can receive the requires from Scenario Controller, generate data from databased by sending SQL commands and send back the generated data model as required. Scenario Model contains different kinds of models used for scenario generations, which can be extended by adding more models related to the scenario model.

3.2.5 Database Design

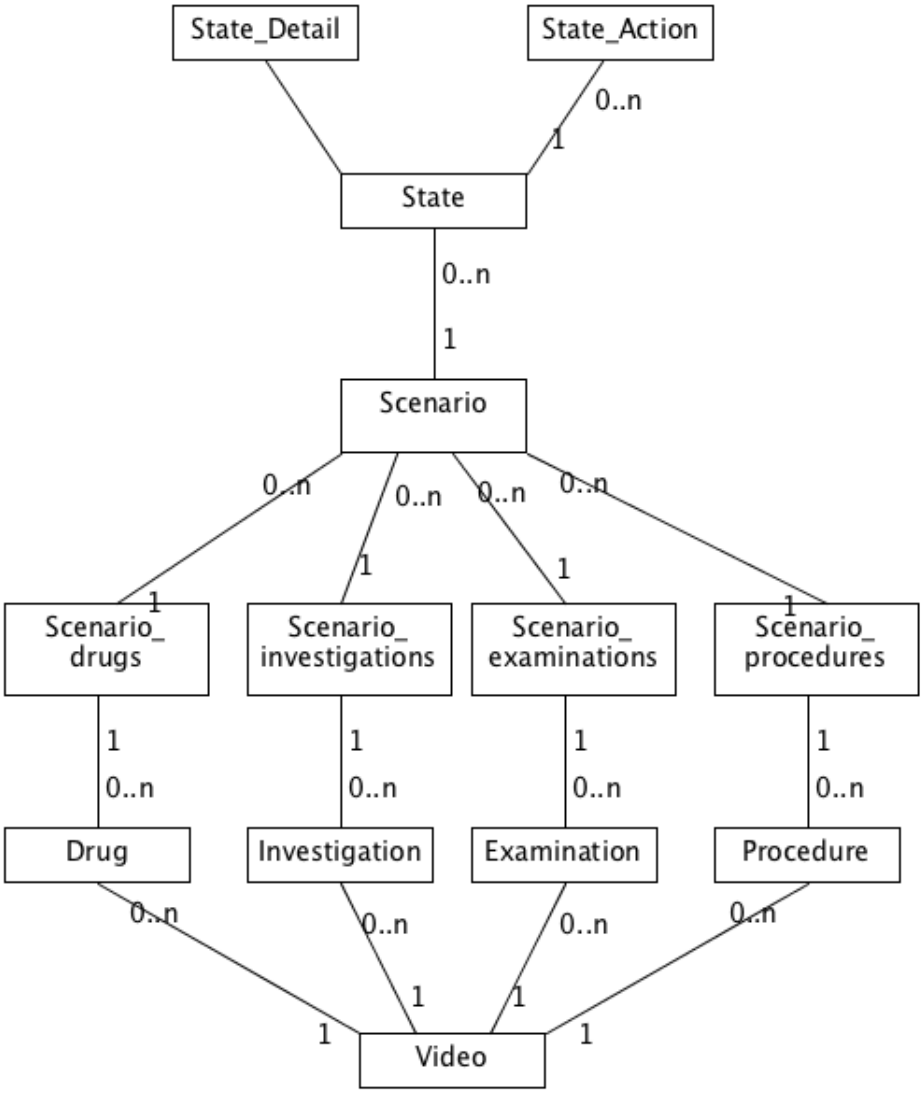

Figure 9: Database Design

The figure above is the database diagram. In order to achieve the reuses of scenario information, we design and implement many-to-many relationships between scenario and the scenario information that needed to be reused. In addition, we use abstraction in the design of scenario state with state detail table.

