**Technical Documentation**

**Architecture**

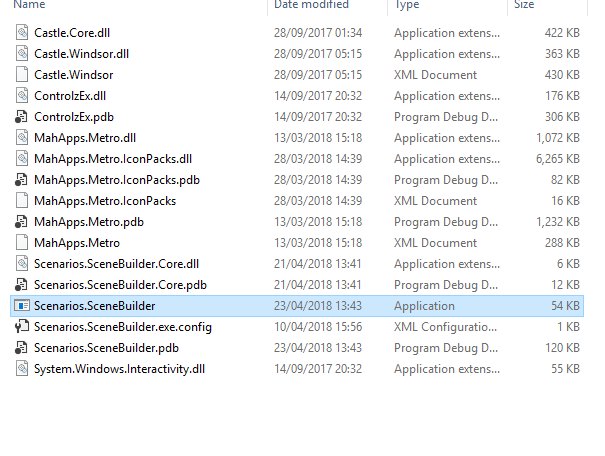
The architecture that we used in our project was MVVM design pattern, within the software engineering side we felt like this would be the best option to have Models, Views and View-Models that we could control individual to set up different aspect of the design that the client was wanting. This gave us the control of the Views as such that when we were defining in the XAML we can design the of the panel, we would put no logic behind the Views which instead we would bind the aspects of the View to the corresponding View-Model. A View-Model would be the model of the corresponding view of our application, so this would allow us to create a model for each individual view which would allow use to only expose the relevant data to individual views. The Model would be responsible for exposing data in a wat that is easily consumable by the type of GUI framework we choose to work within.

The more relevant benefits to why we chose to use the MVVM pattern was because of during development process, we could all work independently on individual components. As one of our team could work on the views of the project and the other to work on the view models and models. A bonus which came from using this pattern was that it helped splitting up tasks so when letting people choose what they would like to complete within their sprint there was lots to chose from. It would also allow us to create end to end testing which would allowed us to make sure if our application ran as intended when we gave a certain input, this helped with been discovering bugs and aspects which we thought we could improve. By splitting up sections of the project such as the ‘View’, when we deliver the project to the client and if they didn’t like the user interface then we would be able to change to the specification that they would want to see from the user interface of this application. Since we have been able to use ‘View Models’ to control this means that we can separate out the UI logic from the views to be present in the View Models, this benefits us because of instead of having one massive class that controls everything and navigating through that in the future would be not ideal therefore future expansion this is a great benefit of giving the developer the best tools to find what they are wanting to expand

**Best Aspects**

We thought the use of dependency injection would be a good tool to use within our project. This would allow us to delegate responsibility for construction and the dependency injection pattern would let us provide dependencies after an object has been created. While using dependency injection it would however most importantly make it easy for us to follow the ‘Single Responsibility Principle’ throughout our code. With the use of dependency injection, we were able to do more tests on the application was we can now use mock dependency injection which will allow us to see if the mock objects will behaviour the same as the actual object will when the application is running to help prevent bugs when shown and delivered to the client.

To highlight what we think is a great feature within our application is the simple design that allows the user to add and remove features from a story scenario. For instance, the use of ‘User Controls’ throughout the application that can give the user a large array of option that can be changed from each scenario is which will give the application the ability for the user to have heavy customisable scenarios. The ability to add extra features in the future will be very straight forward with a Boolean and Slider control already within the solution.

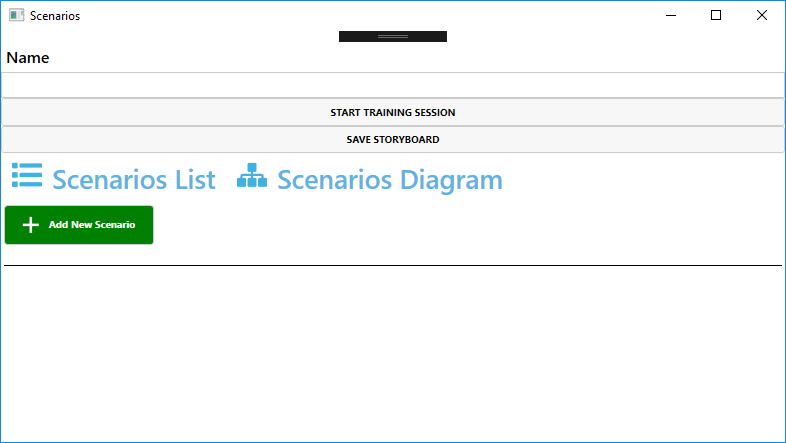
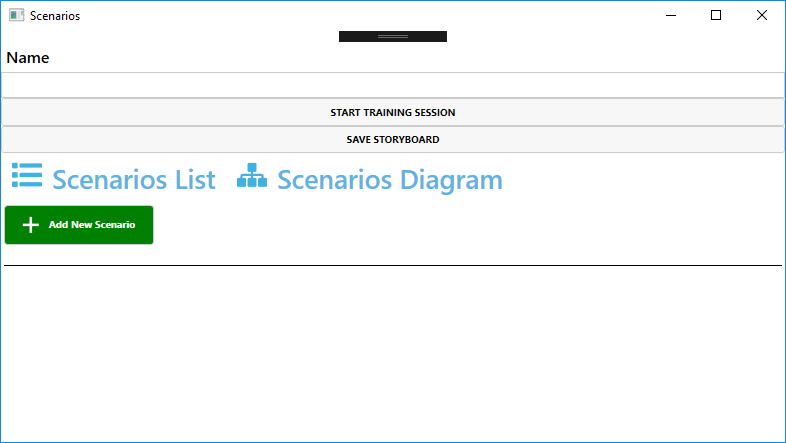
**User Guide**

