

Software Requirements Specification

VR Training Tool



February 13, 2023

Version 1.0

Jeremy Cavallo, Robert Tedeschi, Slav Ivaskiv, Luka Cvetko

**Table of Contents**

[1 - Introduction 2](#_Toc1757449070)

[1.1 Purpose 3](#_Toc145924786)

[1.2 Scope 3](#_Toc838756493)

[1.3 Definitions, Acronyms, and Abbreviations 4](#_Toc1430095643)

[1.4 References 5](#_Toc1268626508)

[1.5 Overview 6](#_Toc177507914)

[2 - General Description 6](#_Toc1751466805)

[2.1 Product Perspective 7](#_Toc1077487100)

[2.2 Product Functions 7](#_Toc1192433771)

[2.3 User Characteristics 9](#_Toc1502651662)

[2.4 General Constraints 9](#_Toc1730400582)

[2.5 Assumptions and Dependencies 10](#_Toc1727163123)

[3 - Specific Requirements 11](#_Toc1207520457)

[3.1 External Interface Requirements 12](#_Toc1685065088)

[3.1.1 User Interfaces 12](#_Toc717034095)

[3.1.2 Hardware Interfaces 15](#_Toc2070471492)

[3.1.3 Software Interfaces 15](#_Toc363984905)

[3.1.4 Communication Interfaces 15](#_Toc2127390842)

[3.2 Functional Requirements 16](#_Toc1814522760)

[3.2.1 Functional Requirement 1 16](#_Toc1065669163)

[3.2.2 Functional Requirement 2 16](#_Toc1191446135)

[3.2.3 Functional Requirement 3 17](#_Toc1394205246)

[3.2.4 Functional Requirement 4 17](#_Toc1215013582)

[3.2.5 Functional Requirement 5 18](#_Toc1352220981)

[3.2.6 Functional Requirement 6 18](#_Toc473835384)

[3.2.7 Functional Requirement 7 19](#_Toc1732427666)

[3.2.8 Functional Requirement 8 19](#_Toc1990341382)

[3.2.9 Functional Requirement 9 20](#_Toc1278014934)

[3.2.10 Functional Requirement 10 20](#_Toc1267557714)

[3.2.11 Functional Requirement 11 21](#_Toc1257394715)

[3.2.12 Functional Requirement 12 22](#_Toc1475858869)

[3.2.13 Functional Requirement 13 22](#_Toc1474182523)

[3.2.14 Functional Requirement 14 23](#_Toc1249436539)

[3.2.15 Functional Requirement 15 23](#_Toc2145183353)

[3.2.16 Functional Requirement 16 24](#_Toc1959847644)

[3.2.17 Functional Requirement 17 25](#_Toc1392456969)

[3.2.18 Functional Requirement 18 25](#_Toc228379852)

[3.2.19 Functional Requirement 19 26](#_Toc454219197)

[3.2.20 Functional Requirement 20 26](#_Toc1085403674)

[3.2.21 Functional Requirement 21 27](#_Toc449721890)

[3.2.22 Functional Requirement 22 27](#_Toc1948915660)

[3.2.23 Functional Requirement 23 28](#_Toc54385796)

[3.2.24 Functional Requirement 24 28](#_Toc415587446)

[3.2.25 Functional Requirement 25 29](#_Toc1219265722)

[3.2.26 Functional Requirement 26 30](#_Toc308671757)

[3.2.27 Functional Requirement 27 30](#_Toc1747738554)

[3.2.28 Functional Requirement 28 31](#_Toc1514226839)

[3.2.29 Functional Requirement 29 31](#_Toc509241994)

[3.2.30 Functional Requirement 30 32](#_Toc1511069265)

[3.2.31 Functional Requirement 31 32](#_Toc162250371)

[3.2.32 Functional Requirement 32 33](#_Toc1856514991)

[3.2.33 Functional Requirement 33 34](#_Toc1057028586)

[3.2.34 Functional Requirement 34 35](#_Toc833263158)

[3.2.35 Functional Requirement 35 36](#_Toc1304176797)

[3.2.36 Functional Requirement 36 37](#_Toc1790334566)

[3.2.37 Functional Requirement 37 38](#_Toc331949182)

[3.2.38 Functional Requirement 38 39](#_Toc563095042)

[3.3 Non-Functional Requirements 39](#_Toc688984840)

[3.3.1 Performance 39](#_Toc359067353)

[3.3.2 Reliability 40](#_Toc665129197)

[3.3.3 Availability 41](#_Toc1295242310)

[3.3.4 Security 41](#_Toc1650468633)

[3.3.5 Maintainability 41](#_Toc765827317)

[3.3.6 Portability 42](#_Toc1952051132)

[3.4 Design Constraints 42](#_Toc348003667)

[3.5 Logical Database Requirements 43](#_Toc571695755)

[3.6 Other Requirements 44](#_Toc214592514)

[4 - Analysis Models 44](#_Toc1307633330)

[4.1 Data Flow Diagram (DFD) 45](#_Toc933473528)

[5 - Appendices 45](#_Toc1183947528)

[5.1 Appendix A: Development Plan 46](#_Toc852926843)

[5.1.1 Project Overview 46](#_Toc378764180)

[5.1.2 Scope, Purpose, and Objectives 46](#_Toc920899750)

[5.1.3 Team Organization 47](#_Toc1772888502)

[5.1.4 Problem Resolution Policy 48](#_Toc775699071)

[5.1.5 Project Plan 48](#_Toc1018999783)

[5.1.6 Configuration Management Plan 50](#_Toc1091775725)

[5.1.7 Technology 50](#_Toc2091021164)

[5.2 Appendix B: Priority Matrix 51](#_Toc98815471)

# Introduction

## Purpose

The purpose of this document is to provide an extensive outline of the requirements we will be implementing in our virtual reality medical training tool. We will outline the overall scope of the application, including what we will be responsible for implementing as well as things that we are not responsible for implementing. We will discuss a general description of the training tool, including what the tool will look like to users, a high-level overview of the functions needed to create the tool, the characteristics of the users of our device, and any constraints, assumptions, or dependencies required to produce the training tool. Next, we will give an in-depth description of all the requirements themselves, which will be the bulk of this document. This includes the discussion of functional and non-functional requirements.

The intended audience of this document is our client, our managers, and anyone else looking to critique the requirements for our virtual reality medical training tool. This document is a living document and will be updated as we receive feedback on our requirements and adjust throughout the development process.

## Scope

The main goal of our application is to provide hands-on medical training through virtual reality. The two existing options for medical training, online instruction and in-person classes, each have their own drawbacks. Online classes provide no hands-on experience, meaning students will be unable to practice and apply what they learn. This can reduce their confidence in situations where their medical training will be needed. In-person classes are both expensive and inconvenient, which can discourage the general population from taking these classes. Hence, our tool will provide the accessibility of online classes with the hands-on experience of in-person classes.

To accomplish this goal, there are objectives that we need to meet. First, our tool will provide four training scenarios. These scenarios will consist of two stages: a training stage and a testing phase. The training phase will have step-by-step guides that explain exactly what the user should do. These guides will be in the form of both text on UI elements and vocal queues from characters within the game. The four scenarios will be training for CPR and resuscitation, handling head trauma, bandaging wounds and applying a tourniquet, and the Heimlich maneuver.

In the development of this training tool, we have also considered a few features we will not be implementing. We will not include a ranking system or online capabilities. This tool is intended to teach a single individual how to carry out various medical techniques, and providing a ranking system may reduce user’s confidence in real-world situations. We also will not be providing certification for the medical techniques since this requires certain standards to be met and will greatly escalate the complexity of this training tool. More information on the scope of this project is available in [Appendix A: Development Plan](#_Appendix_A:_Development).

## Definitions, Acronyms, and Abbreviations

Table 1: Definitions and Acronyms

|  |  |
| --- | --- |
| Term | Definition |
| VR | Virtual Reality |
| CPR | Cardiopulmonary Resuscitation – A technique used to save a person’s life if their heart stops, and they stop breathing |
| Heimlich Maneuver | A technique used to dislodge a piece of food or other debris from a person's throat if their airway is blocked. |
| SQL | Structured Query Language |
| Unity | A game engine that allows for VR development, utilizes the C# programming language |
| SRS | Software Requirements Specification |
| DFD | Data Flow Diagram |
| UI | User Interface |
| navel | A rounded knotty depression in the center of a person's belly caused by the detachment of the umbilical cord after birth; the umbilicus.  Alternative naming: belly button |
| FPS | Frames per second – indication of the rate at which the display refreshes. |
| XR Interaction toolkit | Component based interaction system for VR development in Unity Engine |

## References

Mayo Foundation for Medical Education and Research. (2020, November 25). *Head trauma: First aid*. Mayo Clinic. Retrieved February 10, 2023, from https://www.mayoclinic.org/first-aid/first-aid-head-trauma/basics/art-20056626

Mayo Foundation for Medical Education and Research. (2021, November 17). *Cuts and scrapes: First aid*. Mayo Clinic. Retrieved February 10, 2023, from https://www.mayoclinic.org/first-aid/first-aid-cuts/basics/art-20056711

Mayo Foundation for Medical Education and Research. (2022, February 12). *Cardiopulmonary resuscitation (CPR): First aid*. Mayo Clinic. Retrieved February 10, 2023, from https://www.mayoclinic.org/first-aid/first-aid-cpr/basics/art-20056600

Mayo Foundation for Medical Education and Research. (2022, October 11). *Choking: First aid*. Mayo Clinic. Retrieved February 10, 2023, from https://www.mayoclinic.org/first-aid/first-aid-choking/basics/art-20056637

Mayo Foundation for Medical Education and Research. (2022, October 8). *Severe bleeding: First aid*. Mayo Clinic. Retrieved February 10, 2023, from https://www.mayoclinic.org/first-aid/first-aid-severe-bleeding/basics/art-20056661

## Overview

The rest of this document is organized into three sections: general description, specific requirements, and analysis models. Following these sections, there is an appendix with supplemental information that supports the statements made in this document. The general description section is intended to discuss general factors and considerations that affect the design and requirements of our training tool. It includes an outline of the product perspective, which is how the tool looks to users, product functions, which explain the necessary functions for the implementation of the tool, user characteristics, which gives a physical description of our intended users, general constraints, and assumptions and dependencies. The specific requirements section outlines all the requirements for our tool, which are broken down into functional and non-functional requirements. It also discusses interface requirements, design constraints, database requirements, and any additional requirements that don’t fit in the other sections. Finally, the analysis models section contains the data flow diagram, which shows how user data flows through the training tool.

# General Description

This section will provide a description of the general factors and considerations that affect the design and requirements of our training tool. The product perspective gives a description of what the tool will look like to our users. The product functions section provides a summary of the functions of our application. User characteristics gives a description of who our intended users are. General constraints describe the limitations we face as developers that will affect the features of the tool. Finally, assumptions and dependencies will discuss factors that affect the requirements stated in this document.

