

DESIGN DOCUMENT

Version 4.0
April 21, 2011



Revision Sign-off

By signing the following, the team member asserts that he has read the entire document and has, to the best of his knowledge, found the information contained herein to be accurate, relevant, and free of typographical error.

Name	Signature	Date
Michael Fore		
Sean Owen		
Anh Pham		
Jeffrey Regan		
Alex Welsh		
Matthew Williams		

Revision History

The following is a history of document revisions.

Version	Changes	Edited
Version 1.0	Initial draft.	12/01/2010
Version 2.0	Iteration 2 draft. Updated cover page.	1/30/2011
Version 3.0	Updated class diagrams, use case diagram, general fixes.	2/28/2011
Version 4.0	Final updates.	4/21/2011

Contents

- Revision Sign-off ii
- Revision History iii
- 1. Introduction..... 1
 - 1.1 Purpose..... 1
 - 1.2 Identification 1
 - 1.3 Scope..... 1
 - 1.4 Overview..... 1
- 2. Glossary 3
- 3. System Architecture..... 4
 - 3.1 Hardware and Communication Architecture 4
 - 3.2 Software Architecture 4
- 4. Database Architecture 5
 - 4.1 Database Model 5
 - 4.2 Tables..... 5
- 5. Detailed Design..... 7
 - 5.1 Class diagrams 7
 - 5.1.1 Framework Package (Surface)..... 7
 - 5.1.2 *Healing Vision* Package 10
 - 5.2 Use-case model 11
 - 5.3 Sequence diagrams..... 12
 - 5.3.1 Framework 12
 - 5.3.2 Healing Vision 13
- 6. External Interface Architecture..... 19
- 7. Human Machine Interface..... 20

1. Introduction

1.1 Purpose

This document presents a detailed design description of the *Healing Touch* system. Using text and diagrams this document depicts how the user will interact with the system as well as illustrating how each component of the system interacts with one another.

1.2 Identification

The goal of the *Healing Touch* system is the development of a computerized system designed to provide therapeutic and rehabilitative exercises through gaming on multi-user, multi-touch devices. The *Healing Touch* system is devised to be symmetric among the two multi-touch gaming platforms available to the program: the Microsoft Surface and the Apple iPad. Therefore, the user interface for the Microsoft Surface will be almost identical to that of the Apple iPad. Along with the gaming framework, the *Healing Touch* system consists of a standalone application, *Healing Vision*, and a database.

1.3 Scope

Healing Touch will initially be released with the support for: a uniform framework for both platforms that allows for the augmentation of games, a standalone application for clinician usage, and a database that accommodates the patient and game data.

1.4 Overview

The remainder of the document consists of the following sections:

Section 2 - Glossary: Informs the reader of any technical terms and/or abbreviations used throughout this document.

Section 3 - System Architecture: Summary of the hardware, communication architecture, and software architecture. Communication architecture and hardware descriptions focus on the two multi-user gaming platforms utilized. In the software architecture section, a system diagram is included which outlines the interaction between software components.

Section 4 - Database Architecture: Summary of the database architecture as well as an overview of the tables used to store and retrieve patient and game information.

Section 5 - Detailed Design: Information pertaining to the detailed design of the *Healing Touch* system. In the software section, sequence and class diagrams present how the user interacts with the system and how different components within the system interact with each other. In the

database section, diagrams outline the columns of each table and how the tables relate to each other.

Section 6 - External Interface Architecture: Informs developers how to develop a game that can be augmented into the framework.

Section 7 - Human Machine Interface: Overview of the graphical user interface (GUI) using a series of paper prototypes.

2. Glossary

Apple iPad	The Apple iPad is a touch-interface tablet computer designed and distributed by Apple.
<i>Healing Touch</i> Framework	The framework is the backend software that allows a user to login and choose a game. It also handles data collected from game plays and sends this data to the database.
Microsoft Surface	Microsoft Surface is a multi-touch product from Microsoft which is a combination of software and hardware technology that allows a user, or multiple users, to manipulate digital content by the use of gesture recognition.
THR	Texas Health Resources
Session	Time between patient log in and log out
Healing Vision	Standalone application designed to access the <i>Healing Touch</i> database

3. System Architecture

3.1 Hardware and Communication Architecture

The *Healing Touch* system consists of four primary hardware components:

- 1) Multi-touch, multi-user devices (Microsoft Surface and Apple iPad) running the *Healing Touch* application
- 2) Clinician workstation to run the *Healing Vision* application
- 3) Database for the storage of clinician profiles, patient profiles, and game results
- 4) Wireless access point for providing network access and communication between components

3.2 Software Architecture

Figure 3.1 represents the *Healing Touch* software architecture. It outlines the structure of the system, as well as, the flow of data.