1. To run the application, start by clicking the .exe application.

2. You will be prompted with windows, which you will be able to choose a ‘New Storyboard’ that will allow you to create your own customisable story, or ‘Load Storyboard’ which you will be able to select a previously used story from before and ‘Setting’ which will allow you to change certain directives that application uses as default.

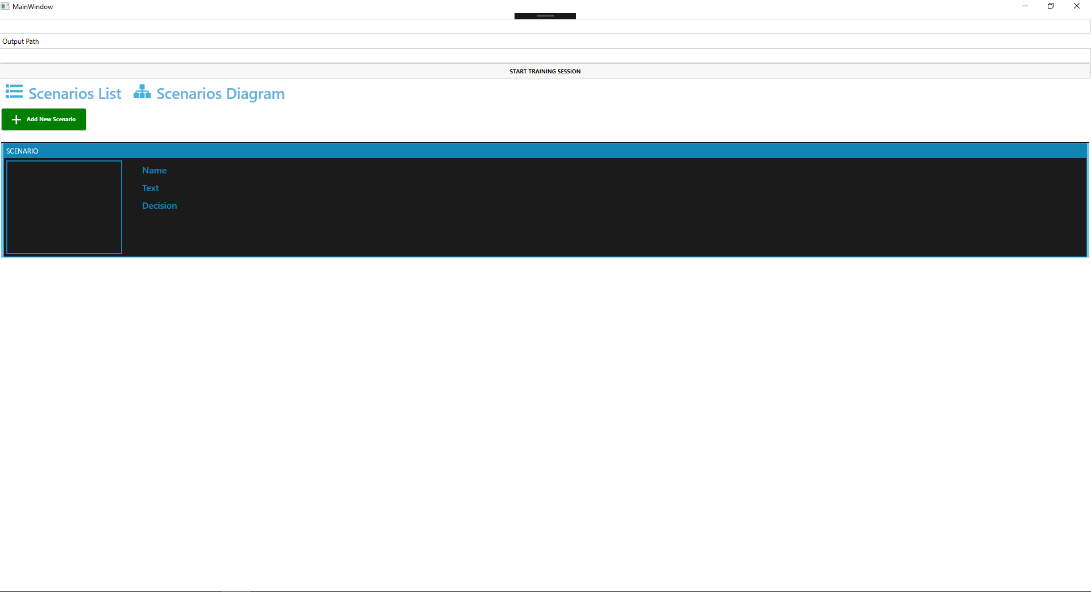


3. You will now be able to name the story board which you are wanting to create.



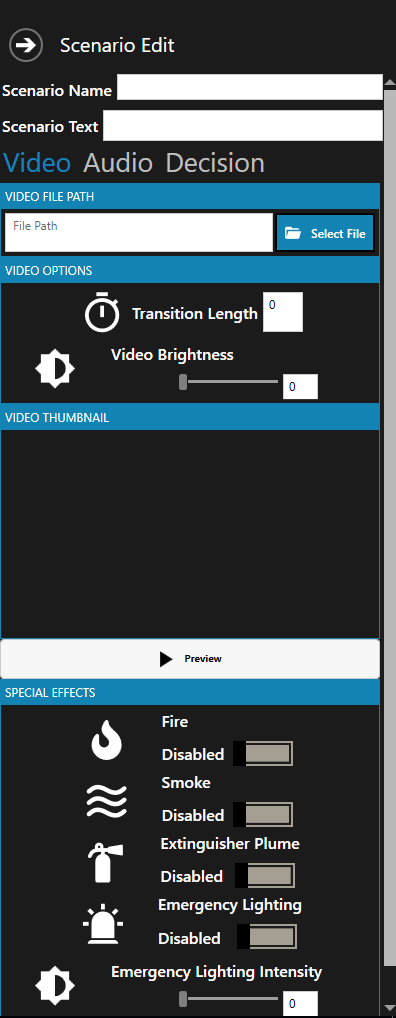
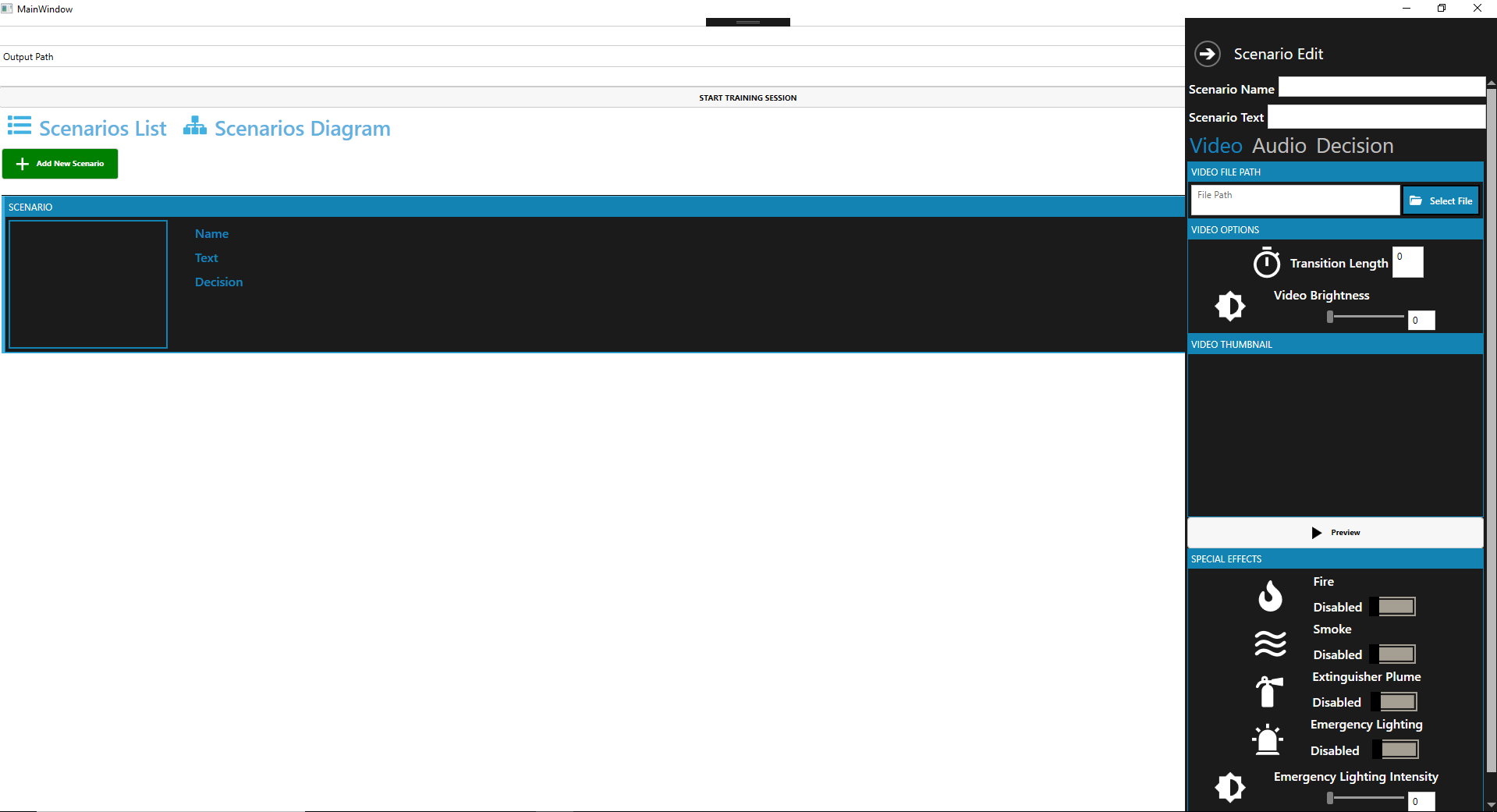
4. To add scenarios to your story board, click the ‘Add scenario’ button highlighted below.