## Product Perspective

Our software should appear as a high-quality medical training tool that can provide hands-on experience with various medical techniques to our users. The key features included in our application are:

* Main Menu – a scene giving information about our training tool and links to the other scenes, which users will interact with to begin a training scenario.
* Medical Scenarios – scenarios that will teach users each of the four medical techniques (CPR, Heimlich maneuver, head trauma, and bandaging wounds).
  + Users will select between small wounds and large wounds.
  + After selection, there will be a training phase, which will guide the user through a scenario and explain exactly what they need to do.
  + Following the training phase, the user will be asked to handle a scenario themselves in a test phase. There will be no assistance in this phase.
* Performance Evaluation - once the user completes a training scenario, they will be evaluated on their performance.

## Product Functions

Below is an outline of the core functions our app will provide.

1. Welcome and main menu
   * Provides information about the training tool
   * Has buttons that take the user to medical training scenarios
2. Bandage Training – Small Cuts and Scrapes
   * Walks user through dealing with small cuts and scrapes
   * Provides interactable items, 3D characters, and wounds
   * Provides step-by-step guide, in the form of text and audio (vocal) cues
3. Bandage Testing – Small Cuts and Scrapes
   * Provides scenario user must do alone
   * No visual cues or step-by-step guide – user must determine what to do themselves
   * User’s performance will be evaluated
4. Bandage Training – Large Wounds and Heavy Bleeding
   * Walks user through dealing with large wounds and heavy bleeding
   * Provides interactable items, 3D characters, and wounds
   * Provides step-by-step guide, in the form of text and audio (vocal) cues
5. Bandage Testing – Large Wounds and Heavy Bleeding
   * Provides scenario user must do alone
   * No visual cues or step-by-step guide – user must determine what to do themselves
   * User’s performance will be evaluated
6. CPR Training
   * Walks user through steps for CPR usage, including when and when not to use CPR
   * Provides interactable items and 3D characters
   * Provides step-by-step guide for specific scenario
7. CPR Testing
   * Provides scenario in which CPR needs to be utilized
   * No visual cues or guide – user must determine what to do alone
   * User’s performance will be evaluated
8. Head Trauma Training
   * Walks user through steps for handling head trauma
   * Provides interactable items and 3D characters
   * Provides step-by-step guide for specific scenario
9. Head Trauma Testing
   * Provides scenario in which head trauma training must be performed
   * No visual cues or guide – user must determine what to do alone
   * User’s performance will be evaluated
10. Heimlich Maneuver Training
    * Walks user through steps for carrying out the Heimlich maneuver, including when and when not to use it
    * Provides interactable items and 3D characters
    * Provides step-by-step guide for specific scenario
    * 3 scenarios – infant, adult and self
11. Heimlich Maneuver Testing
    * Provides scenario in which the Heimlich maneuver needs to be utilized
    * No visual cues or guide – user must determine what to do alone
    * User’s performance will be evaluated

A more in-depth description of the training and testing scenarios will be outlined in the specific requirements section of this document.

## User Characteristics

Our users consist of all people ages 16 and up. The target user should be familiar with the English language. We intend to cater to an audience that has no knowledge of medical techniques and no medical background. Our users should not be looking for certification for the use of these medical techniques. Possible users also could include teachers of medical training classes who are looking for a cheaper way to provide students with hands-on experience. Users should be physically capable of carrying out the given tasks, which may require moving their hands quickly, kneeling, standing up, and moving their hands. Users should also have a basic understanding of virtual reality basics, including how to work the controllers.

## General Constraints

General constraints for this training tool are around VR development tools and VR headset design limitations. VR headsets and controllers have a limited number of buttons. Trying to implement per-finger movement with different combinations of buttons will be confusing to users, increase learning curve, and defeat a purpose of simple controls that are appropriate for VR. There is also no precise implementation of hand tracking, which means there is a limit to the number of fine-motor movements that can be simulated in the scenarios.

We also need to choose a development platform that is supported in the target VR headsets. That platform should be mature enough, and is still being maintained to eliminate potential problems that could affect and complicate the development process.

## Assumptions and Dependencies

For the duration of the project XR Interaction Toolkit version utilized will remain stable and there will be no breaking changes introduced to it during the life cycle of the project that will affect our development.

All equipment required for development, testing, and demos will be available and in working condition throughout the project’s life cycle.

XR Interaction Toolkit will have sufficient functionality available to support implementation of requirements described in this document, primarily for controller velocity and positioning.

The statements offered in this document depend heavily on the timeframe of this project. The scope and specifications of the project will not change after the project has started, and the project will be in working condition by the beginning of April 2023. However, with the hard project deadline, certain requirements may be removed, and the priority of requirements may change. As the due date approaches, we will adjust our priorities to ensure this medical training tool meets the minimum requirements to meet the needs of the user.

All people involved in the project will remain for the duration of the project to ensure the integrity and steady development pace.

All team members possess sufficient skills for the project. However, work quality may differ based on the difference in experience and training. Also, team members might not be in their best condition throughout the duration of project, which might lead to performance gaps and violate the assumption of efficient work.

# Specific Requirements

In this section, we outline the requirements for our medical training tool. These requirements will ensure our tool meets the goals and objectives discussed in the introduction of this SRS document. It is broken up into four sections, including external interface requirements, functional requirements, non-functional requirements, design constraints, logical database requirements, and other requirements. Descriptions of these sections are located at the beginning of each section.

For each of the functional and non-functional requirements, we have provided priority levels that organize the requirements. These priorities were determined based on our own judgement as a team. To organize the selection process, we used a priority matrix, where we each placed our votes for priorities for each requirement. Then, we averaged our votes and rounded to the nearest number. Ties were rounded up. This matrix is provided in [Appendix B: Priority Matrix](#_Appendix_B:_Priority).

We decided on a three-level priority system. These levels are high, medium, and low. High priority requirements will be implemented first and represent the core functionality of our medical training tool. Without the implementation of these requirements, our tool would not meet the needs of our users. Medium level requirements are things that are necessary for the final project, but their implementation depends on the high priority requirements. This means that the high priority requirements must be implemented before working on these. Finally, low priority requirements are things that we will implement last. These are requirements that will be nice to have and will improve the quality of use for the user but may not be essential for meeting the needs of the user.

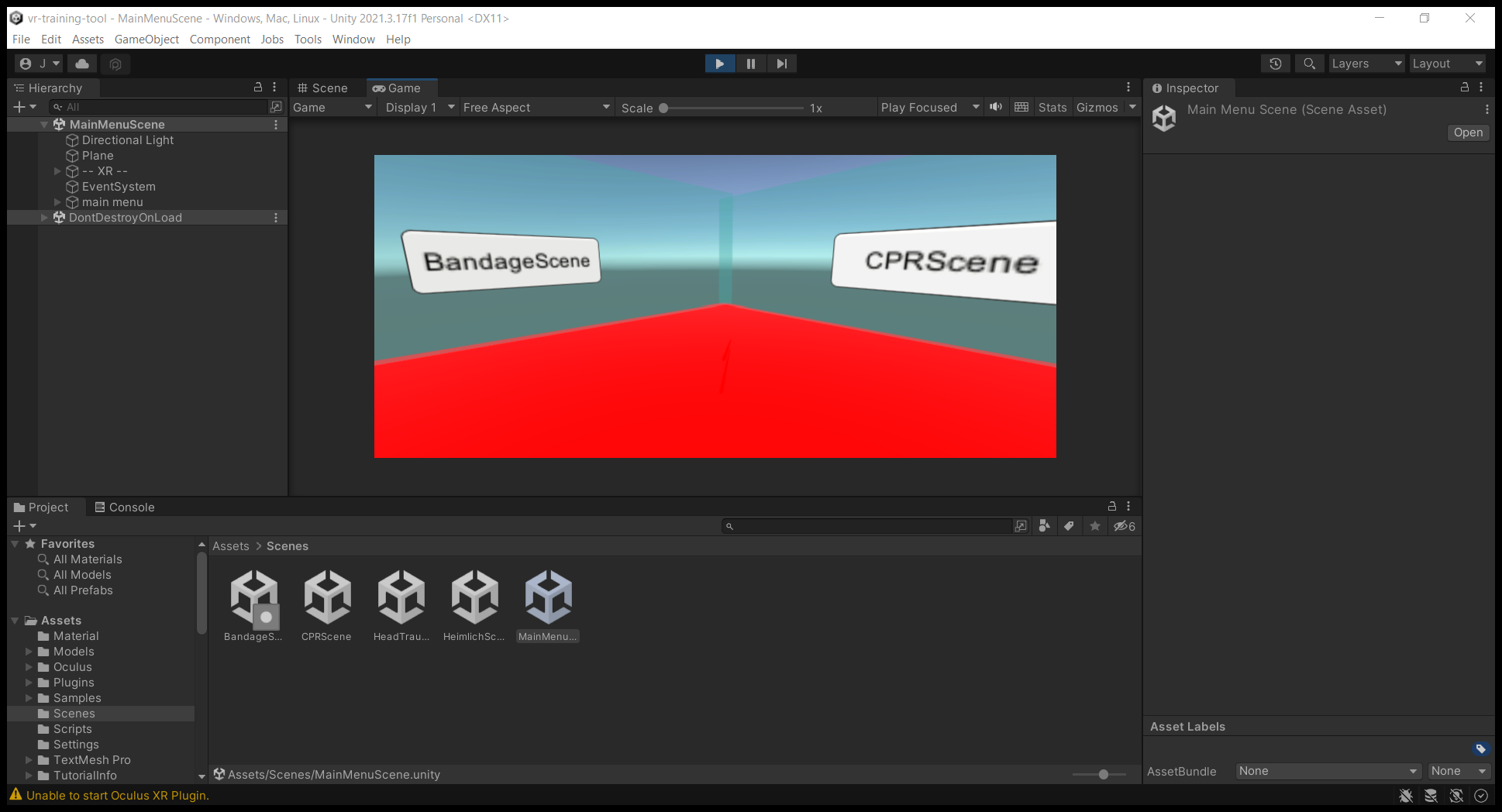
## External Interface Requirements

These requirements outline how our tool will interface with external systems. The main external system we will interact with is our users, which will be discussed in the user interface section. We also discuss the hardware interactions with the controllers for the VR headsets, software interfaces with libraries within the project, and communication interfaces.

### User Interfaces

User interaction will be handled through both user interface elements and interactable objects in the VR settings. User interfaces will contain text explaining scenarios, cues for interactions (such as grabbing an object), and buttons for navigating between scenes. Interactable objects will be highlighted with a different color ray indicator while navigating the tool, as well as user interface cues.

When the user first enters our application, they will be placed in a scene containing the main menu. This scene will contain a navigation menu, with buttons corresponding to the bandage, CPR, head trauma, and Heimlich maneuver scenes. Pressing these buttons will allow users to navigate to the corresponding scenarios.