3.2.6 User Interface Design

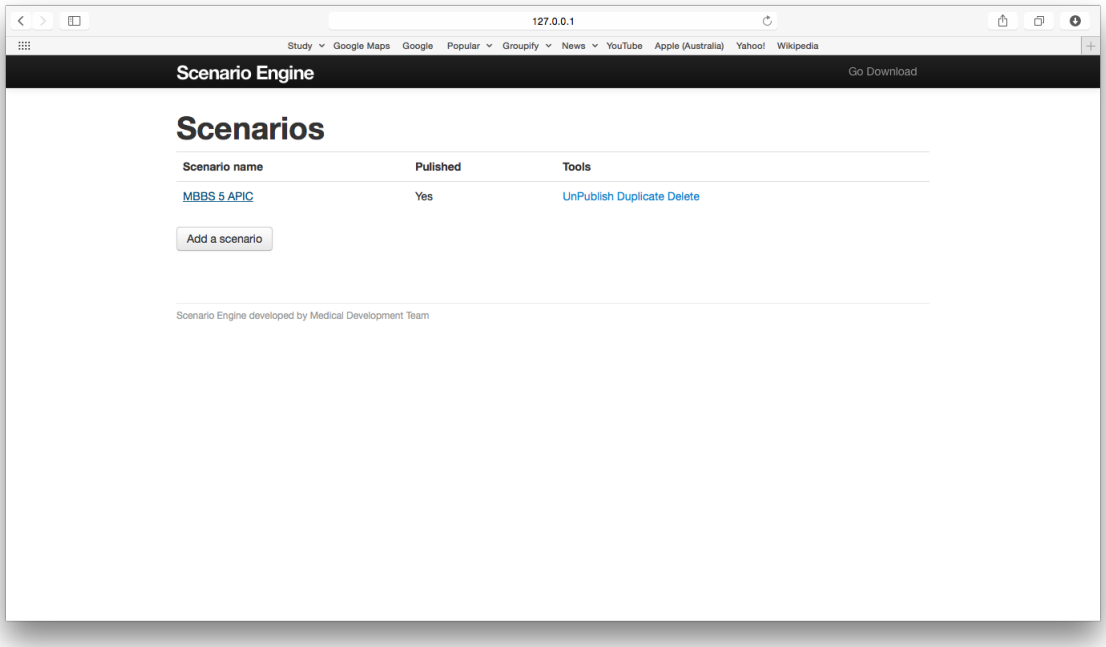

Figure 10: The Screenshot of Main Page

The Scenario Engine uses Bootstrap 2.3.2 with its built-in template stylesheet. All layout, buttons, forms and other components are followed the template of Bootstrap's template. The Bootstrap libraries offer readymade pieces of code that can pump life into a website. Therefore, we do not need to spend time working and implementing stylesheets. In addition, Bootstrap template has a professional user interface of website, which is better than self-designed and self-implemented stylesheets.

4 Methodology

4.1 Task Division Process

In order to work effectively and efficiently as a team we needed a process in order to divide the overall problem into several tasks so that different components of the system could be worked on concurrently. As the project is made of two main components this led to a difficulty in the division of labour that is fair and balanced for everyone involved in the project. In order to effectively manage this we split the application development into two different sections: the user interface, and the underlying logic. By splitting this project into three well defined components each member was given a component that they would be in charge of. This is not to say that other team members did not contribute to the system component of others, but instead the person who was in charge of a component did the majority of the work that was required for that particular component.

Now that we have three bounded components of the system, the process to delegate a task to a team member was to decide which component is most affected by the changes required to complete a task. In most cases the task was delegated to the person in charge of the most affected component, unless the codebase being affected was mostly completed by another developer in which case to avoid issues arising from incomplete or incorrect understanding of the code that developer was put in charge of that task.

4.2 Issues to Overcome

Over the course of the development of this project there were several key issues that we were required to overcome. The sources of these requirements were both intellectual and technical in nature, as we were developing the application in a language that was relatively unknown to us some issues arose from this lack of knowledge. Other sources included requirements change, client technical knowledge, and the generic requirement of the application. Some of the primary issues that have been considered are listed below:

- Graphical Representation of a scenario.
- Allow a non-technical person to create a new scenario.
- Support for other scenarios
- Platform Dependence

4.2.1 Graphical Representation of a scenario

The graphical representation of a scenario deals with how we were going to display the patient and any actions being performed on them. This issue began early within the development process as this application was being developed as an e-learning game we needed visuals to increase the engagement of the student using the software. The requirements related to this issue were of a high priority and included:

- Visual Feedback
- Actions being visualised
- Handover to player

The visual feedback was the main requirement which showed the necessity for the scenario requiring to be shown in a graphical representation. This requirement meant that the software needed to show the patients status at different points of the scenario and needed to be consistent. The requirement of having visualised actions is similar to the visual feedback but is in reference to any drugs, procedures, and investigations. While not all such actions needed to be shown in a graphical manner, some of the more important actions did have this as a requirement to ensure knowledge gaps are addressed. While a textual handover is possible, it is easy for students to miss necessary details about the scenario which could lead to the student failing the scenario. A graphical handover addresses this, especially with speech being a part of the handover as it is easier to listen and comprehend.

Over the course of the project this issue had two main options for its resolution:

1. Animated sequences
2. Live action video sequences

The animated sequences is the embedding of created animations of the handover, patient status, and actions. There are several options for creating these sequences: use a library for the iOS platform for programmatic animations, use a third-party program to generate these sequences and embed them as videos into the application.

The live action video sequences is the embedding of recorded videos into the application. In contrast to the animation sequences the way to create these videos is to use actors to play out these scenarios and taking the videos that were created into the application similar to the second option of the animation sequence.

Both of these options have benefits and issues. In terms of the animated sequences the main benefit is that there is no requirement of hiring actors which will reduce the cost of the creation of the scenarios.

Another benefit is that the size of the files created can be minimised without the quality of the sequence being decreased to an unacceptable level. Some of the issues that are involved with the creation of the animation sequences is that the technical and artistic skills required to create high quality sequences is beyond the abilities of the development team. Other than the learning curve for the development team being too high, the animation of scenario has the issue in that in order for the clients to be able to create the scenarios there are two possibilities:

1. The development team or an animator has to build overly generic models
2. Every scenario needs to have a developer involved to do the animations of the scenario

For the first possibility, there is a tradeoff which both are unacceptable. Either the development team animate a huge number of possibilities which is unacceptable for the time required to perform this task, or the generic animations are too generic and do not accurately portray the scenario due to the limited number of possibilities that can be chosen from.

For the live action video the main issue as mentioned previously is the cost of getting actors to play out the scenario, this combined with the complexity of setting up the scene for the scenario makes this a relatively unattractive option as well. While this issue is constraining the benefits include that it is much easier for a non-technical person to create the videos for a single scenario if required, videos are easily embedded within iOS applications using a highly compressed file format, and editing videos is an easier task than the editing of an animation.