5. To adjust the affects to a scenario, select the scenario in which you are wanting to customise as seen below.

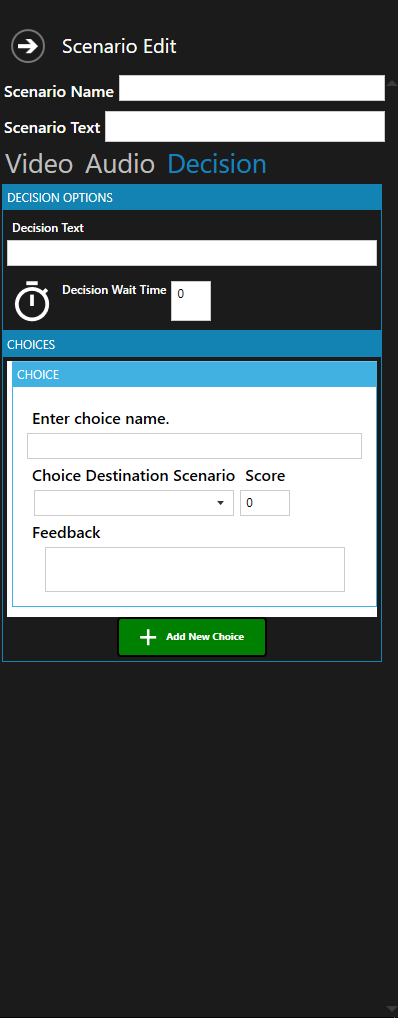


Click here

6. You will now to be promoted with all the default option that you can select on a selected scenario.

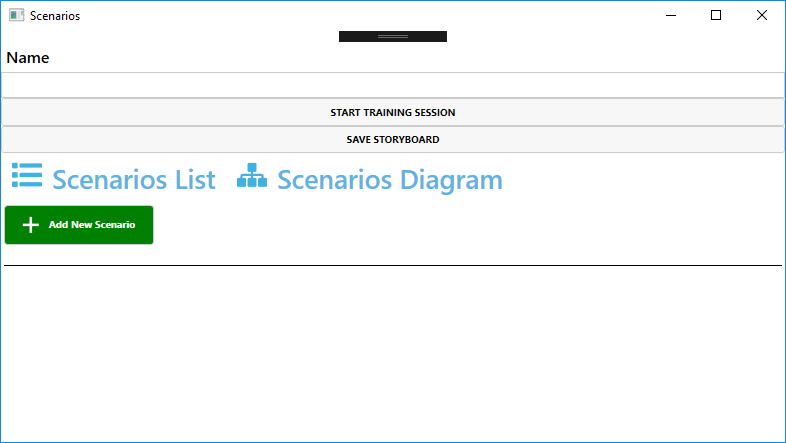


7. To add the video which you are wanting for a certain scenario you can do this by browsing to the ‘Video Panel’ and clicking the highlighted button, in which you will be able to navigate to the desire location of video.

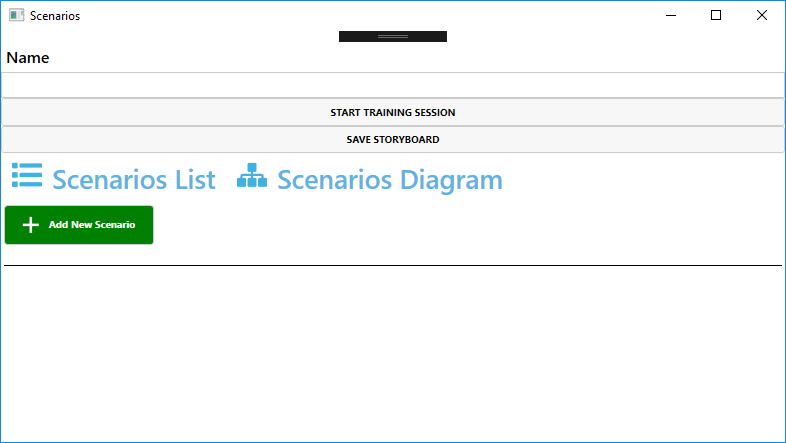


8. To customise the choices the trainee will experience selected the scenario in which you are wanting to choose the next video scenario for by navigating to the ‘Decision Panel’ which you will be able to add a choice by the highlighted button below. With a choice added, you will be now able to customise the choice which you are wanting from the user control displayed below.

9. To launch the video application, press the button which is highlighted below.



10. To save a created story board, click the highlighted button below.



11. To load a created story board within the application, go back to step 2 and click ‘Settings’ you will be greeted by the interface below. Click the highlighted button to open the directory to navigate to where the saved story board is. Save your settings and click the ‘Load Storyboard’ from step 2.