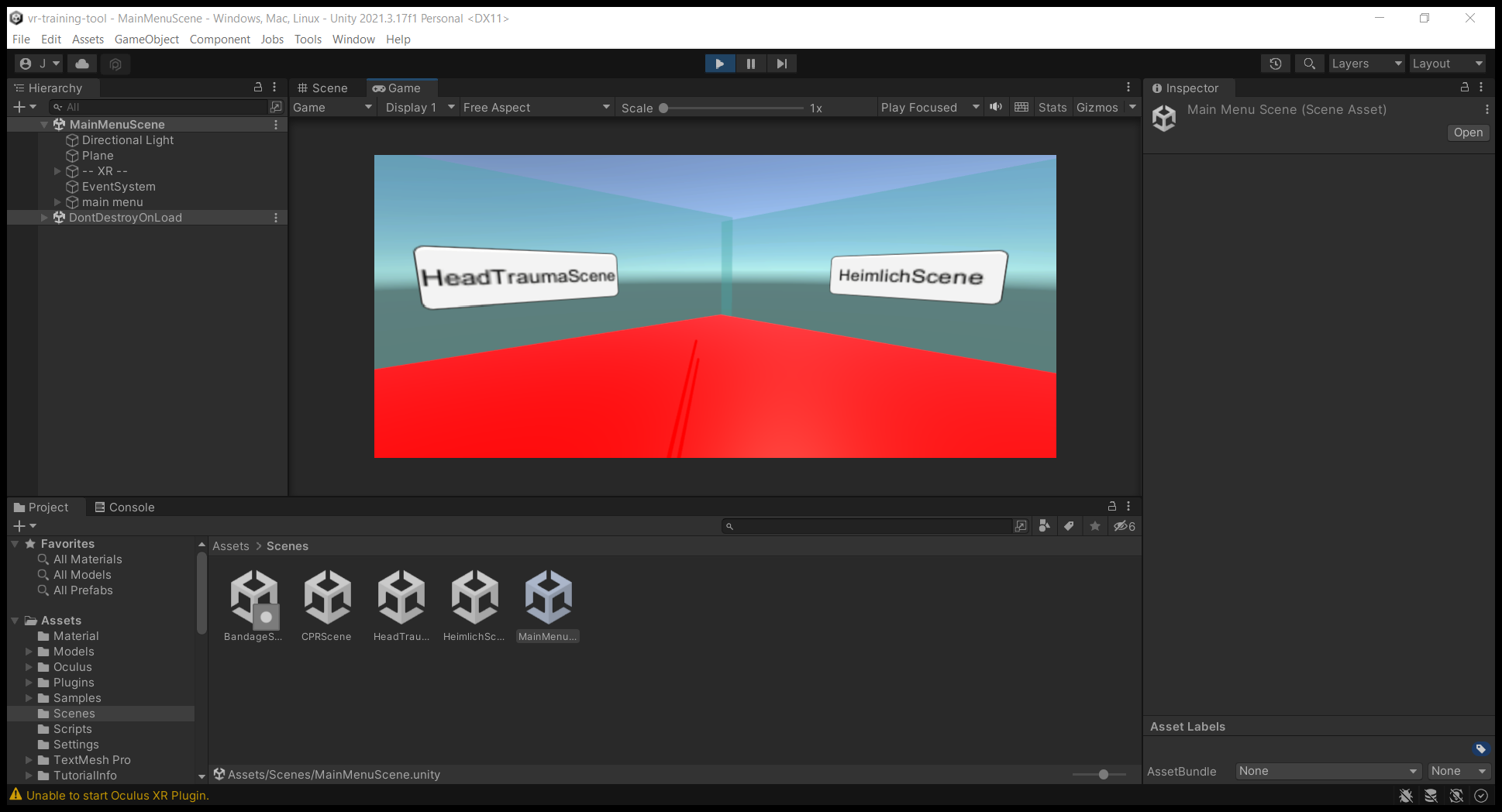


Figure 1: Main Menu User Interface with Selectable Scenarios

When the user navigates to a scenario, there will be information posted to a UI corresponding to the scenario. This information will explain the scenario and allow selection between the training scenario and testing scenario if it is unlocked. The bandage scene will have one extra user interface, which will ask the user to select between the small cuts and scrapes and large wounds scenarios, which will each have sub-menus for training and testing scenarios.

Text

Description automatically generated

Figure 2: Bandage Scene UI for Information and Sub-Scenario Selection

Once users enter a training or testing scenario, user interaction will be tracked based on interactions with objects rather than specific user interface elements. These interactions include grabbing objects, dropping objects, moving the controllers to a specific position, or moving controllers at a specific velocity. These interactions are handled by the XR Interaction Toolkit in Unity and will trigger events to occur within the scenario.

Graphical user interface, text, application

Description automatically generated

Figure 3: Outline of Steps for Small Cuts and Scrapes Scenario

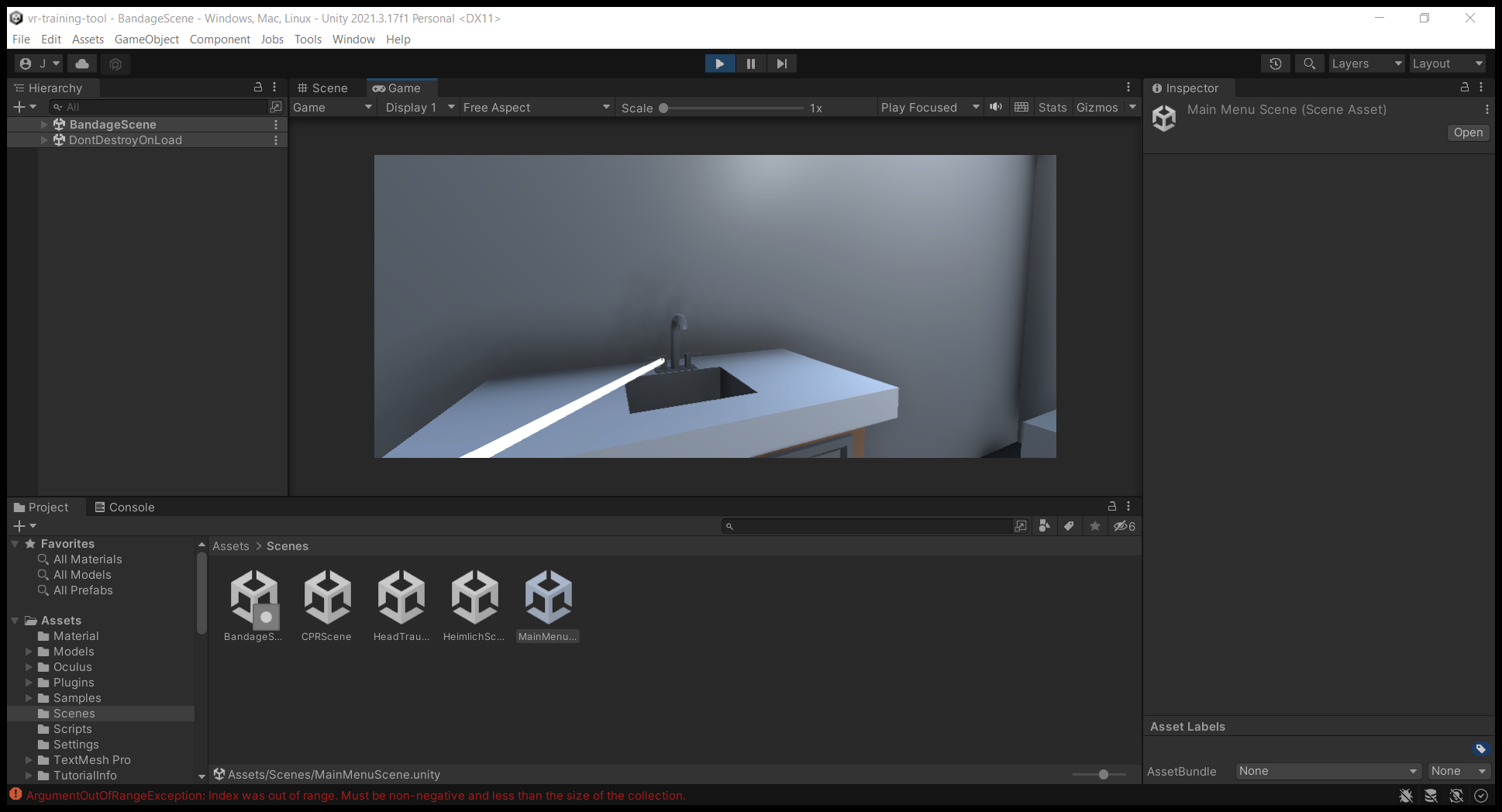


Figure 4: Interactable Sink Handle in Small Cuts and Scrapes Scenario

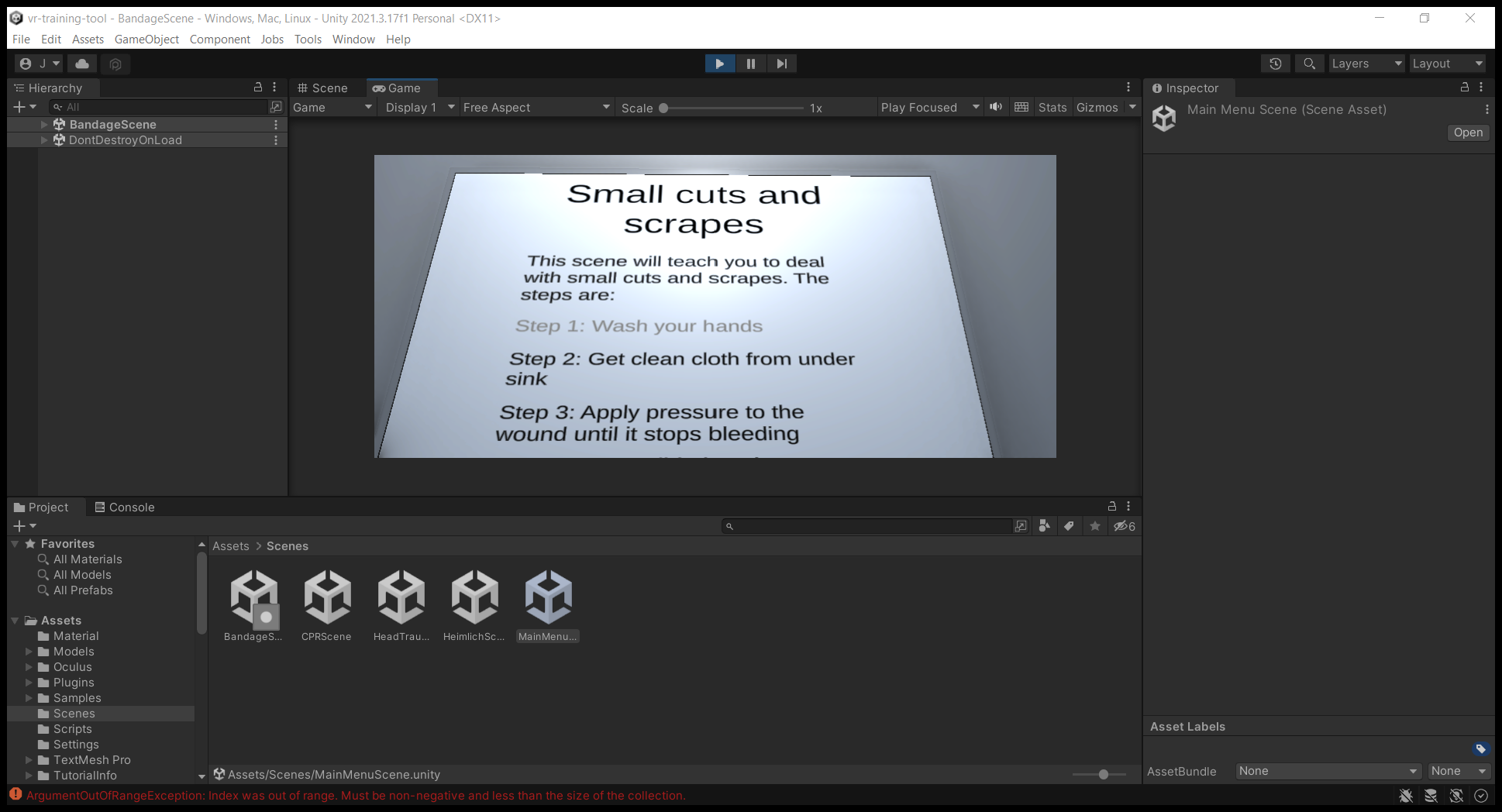


Figure 5: UI is Updated After Object Interaction

### Hardware Interfaces

Our tool will interface extensively with the VR headset and controllers. In order for the player to navigate the scene, interact with objects, move their virtual character, and carry out the hands-on portion of the training, their inputs to the controller will need to be captured and passed to our software. We will handle this interfacing through the XR Interaction Toolkit, which is offered in Unity by default. This toolkit handles the user’s inputs to controllers, including grab interactions, click (or select) interactions, controller location, and controller velocity. All these inputs will be utilized throughout the scenarios in our training tool.