In the project we made the decision to use live action videos to graphically represent the scenarios. This decision was made as the ability for the staff at the Adelaide Health & Simulation Center to add new scenarios was high priority and the live action videos was the option which allowed this with more ease when compared to the animated sequences that were also discussed.

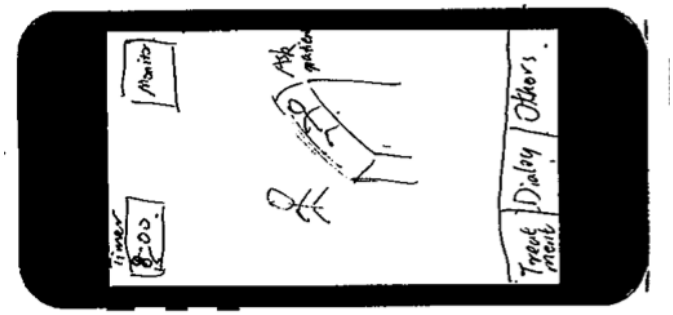

Figure 11: Early conceptual design for the animation sequence.

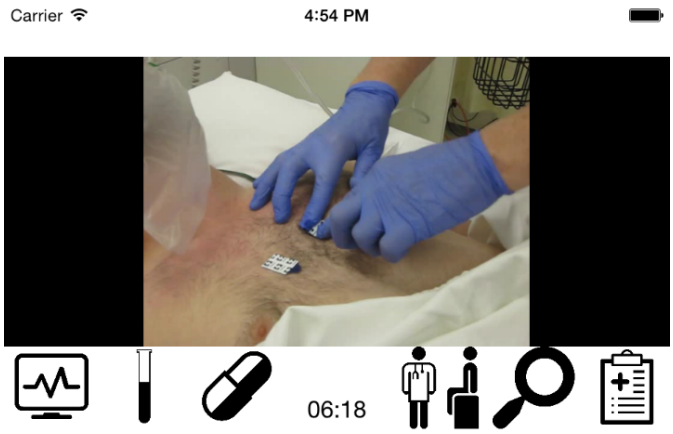

Figure 12: Final graphics with live action video

4.2.2 Scenario Creation

As discussed in the graphical representation issue, the client needs to be able to generate the scenarios as required without extensive technical knowledge. As developers we are not the average user so that our ideas of a simple interface for creation of these scenario inputs would be significantly different from those that a medical teacher or admin person would consider to be simple. Early on in the development process we were aware that the data format that the application would be able to read easily is not feasible to be created by the user as the data format requires relationships between data points to be defined.

In order to create these scenarios the user is to use a "*Scenario Engine*" which lets the user input data and generates the scenario files in the required machine readable format. For this issue we knew that the Scenario Engine would have to be easy to use, understandable, and generic. To solve this issue the following possibilities were considered:

- Desktop Application
- Website with file storage on server
- Website with database storage on server

The idea behind the desktop application is to have an application which is opened on the users computer and allows them to create and edit scenarios using a form system. In order to publish the scenarios the user generates the files and distributes them to the users of the application.

The two website options have a similar idea in terms of the front-end but the backend is completely different in terms of usage. For the front-end the websites will be a series of forms which allow the user to create, edit, and delete scenarios. The file storage method of storing scenarios is to have many folders with each of them representing one scenario. The database storage method allows for the usage of relationships between data.

The desktop application was an early consideration for the design of the Scenario Engine, this is due to a lot of negatives in using this method. There are two primary issues with this method of solving this issue: platform dependence, and scenario distribution. Unless we choose to develop the application in a language which runs in a virtual environment there will always be platform dependency issues with desktop applications. For scenario distribution the users have no way to get the scenarios except directly from the admin user, which would have to involve setting up a FTP server or similar.

The two websites both have the benefit of already being somewhere in the internet topology and forms are easy to generate for websites, but the file storage method has an issue with file redundancy where identical files which are associated with multiple scenarios are placed in multiple locations increasing

the storage requirement of the server. As the scenario model is relational a database storage method where different files and data can be associated with many scenarios is a better choice in the reduction of duplication on the server. Also the data format of the application (See Section 3) is being implemented in a relational way so a direct translation from the database to the required files are possible.

In the project we decided on using the website with database storage as this has the benefit of being reusable within both the storage and the creation/editing of the scenarios i.e. the user can select a drug from another scenario.

4.2.3 Multiple Scenario Support

After being able to create the scenarios, the application needs to be able to play them. This implies that the application needs to be designed to be able to handle multiple scenarios with a choice of which scenario to play. In order to solve this issue we needed to consider two different issues as a part of this: scenario storage, and scenario reading. While these two issues are inter-related i.e. the way that the scenarios are read is heavily dependent on how they are stored they still needed to be considered separately to come up with the best solution.

In terms of the scenario storage there were two main options: object archives, and property list files. While the implementation of object archives are much simpler with most of the work can be saved and restored in single lines of code, the property list files are easier to generate with the scenario engine as they are xml files with a particular format.

In terms of the scenario reading there were two options: create array of objects with relations, and create multiple dictionaries which are used to move through the scenario. The object array is mostly intertwined with the usage of a object archive, whereas the dictionary is intertwined with the usage of property list files.

In this project we decided to use the property list files with dictionaries showing relations as they are simple to understand and can be easily generated using the Scenario Engine with a simple relational model.

4.2.4 Platform Dependence

A major issue that came up during the early stages of development was the question of what operating environment the application should be developed for. While the client wished the application to be multi-platform early on in the piece we worked out that it was a better idea to build a proof of concept of the application on one platform well, rather than on all platforms with less refinement.