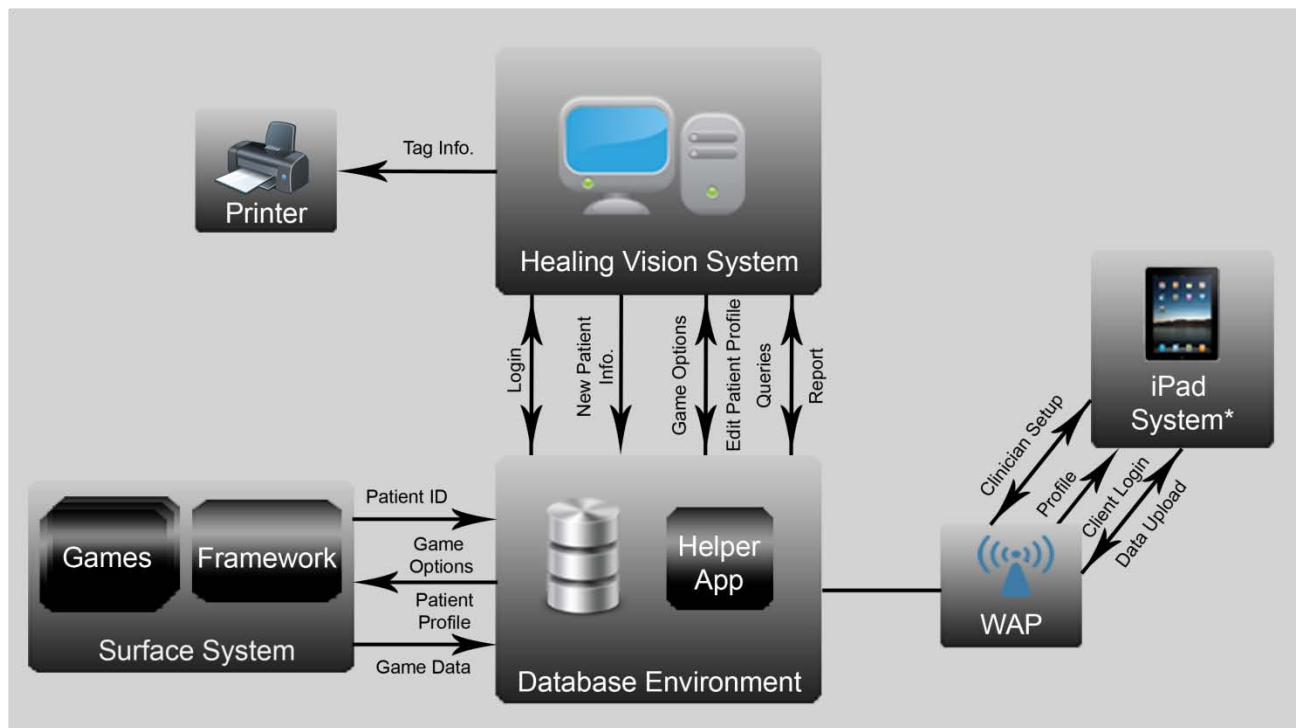


Figure 3.1 System Architecture

4. Database Architecture

4.1 Database Model

Figures 4.1 and 4.2 represent the database architecture. It outlines the contents of the database, as well as how the tables within the database relate to each other.

4.2 Tables

There are two parts to the database design: the patient tables and the game tables. All patient tables are interconnected through the key PatientInternalID which is a unique number assigned to the patient once they are registered in the system. In the Game table, each game is given an ID which is referenced throughout the other games tables. Each game will have a game options and game data table. The PatientInternalID is again referenced in the GameData table where data on all gameplay sessions for each patient is recorded.