### Software Interfaces

To handle VR interaction, we are utilizing the XR Interaction Toolkit, which ships with Unity. This greatly reduces the complexity in implementation with the VR controller movement tracking and button presses. Features that require specific controller positions and movement patterns will use this package’s functionalities heavily.

### Communication Interfaces

The only communication interface our software will utilize is communication between the controllers and our software. This interface is, again, provided by the XR Interaction Toolkit, so we will not need to implement any of our own communication interfaces.

## Functional Requirements

Functional requirements outline the features that must be implemented in our medical training tool to successfully carry out our goal. They describe how our medical training tool will function under certain conditions, and how it will respond to certain user interactions.

### Functional Requirement 1

|  |  |  |
| --- | --- | --- |
| **ID:** FR1 | **Title:** Scenario selection menu | |
| Upon loading the main menu, a menu should be displayed with button options for each scenario, including bandage training, CPR training, head trauma training, and Heimlich maneuver training. The user should be able to select these different button options to teleport to the corresponding scene. | | |
| **Inputs:** User selects one of the buttons on the main menu selection screen | | |
| **Processing:** Determine which button the user pressed. Upon pressing the button, the software should relocate the user to the corresponding scene. | | |
| **Outputs:** User will be teleported into the scene they selected and should be able to begin interacting with the new scene. | | |
| **Error Handling:** If the scene fails to load, the user should be teleported back to the main menu scene. | | |
| **Priority:** HIGH | **Backward Dependencies:** None | **Forward Dependencies:** All other requirements |

### Functional Requirement 2

|  |  |  |
| --- | --- | --- |
| **ID:** FR2 | **Title:** Bandage scenario: sub-scenario selection | |
| After the user selects the bandage scene from the scenario selection menu and the user is teleported into the bandage scene, there should be another menu available to the user. This menu should contain options for selecting between the small cuts and scrapes sub-scenario and the large wounds with heavy bleeding sub-scenario. Upon selecting one of these options, the software should allow the user to select between training and testing, as outlined in FR3. | | |
| **Inputs:** User should select either small cuts and scrapes or large wounds with heavy bleeding. | | |
| **Processing:** Determine which button the user pressed. Upon pressing the button, the software should relocate the user to the corresponding scene and display a selection for either training or testing, as outlined in FR3. | | |
| **Outputs:** Software should display new menu that allows user to select between training and testing, as outlined in FR3. | | |
| **Error Handling:** If the scene fails to load, the user should be teleported back to the main menu scene. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR1 | **Forward Dependencies:** FR3 |

### Functional Requirement 3

|  |  |  |
| --- | --- | --- |
| **ID:** FR3 | **Title:** Training versus testing option | |
| After the user selects either the CPR, head trauma, Heimlich maneuver, small cuts and scrapes, or large wounds with heavy bleeding scenarios, there should be a UI menu for the user to select between training and testing. When the user first loads one of the aforementioned scenarios, the testing option should be locked by default. However, after the user completes the training scenario, the testing option should be unlocked. Upon selecting one of the options, the scenario should begin. | | |
| **Inputs:** User should select between either training or testing. If they haven’t completed the training scenario yet, they should only be able to select the training option. | | |
| **Processing:** Determine which button the user pressed. Upon pressing the button, the software should trigger the beginning of either the training or testing scenario according to which button was pressed. | | |
| **Outputs:** User should be placed in either the training or testing scenario, depending on which button was pressed. | | |
| **Error Handling:** If the user selects the testing option before they have completed the training scenario, the software should not let them begin the testing scenario. There should be a message that appears on the UI that tells the user that they must first complete the training scenario in order to unlock the testing scenario. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR1, FR2 | **Forward Dependencies:** FR10, FR14, FR21, FR25, FR29, FR33, FR37-FR41 |

### Functional Requirement 4

|  |  |  |
| --- | --- | --- |
| **ID:** FR4 | **Title:** Progress bar | |
| In the main menu scene, a progress bar should be displayed for returning users. The bar will be updated as the user completes the different scenarios. | | |
| **Inputs:** Each time a training or testing scenario is completed, its completion status should be passed to the progress bar. | | |
| **Processing:** As the user completes scenarios, the progress bar will track these completions as a percentage of the total number of scenarios (10). | | |
| **Outputs:** Both the percentage of scenarios completed and a physical progress bar with the corresponding percentage filled in should be visible to the user. | | |
| **Error Handling:** None | | |
| **Priority:** LOW | **Backward Dependencies:** FR1, FR5, FR14, FR21, FR25, FR29, FR33, FR37-FR41, FR44 | **Forward Dependencies:** None |

### Functional Requirement 5

|  |  |  |
| --- | --- | --- |
| **ID:** FR5 | **Title:** Local progress storage | |
| Upon the completion of each training and testing scenario, a local database should be updated to store the completion of the given scenario. When the user returns to the software after closing it, this progress should be restored, and the progress bar from FR4 should be updated to reflect the user’s progress. In addition, within each training scenario, progress should be saved upon completion of each step of training. During testing scenarios, it is unnecessary to store progress since these should be completed in one session. | | |
| **Inputs:** When a user completes a scenario, its completion status should be passed to local storage. Also, when the user completes each step of a training scenario, it’s completion should also be passed to local storage. | | |
| **Processing:** Completion status should be stored in a local database. Upon reloading the app, the completion status should be loaded, the progress bar (FR4) should be updated, and testing scenarios should be unlocked if the corresponding training scenario has been completed (FR3). | | |
| **Outputs:** Data will be stored in database. Upon reloading, progress bar should be updated, and testing scenarios should be unlocked if the corresponding training scenario has been completed. | | |
| **Error Handling:** If the user exits software in the middle of a training scenario, their progress within that scenario should be saved and reloaded when they re-enter the software. However, if the user exits during a testing scenario, the testing scenario status should remain incomplete, and the user must restart from the beginning. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR1, FR14 -FR42 | **Forward Dependencies:** FR4, FR14-FR42 |

### Functional Requirement 6

|  |  |  |
| --- | --- | --- |
| **ID:** FR6 | **Title:** Training scenario success banner | |
| Upon the completion of each training scenario, a UI banner should appear. This banner should congratulate the user for completing the training scenario, inform them that the testing scenario has been unlocked, and contain a button that allows the user to navigate back to the training/testing menu (FR3). | | |
| **Inputs:** When a user completes a scenario, it’s completion status should trigger a banner to appear. Then, the user should be able to select the button on the UI that returns them to the training/testing selection menu. | | |
| **Processing:** Completion status should update the training/testing menu to unlock the testing scenario. This completion status should also be sent to local storage (FR5) and to the progress bar (FR4). | | |
| **Outputs:** Corresponding testing scenario should be unlocked, local storage should be updated, progress bar should be updated, user should be moved to the training/testing selection menu. | | |
| **Error Handling:** None | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR14 -FR36 | **Forward Dependencies:** None |

### Functional Requirement 7

|  |  |  |
| --- | --- | --- |
| **ID:** FR7 | **Title:** Training scenarios: UI tracking the steps to complete scenario | |
| For each of the training scenarios, there should be a UI that outlines all of the steps required to complete the scenario. This should be loaded as soon as the user enters the scenario. As the user completes the steps for the training scenario, this UI should be updated by crossing off or otherwise indicating that the step has been completed. | | |
| **Inputs:** Completion status of each step of the training scenario | | |
| **Processing:** Update UI based on which steps have been completed: if a step is completed, it should be crossed off, if it has not been completed, it should not be crossed off. | | |
| **Outputs:** UI should indicate which steps the user has completed and what step they are currently on. | | |
| **Error Handling:** None | | |
| **Priority:** HIGH | **Backward Dependencies:** None | **Forward Dependencies:** FR14 -FR36 |

### Functional Requirement 8

|  |  |  |
| --- | --- | --- |
| **ID:** FR8 | **Title:** Training scenarios: UI interaction indicators | |
| For each step of each training scenario, there should be UI interaction indicators near the item that is to be interacted with for that given step. This UI should only appear near one item, and it should tell the user what interaction to carry out (grab, select, etc.) to complete the given step. The items and their corresponding interactions are outlined in requirements that follow. | | |
| **Inputs:** Current step the user is on in a given training scenario | | |
| **Processing:** Load the UI indicator corresponding to the step the user is currently on in the training scenario. | | |
| **Outputs:** UI should be displayed to the user near the interactable object indicating how to interact with the object to complete the step. | | |
| **Error Handling:** If the user attempts to complete the steps out of order, the UI should not be affected. The UI should remain visible until the user completes the step correctly. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** None | **Forward Dependencies:** FR14 -FR36 |

### Functional Requirement 9

|  |  |  |
| --- | --- | --- |
| **ID:** FR9 | **Title:** Training scenarios: audio indicators | |
| For each step of each training scenario, there should be an audio clip that indicates what the user is expected to do for the current step. This audio clip should play whenever the user completes the previous step. | | |
| **Inputs:** Completion of the previous step of a training scenario, or the loading of the scenario itself. | | |
| **Processing:** Load and play the audio indicator corresponding to the step the user is currently on in the training scenario. | | |
| **Outputs:** The audio clip should play to the user. | | |
| **Error Handling:** If the user completes a step out of order or interacts with the wrong object in the scene, another audio clip should play telling them that they interacted with the wrong object, and the original audio clip should replay to remind them of the step they are currently on. | | |
| **Priority:** LOW | **Backward Dependencies:** None | **Forward Dependencies:** FR14 -FR36 |