There were three options for this that we considered during these early stages of requirements:

- Web-based application
- iOS Application
- Android Application

While deciding on which platform the application should be developed for there are several major considerations to take into account including performance, network dependence, memory usage, storage requirements, and usability. As the application is required to work on mobile devices performance, memory usage, and storage requirements are high priority considerations due to the limited processing and memory capabilities of mobile devices. Network dependence needs to be taken into consideration as high dependence on networks a reliable internet connection is required throughout the entire gameplay which is infeasible on mobile devices in most cases.

Even though the web-based application covers the multi-platform requirement that the client had, this application had several downfalls in terms of performance and usability. The performance impact is due to the application needing to run a browser in the background which is usually light weight but will still

have a negative impact on the performance of the application. The requirement of the web application having a constant reliable network connection is a downfall as most network coverage is not constantly reliable. The usability is decreased on a web application as they tend to be less responsive than a native code counterpart.

In terms of the iOS application the code will be developed to run on a specific platform the compilers for the language (Objective-C) will have optimisations for the platform being used, greatly increasing the performance. The performance will also be increased by the code being run natively instead of being embedded within a web-browser like the web based application. As the iOS device is able to store files independent of the network the application will be able to be used when the network is down or unreliable, with the network dependence being in relation to the download of new scenarios. Unfortunately the storage requirements that iOS needs is much higher than the web-based application as files need to be stored on the device to achieve network independence.

As with the iOS application the Android application will be network independent, and have higher performance in comparison to the web-based application. Most of the positives of the iOS application can be related directly towards the Android application as well. Unfortunately the user demographic leans more towards the usage of iOS devices in comparison to Android devices which means that using Android for development will make the software less useful than choosing an iOS application.

For these reasons we decided on developing the application for iOS devices.

4.3 Software Development Techniques

As with all medium to large sized software development projects, techniques need to be used to manage the complexity of the tasks so that the project is constantly and consistently moving in a forward motion. In contrast projects that do not have these process tend to come across blocking points often which can slow down the development of the software due to duplication of effort, incorrect assumptions being made about the code base, some members not having work to do on the project, etc.

The software development process model was a modified version of the agile scrum model where the major modifications was to include some aspects of the mob programming methodology where every member of the group is involved in the code review process. We used the generic weekly sprints of the agile model, with a client meeting every second week to ensure that the last two sprints were adding value to the software. In terms of group interaction we had communication every day, where it was either in person, or by email. This daily communication made sure that every group member was aware of the current status of the project, as well as for any of the tasks that are currently being worked on. This high frequency communication made it easy to keep on top of the project progress to ensure that we met any client expectations of progress at the end of the major sprint.

As mentioned before we used a mob programming methodology for the code review where everyone discussed the feature being added, the code style, and any concerns on the new functionality for the system. This ensured that the master branch had the highest possible quality and we were all happy with the code style and quality being used in the project.

4.4 Version Control Management

In any project version control is a requirement for the production of high quality code. While for individual projects version control should be used in a group project version control is mandatory to allow for the collaboration of all team members. Over the course of this project we used Git as our version control system with the repository being stored on the Github hosting service.

In order to keep the master branch as clean as possible we decided to utilise the branching capabilities of Git heavily with each feature having its own branch which meant that only people working on a particular feature should be using that branch. In terms of the master branch we had some higher level branches which were the components of the system that were pulled directly from master. These branches were most often those that the feature branches were based on, in order to have a well structured repository

with minimal merge conflicts occurring we used the following process when strict control was required for a large system change:

1. Ensure component branch is up to date with master.
2. Create feature branch from the component branch.
3. Make necessary changes to system for the implementation of the feature, committing often.
4. Once implementation is complete, ensure system is behaving as expected.
5. Submit pull request to merge into component branch.
6. Each member of development team needs to review the code produced and accept the pull request.
7. If the component branch is ready to be pulled into master branch then submit another pull request for this.
8. Code review occurs again to ensure the master remains working.

When the changes required was not large or widespread we did not need to use the component branch system that we used in this process but this meant that the master was much more susceptible to low quality code being introduced so the pull request code review system was done much more stringently as there was no component branch to fall back on. The effects that this version control management process had was that everyone knew where the code base was, the quality of the code being produced, and had ownership or responsibility for it. This in turn had the effect of increasing code quality, less defects being found, and there were very few cases where a bug was introduced due to lack of knowledge about how a bit of code worked.

4.5 Testing

4.5.1 Medical Application

Due to constraints on the timing of the project duration it was required to prioritise the testing methodology for the medical application, especially when it came to the automated testing scripts. We used Objective C testing framework XCTest to perform our automated testing of the project. In terms of priority the logic components were the highest priority in terms of automated testing as it is difficult to test all possibilities in the user interface without the use of manual testing. The high priority ordering of testing and their automated test coverage within the logic components is:

Scenario Data 100%

State Machine 48%

Data Lists 40%

While the state machine and data lists automated testing is lower than wanted, manual test procedures have been used to detect bugs in either these cases which would boost the actual coverage of these to approximately 70%.

4.5.2 Scenario Engine

The testing of Scenario Engine has two main parts. The unit testing is mainly performed on the Model and Controller part of Ruby on Rails codes using Ruby on Rails built-in testing mechanisms. The View part is tested using Safari Web Inspector in order to make sure all the HTML and JavaScript codes performing correctly.With the Ruby on Rails testing framework, we have unit-testing on the 90% of codes in Controller part and all the codes in Model part. Also, we have performance testing done on the codes in View part by looking in the performance of website with Safari Web Inspector.