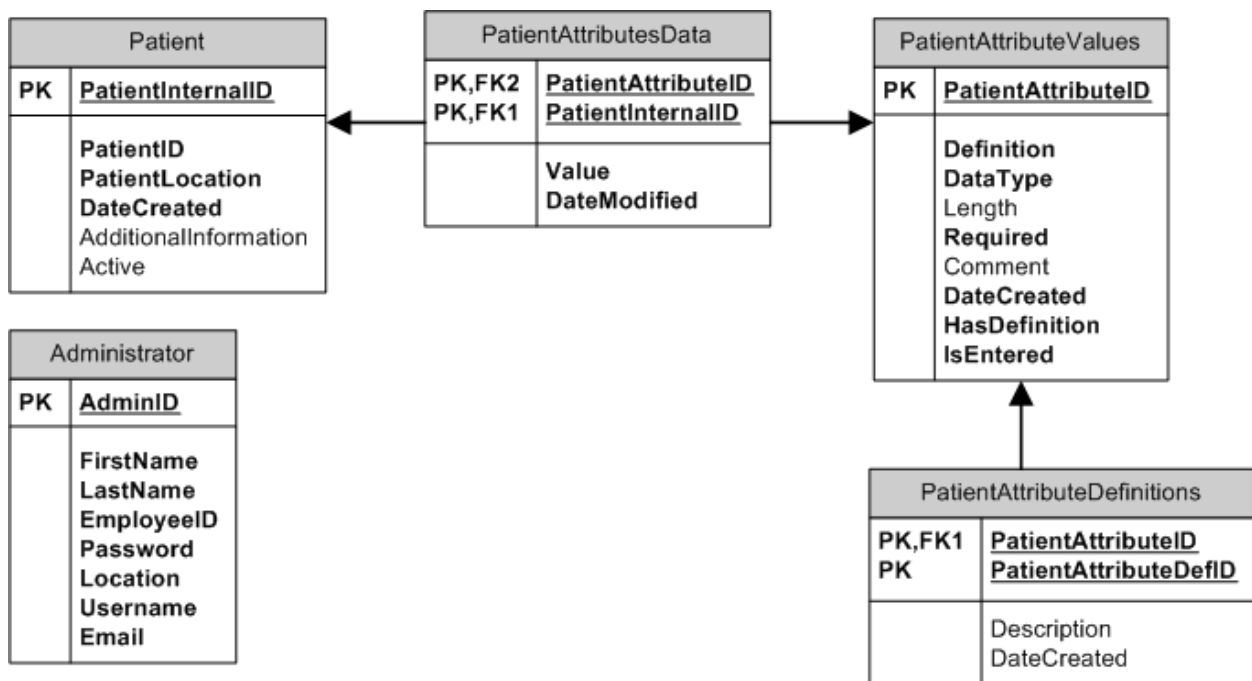


Figure 4.1 Patient Tables

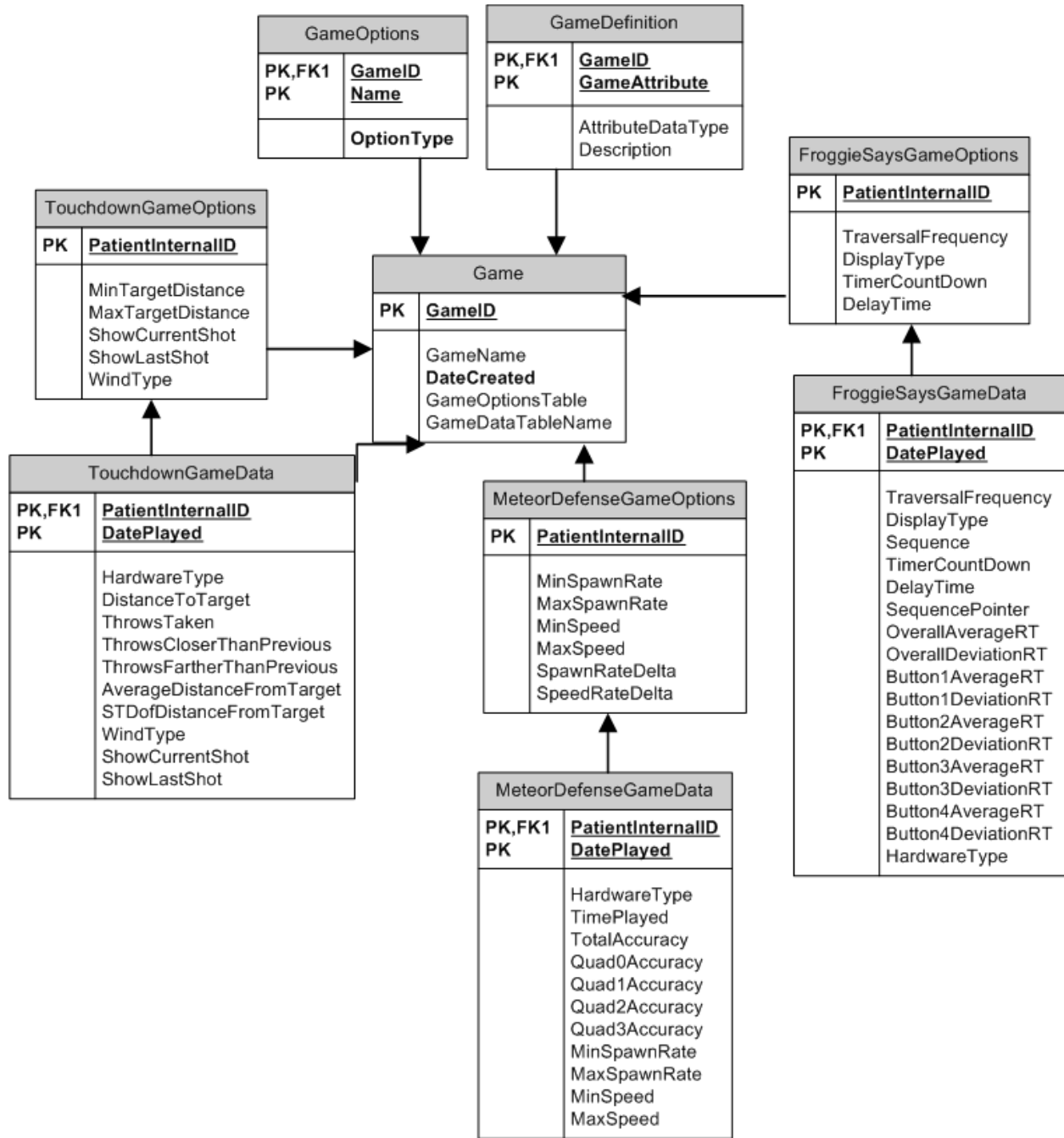


Figure 4.2 Game Tables

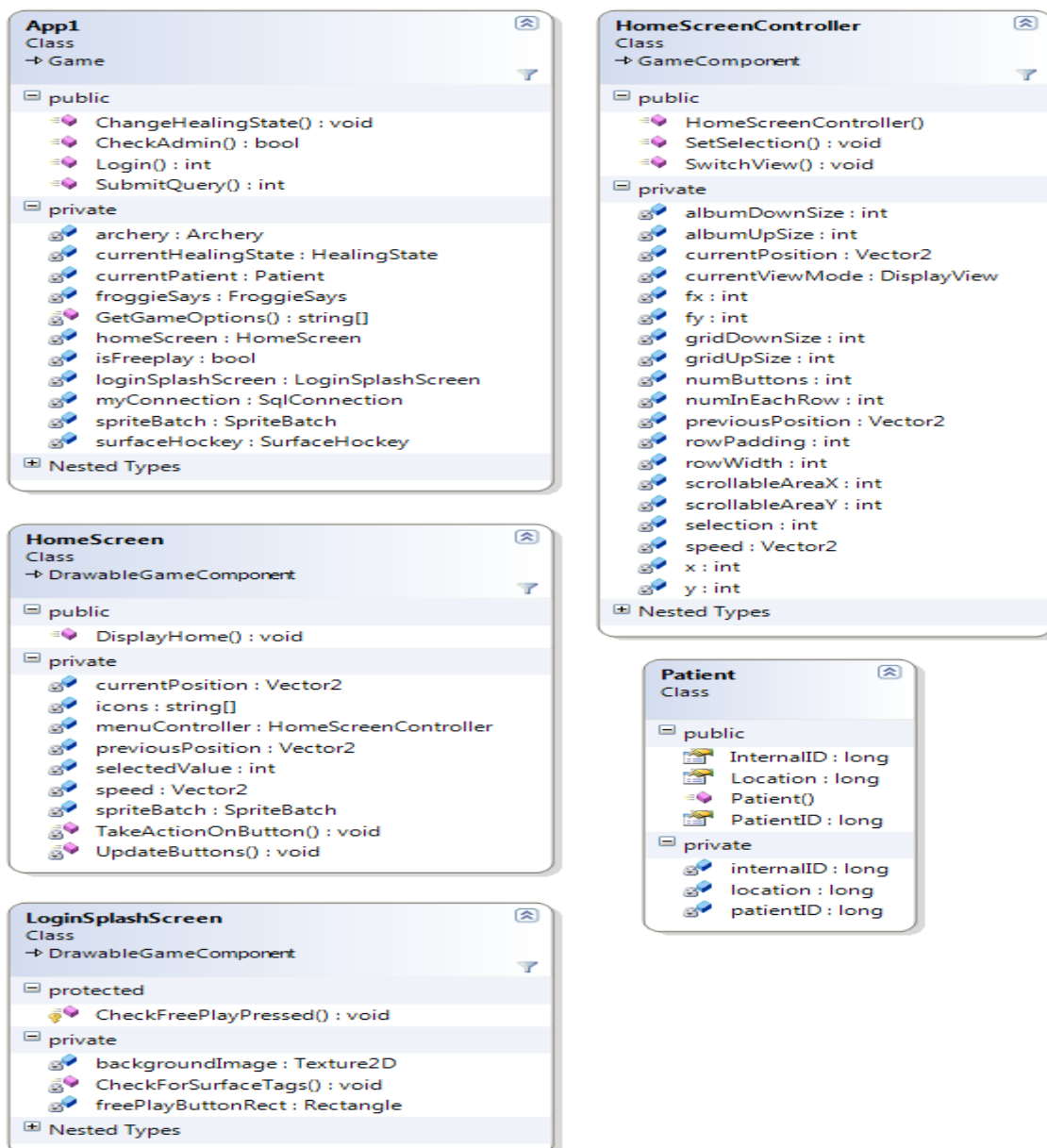
5. Detailed Design

In this section, the design overview is expanded with class and sequence diagrams. It is important to note that these diagrams map *Healing Touch's* software and hardware foundation.

5.1 Class diagrams

Class diagrams are used to illustrate the structure of a system by showing the system's classes, the class's attributes, and inter-class relationships.

5.1.1 Framework Package (Surface)



RadioButton
Class
→ DrawableGameComponent

```

public
    Draw() : void
    Hit_Image_Alpha() : bool
    Initialize() : void
    RadioButton()
    RelocateRects() : void
    TranslateX() : void
    TranslateY() : void
    Update() : void

private
    albumTexture : Texture2D
    color : Color