### Functional Requirement 10

|  |  |  |
| --- | --- | --- |
| **ID:** FR10 | **Title:** Small cuts and scrapes scenario: wash hands step | |
| After the user selects the small cuts and scrapes scenario (FR2) and selects the training scenario, they should be prompted to complete the first step of the training scenario. This step will be washing their hands. To do this, they must interact with a handle on a sink. They will use a “grab” interaction on their controller. This interaction should update the steps UI (FR7) and trigger the next step. Prior to the user interacting with the sink handle, the corresponding UI indicator should be displayed, and an audio clip should be played telling the user what to do. | | |
| **Inputs:** User should grab the handle on the sink. | | |
| **Processing:** Once the user grabs the handle, the UI outlining the steps (FR7) should be updated to indicate they completed the step. The UI indicator for the next step should appear (FR8), and the audio clip for the next step should play (FR9). The completion status should also be sent to local storage (FR5). | | |
| **Outputs:** UI tracking the steps the user has completed should be updated, the audio clip for the next step should be triggered to play, the UI indicator for the next step should appear, and the progress data should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9 | **Forward Dependencies:** FR11 |

### Functional Requirement 11

|  |  |  |
| --- | --- | --- |
| **ID:** FR11 | **Title:** Small cuts and scrapes scenario: apply pressure step | |
| After the user successfully washes their hands, they should continue by applying pressure to the wound. This step should involve three user interactions, each of which should contain a UI indicator (FR8) and audio queue (FR9). The first interaction should be to open the door under to sink. This will be a simple “select” interaction on the door handle. Next, the user should grab the bandage from under the sink, which will be captured by a grab interaction. Finally, the user should apply pressure to the wound with the bandage. This will be captured by a socket interaction on the wound, which when the user drops the bandage on the wound, it will automatically snap into the correct place to cover the wound. | | |
| **Inputs:** Completion of the previous step, interaction with the door handle, grab interaction on bandage, drop interaction on socket near wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to open the door, and the corresponding audio clip should play. Once they open the door, the UI indicator for grabbing the bandage should appear, and the audio clip should play. Finally, when they grab the bandage, the UI above the patients wound should appear telling them to drop the bandage on the wound, and the audio clip should play for that. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near door, bandage, and wound, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR10 | **Forward Dependencies:** FR12 |

### Functional Requirement 12

|  |  |  |
| --- | --- | --- |
| **ID:** FR12 | **Title:** Small cuts and scrapes scenario: antibiotic cleaning step | |
| After the user successfully applies pressure to the wound, they should continue by applying an antibiotic to the wound. This step should involve two user interactions, each of which should contain a UI indicator (FR8) and audio queue (FR9). The first interaction should be to grab the antibiotic substance from the sink. This will be a “grab” interaction on the antibiotic bottle. Next, the user should drop the antibiotic substance near the wound. This interaction will be captured with a socket interaction, which will trigger the antibiotic to be applied to the correct location. Upon completion of these interactions, the steps UI should be updated to indicate the completion and completion should be stored locally. | | |
| **Inputs:** Completion of the previous step, user grab interaction with the antibiotic bottle, user drop interaction on the socket near the wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to grab the antibiotic bottle, and the corresponding audio clip should play. Once they grab the antibiotic, the UI indicator for applying the antibiotic to the wound should appear, and the audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators antibiotic bottle and wound, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR11 | **Forward Dependencies:** FR13 |

### Functional Requirement 13

|  |  |  |
| --- | --- | --- |
| **ID:** FR13 | **Title:** Small cuts and scrapes scenario: cover wound with bandage | |
| After the user successfully applies the antibiotic, they should continue by applying pressure to the wound. This step should involve two user interactions, each of which should contain a UI indicator (FR8) and audio queue (FR9). The first interaction should be to grab the bandage from wherever they left it after its previous use. This will be a simple “grab” interaction on the bandage. Next, the user should apply the bandage to the wound. This will be captured by a socket interaction on the wound, which when the user drops the bandage on the wound, it will automatically snap into the correct place to cover the wound. | | |
| **Inputs:** Completion of the previous step, user grab interaction with the bandage, user drop interaction on the socket near the wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to grab the bandage, and the corresponding audio clip should play. Once they grab the bandage, the UI indicator for applying the bandage to the wound should appear, and the audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near bandage, and wound, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR12 | **Forward Dependencies:** FR6 |

### Functional Requirement 14

|  |  |  |
| --- | --- | --- |
| **ID:** FR14 | **Title:** Large wounds with heavy bleeding: call 911 | |
| Once the user selects the training scenario for large wounds with heavy bleeding, the first step should be initialized. This step is for calling emergency services. To accomplish this, they should interact with the telephone on the wall. This will be a simple grab interaction. When they are finished, they can hang up the phone again, or just drop it. This interaction should be prompted by both an interaction UI and an audio clip queue. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Loading of the large wounds with heavy bleeding scenario, phone grab interaction. | | |
| **Processing:** When the user loads the large wounds with heavy bleeding scenario, the UI indicator for the grab interaction will appear near the phone. Once the user grabs the phone, the UI indicator for the next step should appear, the audio clip for the next step should play, the steps UI should be updated to indicate the completion of the step, and the progress should be stored locally. | | |
| **Outputs:** UI indicator near phone, as well as corresponding audio clip. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9 | **Forward Dependencies:** FR15 |

### Functional Requirement 15

|  |  |  |
| --- | --- | --- |
| **ID:** FR15 | **Title:** Large wounds with heavy bleeding: remove debris | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to remove debris from around the wound. This will be accomplished through a simple interactable near the wound. Upon completing this step, the steps UI should be updated to reflect the progress, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, user interaction with simple interactable near wound. | | |
| **Processing:** Upon completion of the previous step, the interaction indicator UI should appear near the wound and the corresponding audio clip should play. Once the user interacts with the wound, the completion of the step should be indicated on the steps UI, and the progress should be stored locally. | | |
| **Outputs:** UI indicator near wound, as well as corresponding audio clip. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR14 | **Forward Dependencies:** FR16 |

### Functional Requirement 16

|  |  |  |
| --- | --- | --- |
| **ID:** FR16 | **Title:** Large wounds with heavy bleeding: stop bleeding | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to stop or slow bleeding. This will be accomplished through three interactions. The first interaction should be to open the door under to sink. This will be a simple “select” interaction on the door handle. Next, the user should grab the bandage from under the sink, which will be captured by a grab interaction. Finally, the user should apply pressure to the wound with the bandage. This will be captured by a socket interaction on the wound, which when the user drops the bandage on the wound, it will automatically snap into the correct place to cover the wound. | | |
| **Inputs:** Completion of the previous step, interaction with the door handle, grab interaction on bandage, drop interaction on socket near wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to open the door, and the corresponding audio clip should play. Once they open the door, the UI indicator for grabbing the bandage should appear, and the audio clip should play. Finally, when they grab the bandage, the UI above the patients wound should appear telling them to drop the bandage on the wound, and the audio clip should play for that. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near door, bandage, and wound, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR15 | **Forward Dependencies:** FR17 |

### Functional Requirement 17

|  |  |  |
| --- | --- | --- |
| **ID:** FR17 | **Title:** Large wounds with heavy bleeding: help patient lie down | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to help the patient lie down. This will be accomplished through a simple interactable on the patient, which should be accompanied by a UI interaction indicator and an audio clip. Once the user interacts, the steps UI should be updated, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, interaction with the simple interactable near patient. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to interact with the patient., and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR16 | **Forward Dependencies:** FR18 |

### Functional Requirement 18

|  |  |  |
| --- | --- | --- |
| **ID:** FR18 | **Title:** Large wounds with heavy bleeding: apply more bandages | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to apply more bandages to slow bleeding. This will be accomplished through a grab interactable on the bandage, as well as a socket on the wound. The user should grab the bandage and drop it on the socket near the wound to apply the bandage. Each interaction should be accompanied by a UI indicator and audio clip. Once the user completes the interactions, the steps UI should be updated, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, grab interaction with bandage, socket interaction on wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to grab the bandage, and the corresponding audio clip should play. After they grab the bandage, a UI indicator should tell them to drop it on the socket near the wound, and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR17 | **Forward Dependencies:** FR19 |

### Functional Requirement 19

|  |  |  |
| --- | --- | --- |
| **ID:** FR19 | **Title:** Large wounds with heavy bleeding: apply tourniquet | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to apply a tourniquet if necessary. This will be accomplished through a grab interactable on the tourniquet, as well as a socket on the wound. The user should grab the tourniquet and drop it on the socket near the wound to apply the tourniquet. Each interaction should be accompanied by a UI indicator and audio clip. Once the user completes the interactions, the steps UI should be updated, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, grab interaction with tourniquet, socket interaction on wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to grab the tourniquet, and the corresponding audio clip should play. After they grab the tourniquet, a UI indicator should tell them to drop it on the socket near the wound, and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR18 | **Forward Dependencies:** FR20 |

### Functional Requirement 20

|  |  |  |
| --- | --- | --- |
| **ID:** FR20 | **Title:** Large wounds with heavy bleeding: wash hands | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to wash their hands once help arrives. To do this, they must interact with a handle on a sink. They will use a “grab” interaction on their controller. This interaction should update the steps UI and trigger the next step. Prior to the user interacting with the sink handle, the corresponding UI indicator should be displayed, and an audio clip should be played telling the user what to do. | | |
| **Inputs:** Completion of the previous step, simple interaction with sink handle | | |
| **Processing:** When the user completes the previous step, the UI indicator should appear telling the user to interact with the sink handle, and the corresponding audio clip should play. Upon completion of the interaction, the steps UI should be updated to reflect the progress, and progress should be stored locally. | | |
| **Outputs:** UI indicator near sink handle, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR2, FR3, FR5, FR7, FR8, FR9, FR19 | **Forward Dependencies:** FR6 |

### Functional Requirement 21

|  |  |  |
| --- | --- | --- |
| **ID:** FR21 | **Title:** CPR: call 911 | |
| Once the user selects the training scenario for CPR, the first step should be initialized. This step is for calling emergency services. To accomplish this, they should interact with the telephone on the wall. This will be a simple grab interaction. When they are finished, they can hang up the phone again, or just drop it. This interaction should be prompted by both an interaction UI and an audio clip queue. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Loading of the large wounds with heavy bleeding scenario, phone grab interaction. | | |
| **Processing:** When the user loads the large wounds with heavy bleeding scenario, the UI indicator for the grab interaction will appear near the phone. Once the user grabs the phone, the UI indicator for the next step should appear, the audio clip for the next step should play, the steps UI should be updated to indicate the completion of the step, and the progress should be stored locally. | | |
| **Outputs:** UI indicator near phone, as well as corresponding audio clip. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9 | **Forward Dependencies:** FR22 |