5 Final Product

At the end of this project, we produced two main deliverables, the Game and Scenario Engine. Both of products achieve the main requirements from our clients, and the overall performance are satisfied by the clients.

5.1 Main Deliverables

5.1.1 Game

Description The Medical Game is the iOS application that will allow medical students and professionals to hone their skills. The majority of functional requirements have been completed with some improvements to be made with the remainder of the software.

Achievements

Generic State Machine The game uses a state machine which allows for the scenarios to be easily changed with immediate effect.

Scenario Loading The game is able to load all the files associated with a particular scenario on selection.

Clean Interface The games interface is finished with high quality visual icons,

Video Playback When a particular action is selected a video is able to play, showing what is occurring to the patient. Also a handover video plays when beginning the scenario.

Patient Status Monitor As the scenario is played a patient status monitor changes showing what condition the patient is currently in.

Not implemented

Other Platforms: The Game only can be run on the iOS devices. Therefore, it is still need to be developed to be compatible with other systems.

Download new Scenarios: The download new scenario feature that can visit the Scenario Downloader to download new scenario and save it into assets folder in the game.

Game deployment: Deployment of the Game on Apple App store.

Reflections Basic reflection interface is available, but needs a lot more work to be useful.

Asynchronous Test Results Test results return straight away need to implement the ability for them to come after a time delay.

5.1.2 Scenario Engine

Description The Scenario Engine is the web service that can help administrator of the Game to create the game scenario using HTML forms without any software background knowledge. The main functional requirements are finished as scheduled, but some forms still need improvements.

Achievements

Scenario Generator: A web service that can take input of game scenario data from admin and generate a zip file that contains all files needed for gameplay.

Scenario Downloader A website that lists all the scenario that is ready to download.

Reuse of game data: The scenario data, such as drugs and procedures, can be reused in the later scenario generation.

Not implemented

Action form: A form that let admin to input the action information for game states.

Website deployment: Deployment of Scenario Engine on AWS EC2 server.

5.2 Difficulties

The major technical difficulty was the implementation of the generic scenarios. The most challenging aspect of this was the construction of the state machine to be read in from a file. This method of generating states of an application was completely new and had many design challenges that had to be faced including the formatting of the data, loading a new state, relationships between states, and finding what result a particular action had on the state. These challenges were time consuming and complex to overcome, but by using the iOS dictionary like property list files to generate the scenario, these issues were overcome.

5.3 Product Evaluation

5.3.1 Game

The clients are happy with how the Scenarios can be played using the application as well as the new interface for playing the scenarios. While the main functionality is working, some of the other requirements were not met meaning there is still some work to be done on the application including a better player reflection system, and a better testing framework.

5.3.2 Scenario Engine

The Scenario Engine performs exactly as our clients want, which means that it satisfies our main requirement specifications that help them to create scenario by themselves. In addition, the Scenario Engine using Bootstrap stylesheets with JavaScript function that makes the website easy to use and maintain. However, some features of Scenario Engine are still need to be implemented and improved, such as action forms and scenario forms.

A Project Contribution

Table 1: Project Contribution

Member Name	Time per week	Major Responsibility
Ben	13 hours	Backend of Medical Application
Iavn	14 hours	Scenario Engine
Richard	15 hours	User Interface of Medical Application

B Meeting Agenda

The *first* Medical software project meeting will be held in **4.36 in Ingkarni Wardl,** at **10am** on **Tuesday 25th March 2014.**

Agenda

Chair: Ivan Chen

1. Attendees

Halina Tam, Ali babar, Benjamin Ramsey, Ivan Chen, Richard Li

2. Project detail

Halina will give an introduction of a scenario that would provide an insight as to how the simulation sessions could run in Adelaide Health Simulation & Skills Centre

2.1. Improvised Questions

Medical Team will ask questions depending on the information given during the meeting.

3. Project proposals

Three proposals for different types of game will be discussed in this section in order to let client decide which one will be applied in this project.

- **Android-based Mobile Game**
- **IOS-based Mobile Game**
- **Web-based Mobile Game**

4. Milestones

Medical Team will present milestones planned so far, and discuss it with clients

- **Research & Design Phase – 8th April 2014**
- **Initial game version – 1st July 2014**
- **Further features based on feedback**

5. Other Issues

5.1. Access permission

Medical team may require some access permission in Simulation Centre. And we may need a quick tour in Centre to know how to find medical cases for the project.

5.2.Meetings

Regular fortnightly client meeting times need to be arranged.