    currentState : RState
    downRectangle : Rectangle
    downSize : int
    gridTexture : Texture2D
    Hit_Image() : bool
    Hit_Image_Alpha() : bool
    isAlbumView : bool
    isSelected : bool
    radioDownTexture : Texture2D
    radioTexture : Texture2D
    radioTexturePath : string
    singleImage : bool
    spriteBatch : SpriteBatch
    upRectangle : Rectangle
    upSize : int
    x : int
    y : int
    
```

Nested Types

Button
Class
→ DrawableGameComponent

```

public
    Button()
    Draw() : void
    Hit_Image_Alpha() : bool
    Initialize() : void
    Update() : void

protected
    LoadContent() : void

private
    buttonPressedTexture : Texture2D
    buttonTexture : Texture2D
    buttonTexturePath : string
    color : Color
    currentState : BState
    height : int
    Hit_Image() : bool
    Hit_Image_Alpha() : bool
    isButtonPressed : bool
    rectangle : Rectangle
    spriteBatch : SpriteBatch
    width : int
    x : int
    y : int
    
```

Nested Types

ScrollBar
Class
→ DrawableGameComponent

```

public
    ScrollBar()
    UpdateScrollbar() : double

private
    backgroundRec : Rectangle
    diff : int
    font : SpriteFont
    fPressed : bool
    fx : int
    fy : int
    GetSelectedValue() : void
    hasMultiplier : bool
    Hit_Image() : bool
    Hit_Image_Alpha() : bool (+ 1 overload)
    isFocused : bool
    isSliding : bool
    maxValue : double
    minValue : double
    multiplier : double
    offset : double
    prevFPressed : bool
    scrollBackground : Texture2D
    scrollButton : Texture2D
    scrollName : string
    scrollRec : Rectangle
    selectedValue : int
    spacing : int
    spriteBatch : SpriteBatch
    valueDisplay : int
    width : int
    x : int
    y : int
    
```