### Functional Requirement 22

|  |  |  |
| --- | --- | --- |
| **ID:** FR22 | **Title:** CPR: call 911 | |
| Once the user selects the training scenario for CPR, the first step should be initialized. This step is for calling emergency services. To accomplish this, they should interact with the telephone on the wall. This will be a simple grab interaction. When they are finished, they can hang up the phone again, or just drop it. This interaction should be prompted by both an interaction UI and an audio clip queue. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Loading of the large wounds with heavy bleeding scenario, phone grab interaction. | | |
| **Processing:** When the user loads the large wounds with heavy bleeding scenario, the UI indicator for the grab interaction will appear near the phone. Once the user grabs the phone, the UI indicator for the next step should appear, the audio clip for the next step should play, the steps UI should be updated to indicate the completion of the step, and the progress should be stored locally. | | |
| **Outputs:** UI indicator near phone, as well as corresponding audio clip. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR21 | **Forward Dependencies:** FR23 |

### Functional Requirement 23

|  |  |  |
| --- | --- | --- |
| **ID:** FR23 | **Title:** CPR: place patient on their back on a hard surface | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to place the patient on their back on a hard surface. This will be accomplished through a simple interactable on the patient, which should be accompanied by a UI interaction indicator and an audio clip. Once the user interacts, the steps UI should be updated, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, interaction with the simple interactable near patient. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to interact with the patient, and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR22 | **Forward Dependencies:** FR24 |

### Functional Requirement 24

|  |  |  |
| --- | --- | --- |
| **ID:** FR24 | **Title:** CPR: kneel, place hands on chest | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to kneel next to the patient and place their hands on the patient’s chest between the nipples. This will be accomplished through a script that determines if the user’s location is within a certain region near the patient, as well as a script that monitors the location of the controllers. There will be UI indicators and audio indicators that guide the user through these positioning steps. | | |
| **Inputs:** Completion of the previous step, location of the user, location of the controllers. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to position themselves correctly, and the corresponding audio clip should play. Scripts should monitor the location of the user and their controllers, and the completion of this step should only be triggered when both are within the acceptable regions. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR23 | **Forward Dependencies:** FR25 |

### Functional Requirement 25

|  |  |  |
| --- | --- | --- |
| **ID:** FR25 | **Title:** CPR: press straight down at a rate of 100-120 beats per minute | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to press straight down to a depth of two inches at a rate of 100-120 beats per minute. This will be accomplished through a script that monitors the movement of the controllers. There will be UI indicators and audio indicators that guide the user through these steps. | | |
| **Inputs:** Completion of the previous step, location of the controllers. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to begin compressions, and the corresponding audio clip should play. Scripts should monitor the user’s controllers and provide feedback based on both the depth of the compression (too shallow or too deep), as well as the rate at which they are compressing (too fast or too slow). Completion of this step will only occur after help arrives. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR24 | **Forward Dependencies:** FR6 |

### Functional Requirement 26

|  |  |  |
| --- | --- | --- |
| **ID:** FR26 | **Title:** Head trauma: call 911 | |
| Once the user selects the training scenario for head trauma, the first step should be initialized. This step is for calling emergency services. To accomplish this, they should interact with the telephone on the wall. This will be a simple grab interaction. When they are finished, they can hang up the phone again, or just drop it. This interaction should be prompted by both an interaction UI and an audio clip queue. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Loading of the large wounds with heavy bleeding scenario, phone grab interaction. | | |
| **Processing:** When the user loads the large wounds with heavy bleeding scenario, the UI indicator for the grab interaction will appear near the phone. Once the user grabs the phone, the UI indicator for the next step should appear, the audio clip for the next step should play, the steps UI should be updated to indicate the completion of the step, and the progress should be stored locally. | | |
| **Outputs:** UI indicator near phone, as well as corresponding audio clip. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9 | **Forward Dependencies:** FR27 |

### Functional Requirement 27

|  |  |  |
| --- | --- | --- |
| **ID:** FR27 | **Title:** Head trauma: keep person still | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to keep the patient still. This will be accomplished through a simple interactable on the patient, which should be accompanied by a UI interaction indicator and an audio clip. Once the user interacts, the steps UI should be updated, and progress should be stored locally. | | |
| **Inputs:** Completion of the previous step, interaction with the simple interactable near patient. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to interact with the patient, and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicator near patient, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR26 | **Forward Dependencies:** FR28 |

### Functional Requirement 28

|  |  |  |
| --- | --- | --- |
| **ID:** FR28 | **Title:** Head trauma: apply pressure step | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to apply pressure to open wounds. This step should involve three user interactions, each of which should contain a UI indicator and audio queue. The first interaction should be to open the door under to sink. This will be a simple “select” interaction on the door handle. Next, the user should grab the bandage from under the sink, which will be captured by a grab interaction. Finally, the user should apply pressure to the wound with the bandage. This will be captured by a socket interaction on the wound, which when the user drops the bandage on the wound, it will automatically snap into the correct place to cover the wound. | | |
| **Inputs:** Completion of the previous step, interaction with the door handle, grab interaction on bandage, drop interaction on socket near wound. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to open the door, and the corresponding audio clip should play. Once they open the door, the UI indicator for grabbing the bandage should appear, and the audio clip should play. Finally, when they grab the bandage, the UI above the patients wound should appear telling them to drop the bandage on the wound, and the audio clip should play for that. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near door, bandage, and wound, as well as corresponding audio clips. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR27 | **Forward Dependencies:** FR29 |

### Functional Requirement 29

|  |  |  |
| --- | --- | --- |
| **ID:** FR29 | **Title:** Head trauma: monitor patient | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to monitor the patient for changes in breathing and alertness, and if there are signs of loss of circulation, to begin CPR. This step should involve interacting with the patient to help keep them alert through simple interactions until help arrives. | | |
| **Inputs:** Completion of the previous step, interaction with the patient through simple interactions. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to keep the patient alert, and the corresponding audio clip should play. When the user interacts with the patient, their alertness should be increased. Completion of this step will only occur when help arrives. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near patient, prompting user to interact to keep patient alert. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. If the user fails to interact with the interactable within a set time, they will fail the step and must restart. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR28 | **Forward Dependencies:** FR6 |

### Functional Requirement 30

|  |  |  |
| --- | --- | --- |
| **ID:** FR30 | **Title:** Heimlich maneuver adult: address situation | |
| Once the user selects the training scenario for Heimlich maneuver for adults, the first step should be initialized. This step is monitoring the situation and ensuring the person can’t cough or breathe to dislodge the debris from their throat. To accomplish this, the user will interact with a simple interactable on the patient, which will act to simulate the user monitoring the patient and ensuring they need assistance. This interaction should be prompted by both an interaction UI and an audio clip queue. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Loading of the Heimlich maneuver adult training scenario, interaction with the patient through simple interactions. | | |
| **Processing:** When the user loads the Heimlich maneuver adult training, the UI indicator should be updated to tell the user to address the situation and ensure the patient needs assistance, and the corresponding audio clip should play. After they complete this step, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near patient. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9 | **Forward Dependencies:** FR31 |

### Functional Requirement 31

|  |  |  |
| --- | --- | --- |
| **ID:** FR31 | **Title:** Heimlich maneuver adult: give five back blows | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to give the patient five back blows. This step involves monitoring the location of the user as well as the position and velocity of the controllers. Completion of the step should only occur after the user successfully navigates into the proper position behind the patient and simulates giving five back blows. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Completion of the previous step, location of the user and controllers, velocity of controllers. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to navigate to behind the patient and give five back blows, and the corresponding audio clips should play. When the user interacts with the patient, they should receive feedback on the speed of the simulated back blow. After they complete this step by giving a total of five back blows, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near patient prompting user to position themselves and give five back blows, feedback on the speed of their back blows. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR30 | **Forward Dependencies:** FR32 |

### Functional Requirement 32

|  |  |  |
| --- | --- | --- |
| **ID:** FR32 | **Title:** Heimlich maneuver adult: give five abdominal thrusts | |
| Upon completion of the prior step, the user should be prompted through indicator UI’s as well as audio clips to give the patient five abdominal thrusts. This step involves monitoring the location of the user as well as the position and velocity of the controllers. Feedback will be given about the velocity of the abdominal thrusts. Completion of the step should only occur after the user successfully navigates into the proper position behind the patient and simulates giving five abdominal thrusts. Upon completion of this step, the steps UI should be updated, and the completion should be stored locally. | | |
| **Inputs:** Completion of the previous step, location of the user and controllers, velocity of controllers. | | |
| **Processing:** When the user completes the prior step, the UI indicator should be updated to tell the user to navigate to behind the patient and give five abdominal thrusts, and the corresponding audio clips should play. When the user interacts with the patient, they should receive feedback on the speed of the simulated abdominal thrust. After they complete this step by giving a total of five abdominal thrusts, the steps UI should be updated to reflect their progress and progress should be stored locally. | | |
| **Outputs:** UI indicators near patient prompting user to position themselves and give five abdominal thrusts, feedback on the speed of their abdominal thrusts. Steps UI should be updated after the interaction and the progress should be stored locally. | | |
| **Error Handling:** If the user selects a different interactable item, there should be an audio clip that plays telling them that they interacted with the wrong object. | | |
| **Priority:** HIGH | **Backward Dependencies:** FR3, FR5, FR7, FR8, FR9, FR31 | **Forward Dependencies:** FR33 |

### Functional Requirement 33

|  |  |  |
| --- | --- | --- |
| **ID:** FR38 | **Title:** Small cuts and scrapes: testing scenario | |
| After users complete the training scenario, they will unlock the ability to select the testing scenario from the training/testing menu (FR3). Upon selecting testing from the small cuts and scrapes scenario, they will be placed in a scenario identical to the training scenario for small cuts and scrapes, but this time there will be no UI interaction indicators or audio indicators to tell them what to do. They must carry out the steps on their own. To successfully pass the testing scenario, they must carry out the following steps in the proper order with no mistakes:   1. Wash hands 2. Stop bleeding by applying pressure with a clean cloth 3. Clean wound with antibiotic solution 4. Cover the wound with a bandage   If the user interacts with an unnecessary object or performs a step out of order, it will be recorded as a mistake. The number of correct actions versus mistakes will be recorded and displayed to the user after the completion of the testing scenario. | | |
| **Inputs:** Simple interaction with sink handle, simple interaction with sink door handle, grab interaction with bandage, socket interaction from dropping bandage on wound, grab interaction with antibiotic solution, socket interaction from dropping antibiotic on wound, grab interaction with bandage again, socket interaction from dropping bandage on wound again | | |
| **Processing:** Each user interaction will be processed and compared to a ground truth order for the interactions. If they match, a correct action will be logged, and if they don’t match, an incorrect action will be logged. The number of correct and incorrect actions should be displayed to the user at the end of the testing scenario, and a completion certification should only be given if there are zero incorrect actions. | | |
| **Outputs:** The number of correct and incorrect actions should be shown to the user at the end of the testing scenario. | | |
| **Error Handling:** If the user selects an incorrect interactable item, it will be logged as a mistake and displayed at the end of the testing scenario. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR3, FR5 | **Forward Dependencies:** FR44 |