The *second* Medical software project meeting will be held in **4.36 in Ingkarni Wardl,** at **10am** on **Tuesday 8th April 2014.**

Agenda

Chair: Ivan Chen

1. Attendees

Simon Patten, Halina Tam, Benjamin Ramsey, Ivan Chen, Richard Li

2. Project requirement discussion

Medical Team will discuss about the game design

2.1. Prepared Questions

- **How many scenarios game has?**
- **How many kinds of patient for each scenario? (Different ages, gender, initial stats)**
- **What the Drugs and equipment for each scenario could be used. (It should be a list, Do every scenario has the same list?)**
- **How the drugs and equipment affect the patient (React by change of the stats or other thing)**
- **How long the timing is suitable?**
- **The definition of all medical terms, such as ECG, HR, BP, RR and so on.**
- **After starting a scenario, what the students could know about the patient (background, gender, age or some medical history about the patient), show them this information with text or by interacting with the patient?**
- **Do we need a communication function for students to interact with the patient during the treatment (by dialogue for example) or just let students know the currently situation by text way.**
- **Do we need exam during the treatment? (If we need, we have to know different types of exam, for example neck exam, respiratory exam, blood exam and so on) Or we just put all the result of the exam with a page to students automatically.**
- **How to evaluate the playing of the student? What the result page and reflection page would include for each scenario?**
- **Do we need tutorial for the game? The tutorial may teach the students how to deal with patient for correct drugs and equipment or some definition of diseases.**
- **Do we need different types of rooms (emergency room, surgical room, clinic room...)**

- **If the students are stuck in the scenario, do we need let them call consultant to have a hint?**

2.2. Improvised Questions

Medical Team will ask questions depending on the further information given during the meeting.

SEGP Medical Team

Meeting Agenda for Tuesday 29th April, 2014

Atendees: Halina Tam, Justin De Blasio, Ben Ramsey, Ivan Chen, Richard Li

1 Requirement Validation

The development team will discuss the first draft of the software requirements specification so that we can discuss and validate the requirement and the design.

2 UI Design Presentation

The dev team will present the current design of the game and discuss with the client to get the feedback.
The structure of UI design as shown below:

- Start Menu
- Game Screen
- Summary Page

SEGP Medical Team

Meeting Agenda for Tuesday 6th May, 2014

Atendees: Simon Patten, Halina Tam, Justin De Blasio, Ben Ramsey, Ivan Chen, Richard Li

1 Requirements Validation

The development team will discuss and validate the requirements for the software to be developed.

2 Video Recording

The development team will discuss with the client about the strategy for the game.
Issues to discuss:

- Filming location
- Scenario to film. The scenarios that we have access to are:
  - MBBS 5 APIC - Hyperkalaemia V1
  - MBBS 5 APIC - Major Haemorrhage
  - DKA
  - SVT
- People involved with the video production.
- Booking a time for the filming.

3 User Interface Presentation

The team will reshow the user interface of the application as it currently stands for the benefit of Simon to discuss any changes that are required.

SEGP Medical Team

Meeting Agenda for Tuesday 20th May, 2014

Atendees: Simon Patten, Halina Tam, Justin De Blasio, Ben Ramsey, Ivan Chen, Richard Li

1 Briefing on operating platform

The development team will present and discuss the briefing document about the Platform we will develop for. The decision has to be made in the meeting in order to make sure the development of the game will follow the clients requirements.

2 Initial Scenario to be developed

We picked the Renal Failure with Hyperkalaemia as the scenario we will use in the game. So the developing team will discuss with client about the further information of the scenario. Such as possible people may be involved in the video making and some medical stuff in the scenario.

SEGP Medical Team

Meeting Agenda for Tuesday 3rd June, 2014

Atendees: Halina Tam, Simon Patten, Ivan Chen, Ben Ramsey, Richard Li
Location: Adelaide Simulation Centre
Time: 10am

1 Scenario Development

The team will discuss the plan for scenario development, covering items such as:

- The procedure of the scenario in the game.
- Clarification of detail for the scenario
- The plan for creating the videos for the scenario, and booking a time to create the video footage.

SEGP Medical Team

Meeting Agenda for date

Attendees: Simon Patten, Halina Tam, Justin De Blasio, Libby Kentish, Ali Babar, Ben Ramsey, Ivan Chen, Richard Li
Location: Medical School North, Ground Floor, NG42.

1 Prototype Presentation

The development team will show the current state of the mobile application being built for feedback from the client.

2 Discussion for future work

In this part of the meeting various aspects of the project need to be discussed for work to progress on schedule. Some of the aspects to be discussed include:

- The way that the application will visualise the scenario, in particular will video be able to be used or is there a requirement for animation to be created.
- For the summary on the performance of the player, how should the marks be calculated.
- What are some examples of questions that should be asked within the users reflection page of the application.

SEGP Medical Team

Meeting Agenda for Tuesday 2nd September, 2014

Attendees: Simon Patten, Halina Tam, Ali Babar, Ben Ramsey, Ivan Chen, Richard Li
Location: Medical School North, Ground Floor, NG18.

1 Prototype Presentation

The development team will show the current state of the mobile game being built for feedback from the client.

2 Discussion for future work

The major discussion point for future work will be the scenario engine. Some of the aspects of this that will need to be discussed are:

- The way that input will be done by the admin. This could be done by file upload, web forms, or a combination of the two.
- How do you want students to be able to download the scenarios. This can be done within the application, or get the users to download them and place them in a known location.
- Are there any constraints on the response time for the scenario engine.

SEGP Medical Team

Meeting Agenda for Friday 17th October, 2014

Attendees: Simon Patten, Halina Tam, Ali Babar, Padraig O'Leary, Ben Ramsey, Ivan Chen, Richard Li
Location: Medical School North, Ground Floor

1 iOS Application Prototype Presentation

The development team will demonstrate the current state of the iOS Application to the client for feedback.

2 Scenario Engine Prototype Presentation

The development team will demonstrate the current state of the Scenario Engine to the client for feedback.

C Meeting Mintues

Meeting Minutes

Date: 25/03/2014

Attendees: Halina Tam, Benjamin Ramsey, Ivan Chen, Richard Li

Duration: 70mins.