TagData
Class

```

public
    ChangeCurrentIdState() : void
    currentState : IdState
    TagData() (+ 1 overload)

private
    byteTag : byte
    identitySeries : string
    identityValue : string
    location : Vector2
    orientation : float
    
```

Nested Types

SpriteData
Class

```

public
    SpriteData()

private
    location : Vector2
    orientation : float
    scale : float
    
```

RadioButtonController
Class
→ GameComponent

```

public
    CheckRadioButtons() : int
    DisplayRadioButtons() : void
    IsFocused : bool
    RadioButtonController()
    SetSelection() : void

private
    downSize : int
    fPressed : bool
    fx : int
    fy : int
    Initialize() : void
    isFocused : bool
    numButtons : int
    prevFPressed : bool
    radioButtons : RadioButton[]
    selection : int
    upSize : int
    x : int
    y : int
    
```

Sprite
Class

```

public
    Draw() : void
    LoadContent() : void
    position : Vector2
    scale : float
    size : Rectangle
    Sprite()
    spriteTexture : Texture2D
    
```

FroggieSays
Class
→ DrawableGameComponent

public

- ShowButtons() : void
- ToggleOptionScreen() : void

private

- beginTime : double
- bottomLeftAverageRT : double
- bottomLeftDeviationRT : double
- bottomRightAverageRT : double
- bottomRightDeviationRT : double
- currentFroggieType : int
- currentGameState : GameState
- delay : double
- EndGame() : void
- fPressed : bool
- froggyOptions : FroggyOptions
- fx : int
- fy : int
- GenerateRTValues() : void
- GetFroggieType() : string
- initialDelay : double
- initialTraversalFreq : double
- length : int
- location : int
- minute : int
- minuteRec : Rectangle
- overallAverageRT : double
- overallDeviationRT : double
- patientOptions : Button[]
- patternDown : bool
- playerToCompDelay : double
- prevFPressed : bool
- responseTimes : LinkedList<double>
- sec : int
- sequence : int[]
- tenSec : int
- timer : double
- timerIncrement : double
- topLeftAverageRT : double
- topLeftDeviationRT : double
- topRightAverageRT : double
- topRightDeviationRT : double
- totalBottomLeft : int
- totalBottomRight : int
- totalTopLeft : int
- totalTopRight : int
- traversalFreq : double

Nested Types

FroggyOptions
Class
→ DrawableGameComponent

public

- DisplayOptionsScreen() : void

private

- backButton : Button
- compFreq : ScrollBar
- delayScroll : ScrollBar
- gameType : RadioButtonController
- gameTypeNames : string[]
- parent : FroggieSays
- prevFPressed : bool
- timerScroll : ScrollBar
- UpdateOptions() : void

ArcheryOptions
Class
→ DrawableGameComponent

public

- ShowOptionsScreen() : void

private

- btnCancel : Button
- parent : Archery
- UpdateOptions() : void
- windSpeed : ScrollBar
- windStyle : RadioButtonController
- windStyleNames : string[]

SurfaceHockey
Class
→ DrawableGameComponent

private

- BeginGame() : void
- paddleSpeed : Vector2[]
- paddleX : int[]
- paddleY : int[]
- prevPaddleX : int[]
- prevPaddleY : int[]
- prevPuckX : int
- prevPuckY : int
- puckRect : Rectangle
- puckSpeed : Vector2
- puckTexture : Texture2D
- puckX : int
- puckY : int
- score1 : int
- score2 : int
- spriteBatch : SpriteBatch
- TestForCollisions() : void
- xDiff : int
- yDiff : int