### Functional Requirement 34

|  |  |  |
| --- | --- | --- |
| **ID:** FR39 | **Title:** Large wounds with heavy bleeding: testing scenario | |
| After users complete the training scenario, they will unlock the ability to select the testing scenario from the training/testing menu (FR3). Upon selecting testing from the large wounds with heavy bleeding scenario, they will be placed in a scenario identical to the training scenario for large wounds with heavy bleeding, but this time there will be no UI interaction indicators or audio indicators to tell them what to do. They must carry out the steps on their own. To successfully pass the testing scenario, they must carry out the following steps in the proper order with no mistakes:   1. Call 911 2. Remove debris from around the wound 3. Stop bleeding by applying pressure with clean bandage 4. Help patient lie down 5. Apply more bandages to slow bleeding 6. Apply tourniquet to the limb 7. Wash hands after help arrives.   If the user interacts with an unnecessary object or performs a step out of order, it will be recorded as a mistake. The number of correct actions versus mistakes will be recorded and displayed to the user after the completion of the testing scenario. | | |
| **Inputs:** Grab interaction with phone, simple interaction with wound, simple interaction with sink door handle, grab interaction with clean bandage, socket interaction with bandage drop on wound, simple interaction for helping patient lie down, grab interaction on bandages again, socket interaction for applying new bandages, grab interaction with tourniquet, socket interaction for applying tourniquet, simple interaction with sink handle. | | |
| **Processing:** Each user interaction will be processed and compared to a ground truth order for the interactions. If they match, a correct action will be logged, and if they don’t match, an incorrect action will be logged. The number of correct and incorrect actions should be displayed to the user at the end of the testing scenario, and a completion certification should only be given if there are zero incorrect actions. | | |
| **Outputs:** The number of correct and incorrect actions should be shown to the user at the end of the testing scenario. | | |
| **Error Handling:** If the user selects an incorrect interactable item, it will be logged as a mistake and displayed at the end of the testing scenario. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR3, FR5 | **Forward Dependencies:** FR44 |

### Functional Requirement 35

|  |  |  |
| --- | --- | --- |
| **ID:** FR40 | **Title:** CPR: testing scenario | |
| After users complete the training scenario, they will unlock the ability to select the testing scenario from the training/testing menu (FR3). Upon selecting testing from the CPR scenario, they will be placed in a scenario identical to the training scenario for CPR, but this time there will be no UI interaction indicators or audio indicators to tell them what to do. They must carry out the steps on their own. To successfully pass the testing scenario, they must carry out the following steps in the proper order with no mistakes:   1. Call 911 2. Place person on their back on a hard surface 3. Kneel next to patient and position hands at the center of the patient’s chest 4. Perform compressions to a depth of 2 inches at a rate of 100-120 beats per minute until help arrives   If the user interacts with an unnecessary object or performs a step out of order, it will be recorded as a mistake. The number of correct actions versus mistakes will be recorded and displayed to the user after the completion of the testing scenario. | | |
| **Inputs:** Grab interaction with phone, simple interaction with patient to lie them down, controller position, location, and velocity. | | |
| **Processing:** Each user interaction will be processed and compared to a ground truth order for the interactions. If they match, a correct action will be logged, and if they don’t match, an incorrect action will be logged. The number of correct and incorrect actions should be displayed to the user at the end of the testing scenario, and a completion certification should only be given if there are zero incorrect actions. | | |
| **Outputs:** The number of correct and incorrect actions should be shown to the user at the end of the testing scenario. | | |
| **Error Handling:** If the user selects an incorrect interactable item, it will be logged as a mistake and displayed at the end of the testing scenario. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR3, FR5 | **Forward Dependencies:** FR44 |

### Functional Requirement 36

|  |  |  |
| --- | --- | --- |
| **ID:** FR41 | **Title:** Head trauma: testing scenario | |
| After users complete the training scenario, they will unlock the ability to select the testing scenario from the training/testing menu (FR3). Upon selecting testing from the head trauma scenario, they will be placed in a scenario identical to the training scenario for head trauma, but this time there will be no UI interaction indicators or audio indicators to tell them what to do. They must carry out the steps on their own. To successfully pass the testing scenario, they must carry out the following steps in the proper order with no mistakes:   1. Call 911 2. Keep person still 3. Apply pressure to open wounds 4. Monitor patient and keep them alert.   If the user interacts with an unnecessary object or performs a step out of order, it will be recorded as a mistake. The number of correct actions versus mistakes will be recorded and displayed to the user after the completion of the testing scenario. | | |
| **Inputs:** Grab interaction with phone, simple interaction with patient to keep them still, simple interaction with sink door handle, grab interaction with bandage, socket interaction for applying bandage, simple interactions for keeping patient alert. | | |
| **Processing:** Each user interaction will be processed and compared to a ground truth order for the interactions. If they match, a correct action will be logged, and if they don’t match, an incorrect action will be logged. The number of correct and incorrect actions should be displayed to the user at the end of the testing scenario, and a completion certification should only be given if there are zero incorrect actions. | | |
| **Outputs:** The number of correct and incorrect actions should be shown to the user at the end of the testing scenario. | | |
| **Error Handling:** If the user selects an incorrect interactable item, it will be logged as a mistake and displayed at the end of the testing scenario. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR3, FR5 | **Forward Dependencies:** FR44 |

### Functional Requirement 37

|  |  |  |
| --- | --- | --- |
| **ID:** FR42 | **Title:** Heimlich maneuver adult: testing scenario | |
| After users complete the training scenario, they will unlock the ability to select the testing scenario from the training/testing menu (FR3). Upon selecting testing from the Heimlich maneuver adult scenario, they will be placed in a scenario identical to the training scenario for Heimlich maneuver adults, but this time there will be no UI interaction indicators or audio indicators to tell them what to do. They must carry out the steps on their own. To successfully pass the testing scenario, they must carry out the following steps in the proper order with no mistakes:   1. Address the situation and ensure patient needs assistance 2. Give five back blows 3. Give five abdominal thrusts 4. Call 911   If the user interacts with an unnecessary object or performs a step out of order, it will be recorded as a mistake. The number of correct actions versus mistakes will be recorded and displayed to the user after the completion of the testing scenario. | | |
| **Inputs:** Simple interaction with patient to check if they need assistance, controller position, location, and velocity for back blows and abdominal thrusts, grab interaction with phone. | | |
| **Processing:** Each user interaction will be processed and compared to a ground truth order for the interactions. If they match, a correct action will be logged, and if they don’t match, an incorrect action will be logged. The number of correct and incorrect actions should be displayed to the user at the end of the testing scenario, and a completion certification should only be given if there are zero incorrect actions. | | |
| **Outputs:** The number of correct and incorrect actions should be shown to the user at the end of the testing scenario. | | |
| **Error Handling:** If the user selects an incorrect interactable item, it will be logged as a mistake and displayed at the end of the testing scenario. | | |
| **Priority:** MEDIUM | **Backward Dependencies:** FR3, FR5 | **Forward Dependencies:** FR44 |

### Functional Requirement 38

|  |  |  |
| --- | --- | --- |
| **ID:** FR44 | **Title:** Testing scenarios: post-scenario evaluation UI | |
| Upon completing a testing scenario, the user should see a UI that gives them feedback on their performance in the testing scenario. It should contain the number of correct actions taken, the number of mistakes, and whether they passed the scenario or not. | | |
| **Inputs:** Completion of final test scenario step, number of mistakes, number of correct actions | | |
| **Processing:** If the number of mistakes is greater than zero, the user did not pass the testing scenario and should retry. | | |
| **Outputs:** A UI displaying the number of correct actions, number of mistakes, and if the user passed or not. | | |
| **Error Handling:** None | | |
| **Priority:** LOW | **Backward Dependencies:** FR38-43 | **Forward Dependencies:** FR4 |

## Non-Functional Requirements

Non-functional requirements are the general properties of our medical training tool. They are the higher-level requirements that describe the tool itself, rather than specific pieces and parts of the tool. They are broken down into performance requirements, reliability requirements, availability requirements, security requirements, maintainability requirements, and portability requirements.

### Performance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR1 | **Title:** Minimum frame rate of 60fps | | | |
| Throughout the navigation of our training tool, the framerate should not drop below 60 frames per second. Framerates lower than this can cause sickness for our users. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** Framerate over 60 |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** MEDIUM | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR2 | **Title:** Response time | | | |
| When the users interact with an action or elsewise trigger an event, the scenarios should update quickly. | | | | |
| **Inputs:** Any user input | | **Processing:** Any process | | **Outputs:** Rapid processing |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** MEDIUM | | | | |

### Reliability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR3 | **Title:** Tool should be reliable in terms of limited crashes and bugs | | | |
| Throughout the testing of our app, we should experience crashes on none of our testing runs. Also, after testing the tool, we should encounter zero bugs. We should have a way for feedback to be sent to the development team to report bugs. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** Reliable Tool |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** MEDIUM | | | | |

### Availability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR4 | **Title:** Tool should be available to all people who own a VR headset | | | |
| Our tool should be able to be downloaded onto VR headsets. It will not require an internet connection, so the tool should be usable anywhere. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** None |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** LOW | | | | |

### Security

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR5 | **Title:** All data should be stored securely | | | |
| Data collected from users will be stored securely. This includes the passing of data from the tool itself to the source of storage. Data secured includes progress through each scenario and scores for each test scenario. | | | | |
| **Inputs:** Data from user | | **Processing:** Security and encryption of the data | | **Outputs:** Secure data |
| **Error Handling:** If data can’t be stored securely, data should not be maintained, and error message should be shown to user. | | | **Dependencies:** FR13, FR14, FR15 | |
| **Priority:** LOW | | | | |

### Maintainability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR6 | **Title:** Tool should be implemented in a way that allows for extendibility | | | |
| Our tool should be implemented in a way that the code is easy to extend. If desired, software developers should be able to add features and functionality to our tool. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** None |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** MEDIUM | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR7 | **Title:** Tool should be implemented in a way that allows for modification | | | |
| Our tool should be implemented in a way that the code is easy to modify. If desired, software developers should be able to change and add steps to the current scenarios. Often, first aid protocols are updated, so our app should be able to adapt to teach the latest information. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** None |
| **Error Handling:** N/A | | | **Dependencies:** All requirements | |
| **Priority:** LOW | | | | |

### Portability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** NFR8 | **Title:** Tool should be available on Oculus Quest and Rift VR headsets | | | |
| Our tool should run on the main Meta VR headsets available, including Oculus Quest and Rift headsets. | | | | |
| **Inputs:** None | | **Processing:** None | | **Outputs:** None |
| **Error Handling:** N/A | | | **Dependencies:** None | |
| **Priority:** LOW | | | | |

## Design Constraints

Due to the design and implementation of VR controllers, it is impossible to implement some actions that require specific conditions or fine motor skills of human hands. For example, bandage wrapping, and head trauma scenario would require precise interactions with accuracy of human hands like opening a small bandage packaging, unwrapping bandage, or wrapping bandage around body part. Also, with VR controllers being basically in the air, it is also impossible to implement physical resistance like in CPR procedure.

Next are the limitations of XR Interaction Toolkit which is used for VR development in Unity Engine. Oculus Quest supports hand tracking without controllers which could alleviate the issues mentioned in previous paragraph. However, this functionality is not implemented in XR Interaction Toolkit. These limitations call for alternative implementations or simplified versions of certain interactions.

Processing power of Meta VR headsets, which are target platform for the application, should also be accounted for in the project. Graphical options such as complexity of the environment and 3D model resolutions need to be limited to accommodate for smooth and stable performance.