Project Detail

We were given an example scenario form which gives the following information:

- Patient Information
- Participant Briefing (What the user of our application will be given as a summary)
- Scenario Overview (Including time and summary of scenario progression)
- Resources available to the user.
    - Basic Resources
    - Equipment and drugs available
- Simulator parameters and progression of scenario
    - Based on treatment or timing.
- Effects of some possible interventions
- Proposed Correct Treatment.

This will give a basis to the scenarios that the application will go through for the training of the medical students. We have asked for some extra examples so that we can see if there is a possibility of standardising these scenarios to be fed into our application.

We may need to include some form of admin access to be able to generate these scenarios with ease. e.g A form which takes a similar format to that of the scenario form example.

Project Proposals

We gave the client three proposals:

- Android based mobile game
- IOS based mobile game
- Web based mobile game

For each of these we gave the client some of the positives and negatives of each of the proposals. As the client is pushing for a multi-platform game it is likely that development will be done in the

form of a web based game but the positives and negatives we gave may influence decision to one of the mobile application proposals (Android or IOS).

Milestones

Milestones have been shown and approved by the client. The client understood and agreed most of milestone plans, expect our first millstone (R&D Phase) might be a bit short (we might need more than 6 weeks). And we agreed this is an initial brief milestone plan and we will adjust it via the project.

Miscellaneous

- During the Design and Research Phase, it is required to visit the centre so that we can see what is going on during the simulation classes to see what the application is to simulate.
- Consequences of decisions should be implemented.
- Different graphics need to be looked into, current application (Resuscitation) is 2d and has very little visual cues to what is happening (a mostly static image)
- Target audience of program are the Medical Students (doctors in training)
- Some form of distraction to the user may need to be simulated.
- Regular meeting timeslot and visiting time will be confirmed by Halina via Emails (need to be confirmed by Simon)

Meeting minutes

Date: 08/04/2014

Attendees:
Kevin Maciunas, Halina Tam, Simon Patten, Justin De Blasio, Ben Ramsey, Richard Li, Ivan Chen
Duration: 80 minutes.

1 Requirements added

- Player needs to be able to engage in conversation with the patient. This can be done with limited options relating to different ways to diagnose the patient.
- There will be a number of core drugs and equipment (20) which will be available to the player. Other drugs and equipment will be scenario specific.
- Clients need to be able to add scenarios to the game, without programming it. (Details in section 3)
- Platform needs to be iOS as demographic of audience favours apple products.
- Scenario length need to be kept short to hold players interest.
- Reflection page to be accessed after completion of the scenario. More details in section 2.
- Users need to be able to view past reflections and be given an opportunity to comment on them (reflect on change of thinking)
- Visual feedback & deterioration of the patient need to occur, as well as be in keeping with the current game state.
- Player needs to be provided a summary on their performance, with an explanation in where issues occurred and why they were issues.

2 Reflection Page

The reflection page is a major part of this project, this page allows the player to reflect on their actions and reasoning behind their actions. The reflection page is to consist of questions of both free text and specific (short/multiple choice) formats. The free text questions will be limited to be approximately 150 words, and are for questions relating to their reasoning of actions rather than the actions themselves.

3 Game Scenarios

The client advised that development should be kept simple to begin with, one scenario in one type of location (ED). This will give the foundation of the rest of the scenarios and let us be able to develop a scenario engine which will allow users who do not have programming experience to add scenarios to the game.

4 First2Act

To show us an example of a well developed serious game in this field, the client showed us First2Act. This game uses video sequences to visibly show the deterioration of the condition of the patient. Each scenario is to be played within 8 minutes but there are more possible options to be undertaken than there can be done in the time limit. Conversations between the patient and the player is possible with a fixed number of dialogue options, while talking to the patient the countdown timer is stopped. Video sequences of procedures and patients were with different people, this is not good as the players will notice this will not be as immersed within the game. The feedback system of the game gave a score out of 30 but did not say what constituted a perfect score, nor what the player had done wrong.

Meeting minutes

Date: 02/06/2014

Attendees:
Simon Patten,Halina Tam, Nicole ,Justin De Blasio, Ben Ramsey, Richard Li, Ivan Chen
Duration: 90 minutes.

1 Scenario Development

1.1 Equipment List

This section should be renamed to procedures

- Make people think about personal protection equipment
  - Some cases need hand gel, gloves, and/or mask.
  - Majority of other games do not take this into account.
  - This needs to be an automatic reflex
- Penlight should only be needed for neurological scenarios
- Medical References should not be provided, they can already find the information if required, but need to build competency under a time limit.
- Defibilerator
  - Both AED and manual defibleration need to be present
  - While pacing scenarios common, students do not know how to do it.
  - Machines have a standardised interface.
- ECG required
- Suction may be required for some scenarios (though maybe not always)
- IV
  - Canula
  - IO
  - Central line
  - **Have IV as a subcategory in this list**
- Syringe driver or infusion pump not required.