Archery
Class
→ DrawableGameComponent

public

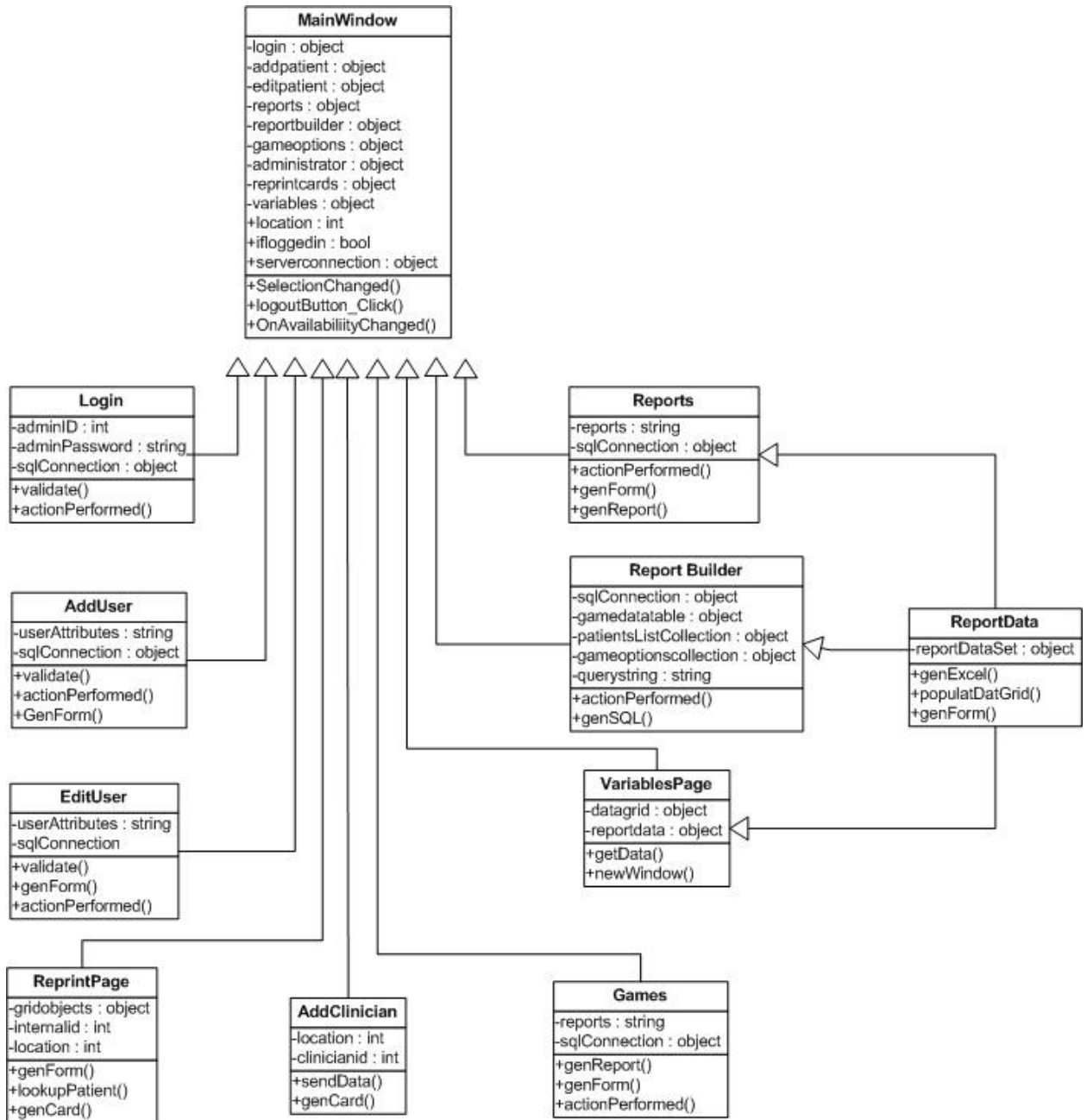
- Average() : void
- Draw() : void
- Initialize() : void
- NewWindValue() : void
- SetWindValue() : void
- ShowComponents() : void
- STD() : void
- SwitchOptionScreen() : void