Considering the time constraint within which the project needs to be completed and set deadlines, there is a limit for scope of certain disciplines, such as what can be researched, implemented, and accomplished. Part of these restrictions are reflected in functional requirements.

## Logical Database Requirements

Our medical training tool does not require any internet connectivity or login feature. The data generated throughout the use of the tool will be native to the device the tool is running on. There are no requirements that relate to storing data online in a cloud-based database, so the data will be stored in a local storage database. The database will store the following information:

For the testing scenarios:

1. Name of test attempt – which scenario has been attempted
2. Score for each test attempt
   * Will be calculated as (correct actions) / (total actions)
3. Mistakes for each attempt – the step at which the mistake was made
4. Date and time at which the attempt was made
5. ID of each record – used to sort records

For training scenario:

1. Save state of training scenario – which part has been completed and where has the user stopped
2. Name of the training scenario
3. Date recorded

Note that for testing scenarios, the step at which the user leaves the scenario is not saved. The testing scenarios are intended to be completed in one attempt, so any time the user leaves the testing scenario, their progress will be reset.

## Other Requirements

Each scene will use a plethora of 3-dimensional models which the user will interact with to accomplish the tasks of each scenario. Animations will also be shown that move each scenario from one state to another. Audio combined with text will be used to convey the instructions of how to complete each step in the scenario. The animations, text, and audio queues should all be aligned in terms of both timing and message. That is, all three modes of communicating which step should be taken next should convey the same message and not contradict one another.

# Analysis Models

## Data Flow Diagram (DFD)

Diagram

Description automatically generated

Figure 6: Data Flow Diagram

# Appendices

## Appendix A: Development Plan

### Project Overview

Currently, CPR and basic medical training is only accessible through online instruction and physical classes. These methods present two problems: online classes lack hands-on experience in medical training, while in-person classes often cost money and are inconvenient to attend. Our solution to this problem is to create a virtual reality training tool that will be widely accessible to anyone with a VR headset. This tool will supply hands-on experience in the comfort of the user’s own home, and will teach various medical tools everyone should know, such as CPR, the Heimlich Maneuver, and more.

### Scope, Purpose, and Objectives

This project provides a virtual reality tool that will make basic medical training accessible to anyone with a VR headset. This tool will be a cheap alternative to in-person classes and will be more convenient for the user. Also, in comparison to online video lessons on the given topics, our tool will provide the user with more hands-on experience through a virtual reality setting. This means our users will be more prepared for real-world scenarios than those who only use online resources to learn these topics.

The scope of the project is to cover the basic medical tools and procedures everyone should know. These include CPR and resuscitation, the Heimlich maneuver, bandaging wounds, dealing with head trauma, and applying a tourniquet. For each of these tools, we will supply basic training within the virtual setting, then ask the users to carry out a test. The user’s performance will be tracked, and feedback will be given to them at the end of the test. The goal is that, after the user has received a passing test score on each scenario, they will be able to recognize when to use the tool and how to apply the tool to save someone’s life in the real world. Our scope does not include car accident first aid, motorcycle accident first aid, shrapnel wounds, or heat stroke.

Our objective with this project is to deliver a VR simulation tool that has four different scenarios. Each scenario will teach the user useful and correct medical first aid for common encounters that everyone should know. The scenarios are bandaging a wound and applying tourniquet, CPR and resuscitation, the Heimlich maneuver, and handling head trauma. Each scenario will start with a brief but detailed explanation (with visual aid) of the equipment being used, when to use it, and how to use it properly. The user will then be walked through using said equipment step by step. Any misuse will cause an alert to pop up informing the user of the proper way to use that equipment. After the user successfully completes the walk-through, the user will be prompted to do the simulation again, without any help. Upon completion of the scenario, the user will be shown the report with each step being evaluated as pass or fail.

### Team Organization

For this project, Jeremy will be the team lead. He has leadership experience from his job and athletic career as a captain of the Cross-Country team. He knows what it takes to lead a group and will ensure everyone is doing their part and staying on track. He is also the UI/Frontend lead since he has experience developing user interfaces from his app development endeavors. He will also serve as the backup documentation lead.

Robert will be the documentation lead for this project and will handle proofreading documents prior to submission and ensuring the formatting is correct and consistent. He will also serve as the backup quality assurance lead and the backup UI/Frontend lead.

Slav will be the presentation lead this project and will handle completing our presentations. This includes ensuring text is readable and minimal, images and graphics are of high quality, and all slides are numbered properly. He will also be the quality assurance lead, which means he will oversee the overall quality of the project in terms of visual quality, performance quality, and general code quality.

Finally, Luka will be the backend lead for this project and will oversee the organization of files and data within the project. If our project ends up requiring a database, he will handle keeping it organized and functional. He will also serve as the backup presentation lead.

For each of the above leadership roles, the section leader will have the final say in their given area of leadership. That is, the documentation lead will have the final say in what is included in a document, the formatting, and the submission of the document. The same goes for all other leadership roles. If the leader needs a second opinion or for some reason cannot fulfill their duty, the backup will step in and aid the leader. Any of the section leaders can also turn to the team lead, Jeremy, if they need help as well.

### Problem Resolution Policy

For this project, we will follow a three-strike policy. The first time a group member misses a meeting unexcused or does not deliver a piece of code without reason, we will discuss the issue within the group and attempt to solve it ourselves. The second time a team member misses or does not deliver, we will turn to the Graduate Teaching Assistant for aid in dealing with the situation, and if that does not solve the issue, we will ask the professor for assistance on the third strike.

When a team member makes a pull request on GitHub, we have a discord bot set up in our server to alert everyone that a pull request was made. Before being merged, at least one other group member will need to review the code and resolve any conflicts. This helps to ensure that no code with issues is merged with our main branch.

If there are disputes on technology, we will simply take a team vote between the different options at hand, and the technology that the majority chooses will be used. Since our group has 4 people, if it is split 2 and 2, we will ask the TA to break the tie and select the technology or propose a different one that they think is more suited for the task.

### Project Plan

During this project, we plan to meet as a group 3 times per week. Of those 3 meetings, one will be with the Graduate Teaching Assistant. This will ensure that we are all on the same page, and if someone is stuck, they can get help quickly. Jim is our client this semester, and we plan to meet with him one day a week for at least an hour.

Weekly Schedule:

* Team meetings: Monday before/after class, Tuesday 2:30pm, Thursday 3:00pm, Sundays as needed
* Meeting with TA: Not determined yet, will be decided based on TA’s schedule
* Meeting with Client: Thursdays at 3:00pm

Chart, waterfall chart

Description automatically generated

*Figure 1. Gantt chart representing the weekly meeting schedule of the team*

Schedule:

* 2/3 First Prototype: Basic menus will be functional, each training scenario will take the user to a different room, with some interactable 3D objects.
* 2/12 Requirements: SRS document, including all functional and non-functional requirements
* 2/26 Design Specification
* 3/3 Second Prototype: Information about how to carry out the scenarios will be displayed, as well as most of the 3D assets being present and functional.
* 3/24 Third Prototype: All critical functionality tested and working, only some UI updates, polishing, and minor tests remain.
* 4/2 Test Plan
* 4/16 Final Prototype: Project should be finished, including all UI updates and testing of all features

Timeline

Description automatically generated*Figure 2. Gantt chart representing the flow of the project with assigned due dates.*

### Configuration Management Plan

The team will use GitHub Project with Kanban board configuration. Issues will be linked to it, so the team and Graduate Teaching Assistant can easily see the visual representation of the current progress of the project.

GitHub was also selected to host the codebase and any changes made to the code will be pushed to the remote repository. Our repository will have a separate development branch for each group member. Members are to commit their changes to their branch. Once team members complete feature development, they will create a pull request to have it reviewed by other team members. Once a pull request is created, it must be reviewed by at least one team member before it can be merged into the main branch. This will ensure our main branch remains intact and limited to bugs. Once the review is finished, code will be merged into the main branch. On the GitHub repository, the main branch will be protected to not allow direct commits.

Documentation and supplemental materials (assets) required for feature functionality that cannot be hosted on GitHub will be shared amongst group members in a shared OneDrive folder, so each team member can easily get features running on their system.

### Technology

Our project will utilize the Unity engine for development (editor version 2021.3.17f1). This relies on the C# programming language. Unity comes with a built-in API which will be used heavily to implement the features selected for implementation. Project will mainly target Oculus VR platform with Rift and Quest 2 headsets used for testing. Blender (V3.4.1) will be used for 3D modelling.

## Appendix B: Priority Matrix

For the matrix below, we used quantitative values rather than qualitative values as shown in the requirements section of this document. This was done so that the priority values could be averaged and rounded in a meaningful way. The quantitative values were then cast to qualitative values using the following scheme:

|  |  |
| --- | --- |
| Quantitative Value | Qualitative Value |
| 3 | HIGH |
| 2 | MEDIUM |
| 1 | LOW |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Requirement | Jeremy | Luka | Slav | Robert | Average | Priority |
| FR1 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR2 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR3 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR4 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR5 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR6 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR7 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR8 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR9 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR10 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR11 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR12 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR13 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR14 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR15 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR16 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR17 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR18 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR19 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR20 | 1 | 1 | 1 | 2 | 1.25 | 1 |
| FR21 | 2 | 2 | 2 | 2 | 2 | 2 |
| FR22 | 1 | 2 | 1 | 1 | 1.25 | 1 |
| FR23 | 2 | 2 | 2 | 3 | 2.25 | 2 |
| FR24 | 1 | 1 | 1 | 1 | 1 | 1 |
| FR25 | 2 | 3 | 3 | 3 | 2.75 | 3 |
| FR26 | 3 | 2 | 3 | 3 | 2.75 | 3 |
| FR27 | 3 | 2 | 3 | 3 | 2.75 | 3 |
| FR28 | 3 | 2 | 3 | 3 | 2.75 | 3 |
| FR29 | 3 | 2 | 3 | 3 | 2.75 | 3 |
| FR30 | 3 | 3 | 3 | 3 | 3 | 3 |
| FR31 | 1 | 2 | 1 | 3 | 1.75 | 2 |
| FR32 | 3 | 2 | 2 | 3 | 2.5 | 3 |
| FR33 | 2 | 1 | 1 | 1 | 1.25 | 1 |
| FR34 | 3 | 2 | 2 | 2 | 2.25 | 2 |
| FR35 | 2 | 3 | 2 | 1 | 2 | 2 |
| NFR1 | 3 | 2 | 2 | 2 | 2.25 | 2 |
| NFR2 | 2 | 2 | 2 | 2 | 2 | 2 |
| NFR3 | 2 | 2 | 2 | 2 | 2 | 2 |
| NFR4 | 1 | 1 | 1 | 1 | 1 | 1 |
| NFR5 | 1 | 1 | 1 | 1 | 1 | 1 |
| NFR6 | 2 | 2 | 2 | 2 | 2 | 2 |
| NFR7 | 1 | 1 | 1 | 1 | 1 | 1 |
| NFR8 | 1 | 1 | 1 | 1 | 1 | 1 |