1.2 Drugs

The drug list should be replaced by a prescribing chart, which the player must fill in. There are several requirements for this prescribing chart:

- Users must type in to the prescribing chart the drug that they require to treat the patient.
- The user must type in the actual drug name, rather than the brand name of a drug. This is a legal issue that must be followed.
- The user must spell the drug name correctly, otherwise the prescription is illegal and invalid.
- The different procedures should still be a part of a list, but the drugs required should be on a list hidden from the player, and they must know what they need.
- Immediate feedback for those drugs that are not on the list, a time penalty still exists as they need to type the entire prescribing chart again.
- When creating a scenario, both beneficial and non-beneficial drugs should be placed on the internal list. If the player wants a non-beneficial drug. At the end of the scenario a player gets feedback and the non-beneficial drugs that the player used are listed there as a bad point.
- The prescribing chart needs to be a complete one with both dose and route as well as the drug that needs to be given.
- After an initial drug list has been created, a more complete one can be generated from what the users have attempted to give the patient. This needs to weed out any data that are brand names of drugs, and spelling mistakes.

Meeting minutes

Date: 06/05/2014

Attendees:
Simon Patten, Halina Tam, Justin De Blasio, Kevin Maciunas, Ben Ramsey, Ivan Chen, Richard Li
Duration: 60 minutes.

1 Requirements Validated

- Operational Environment
- Patient Dialogue
- Scenario Length
- Player Reflection
- Visual Feedback
- Player Summary
- Decision Consequences
- Ability to skip any video sequences.

2 Requirement Tweaking Required

For FUNCREQ003 (Scenario Length) a time of 10-15minutes per scenario was deemed acceptable.
In terms of Visual Feedback, the patient deterioration needs to be as accurate as possible, this could be quite expensive in terms of amount of stored data required for each scenario (number of videos required increase)
Initial prototype with a single scenario needs to be done to be directed towards a lower level of medical student (level 4). This is to limit the amount of complexity for the initial game prototype.

3 UI Presentation

Showcased the basic program flow of the UI as it currently stands. No major issues with it were brought up during this meeting. Some more refinements will be made before next weeks meeting, which will be showing Simon our progress on the UI.

4 Questions for Simon Patten

Which scenario should be chosen for the initial game before creating the generalised scenario engine.

5 Scenario Videos

The videos for the scenario that we will be developing will need to be planned thoroughly. Many factors need to be taken into consideration including: video camera availability, which scenario to choose (this will be up to Simon), actors, location (possibly the Simulation Centre), time availabilities, any additional people that are involved with the scenario, setup time for scenarios is a considerable amount, and videos for the equipment procedures will need to be prioritised so that only major procedures need to be shown.

Meeting Minutes 02/06/2014
———————————————
Category rename to procedures
==========
Make people think about personal protection equipment.
Some cases need hand gel, gloves, mask
Other games do not touch on the hygiene of the doctor.
PPE should be an automatic response.

Penlight only useful in certain scenarios (neurological)
References on medical dictionary, should not be provided. —get rid of

AED and manual defiliberator to be done.
Pacing scenarios common but students do not know how to do this.
Standardised interface.

ECG required

Suction a possibility (possibly not needed)

IV required canula, IO, central line (have IV as a subcategory)

Medical Notes == Evidence
Good for prescribing errors
Prescribing chart — need to write part of this
Use to choose which drugs, type in… need to match a list done by the scenario generation
Only write the actual drug name not the brand name (legal issue with brand name prescriptions)
Spelling is a requirement (legal issue)
If using a list, visual cues will give an issue.
Tools should have visual cues, but drugs should not.
Need immediate feedback
Have typing as a time penalty
Beneficial/non-beneficial drugs, have some anticipated errors.
Build up a drug list, using data that they have put incorrectly. (Recogniser machines)
Need to subscribe dose and route, have them fill out a standard prescription (have them fill out whole thing before feedback)

Don’t need syringe driver or infusion pump. (more for the centre than the application)

On alpha testing, can redefine all of this.
Bigger list

Should not have dialogue with patient if unconscious

Trigger is listed under additional

Videos
======
May still need to do animation.
Don’t need to see all procedures
Communication breakdown is a major issue
Multitasking needs to be taken into account.
Balance is required.
Engagement over fun.
Model as much as possible.
Visualisation is the major aspect.
In the time after administration visualisation of the patient is important.

Dukes university have a avatar system for medical students.
Lockheed Martin have some papers as well on simulation.

Possibly have interaction with the screen for simple procedures.

Real time but maybe sped up slightly (efficiency)

Platform
======

Meeting Minutes 09/09/2014

Prototype Evaluation

Scenario Engine
- Authentication required
- Have library of old entries on form

Video preparation
- Clearly have phases denoted
  - currently known phases
    - nurse handover
    - patient is unconscious: no video required for history as they are unable to respond.
    - patient is conscious but unstable: can have the history videos as required
    - patient is stable: can have required history videos
- Cut sequence lists
  - should we have one for each type of history speech or only current feeling.
  - if oxygen is available for the scenario then the scene should be shown with both oxygen mask on and off.
- Have sequences for each state in the application
- Themes of them
  - ???
- Max length:
  - 20 seconds is approximately 8MB of video at standard compression. May want to have a look at how to compress video further while not being used to save space and just decompress when it is required.
  - 1 minute (24MB) is the absolute maximum for a scene and we want to ensure that this is done as sparingly as possible.
- Formatting:
  - mp4 with MPEG-4 Part 2
  - .mov
  - .mpv
  - .3gp

Key procedures
  - mask
  - defiblirator
  - no compressions
  - ECG
  - Drug administration
  - ALS algorithm

Handovers
- Male has been arrested in some ward

Defibilator shows if shockable or non-shockable

Some drugs
  -adrenaline 1mg
  - amiodarone 300mg
  - insulin
  - dextrose

Have options for different blood tests
  - NVA 20
  - Coags

Re-evaluation of ABCDE