private

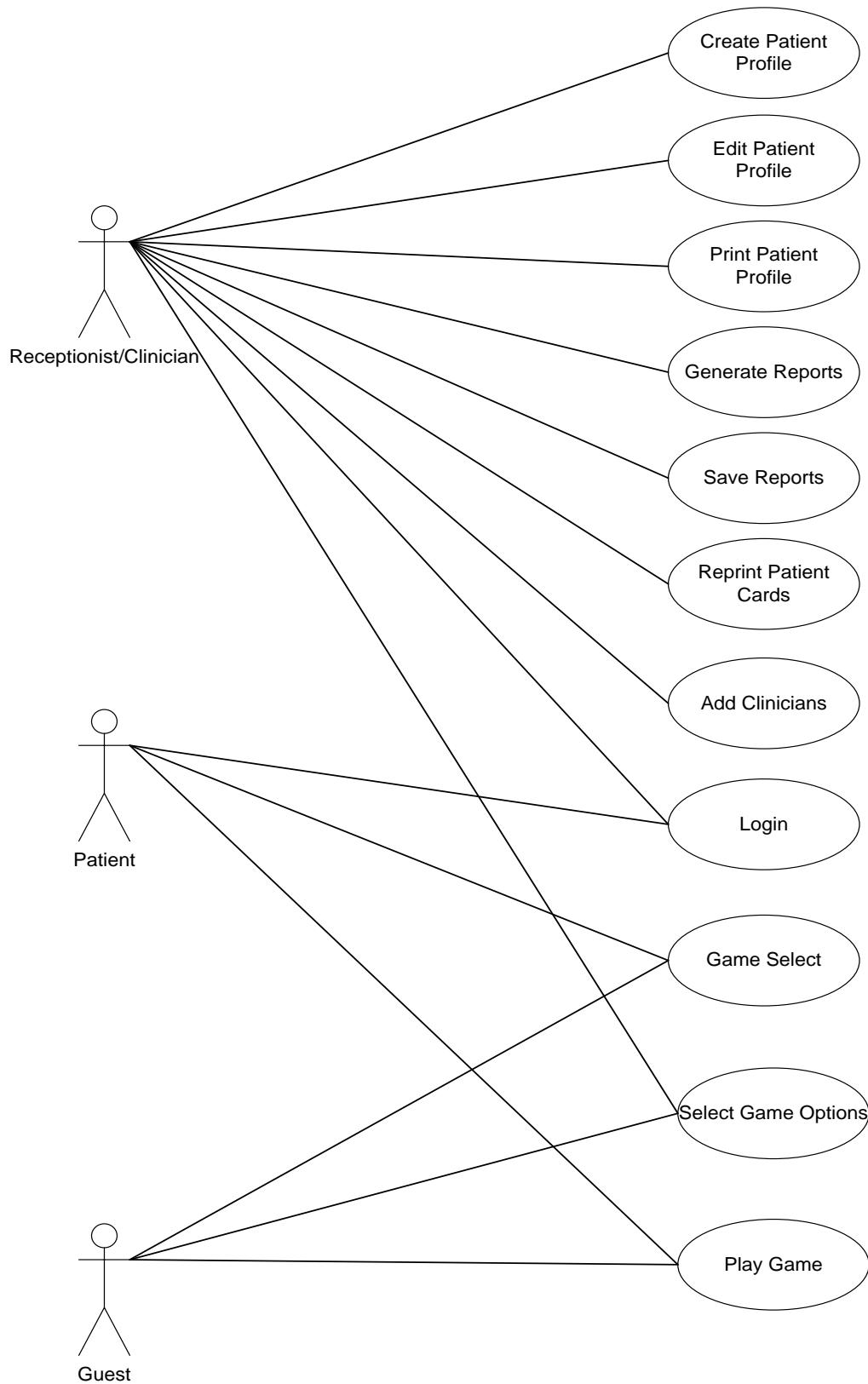
- acceleration : Vector2
- angle : double
- angleMagnitude : int
- average : double
- btnCancel : Button
- btnOption : Button
- current : Vector2
- EndGame() : void
- football : Sprite
- footballLength : int
- fx : int
- fy : int
- initial : Vector2
- LineAngle() : double
- LineMagnitude() : int
- magnitude : int
- maxTarget : int
- minTarget : int
- prevAngle : int
- prevFinger : bool
- prevFootballLocation : int
- prevMagnitude : int
- scrollMapSideways : bool
- shotsClosers : int
- shotsFarther : int
- shotsFired : int
- showCurrentShot : string
- showLastShot : string
- showLastShow : bool
- spriteBatch : SpriteBatch
- std : double
- targetsEnabled : bool
- targetOriginX : int
- targetOriginY : int
- throwsCaught : int
- velocity : Vector2
- wind : Vector2
- windType : string

Nested Types

5.1.2 Healing Vision Package



5.2 Use-case model

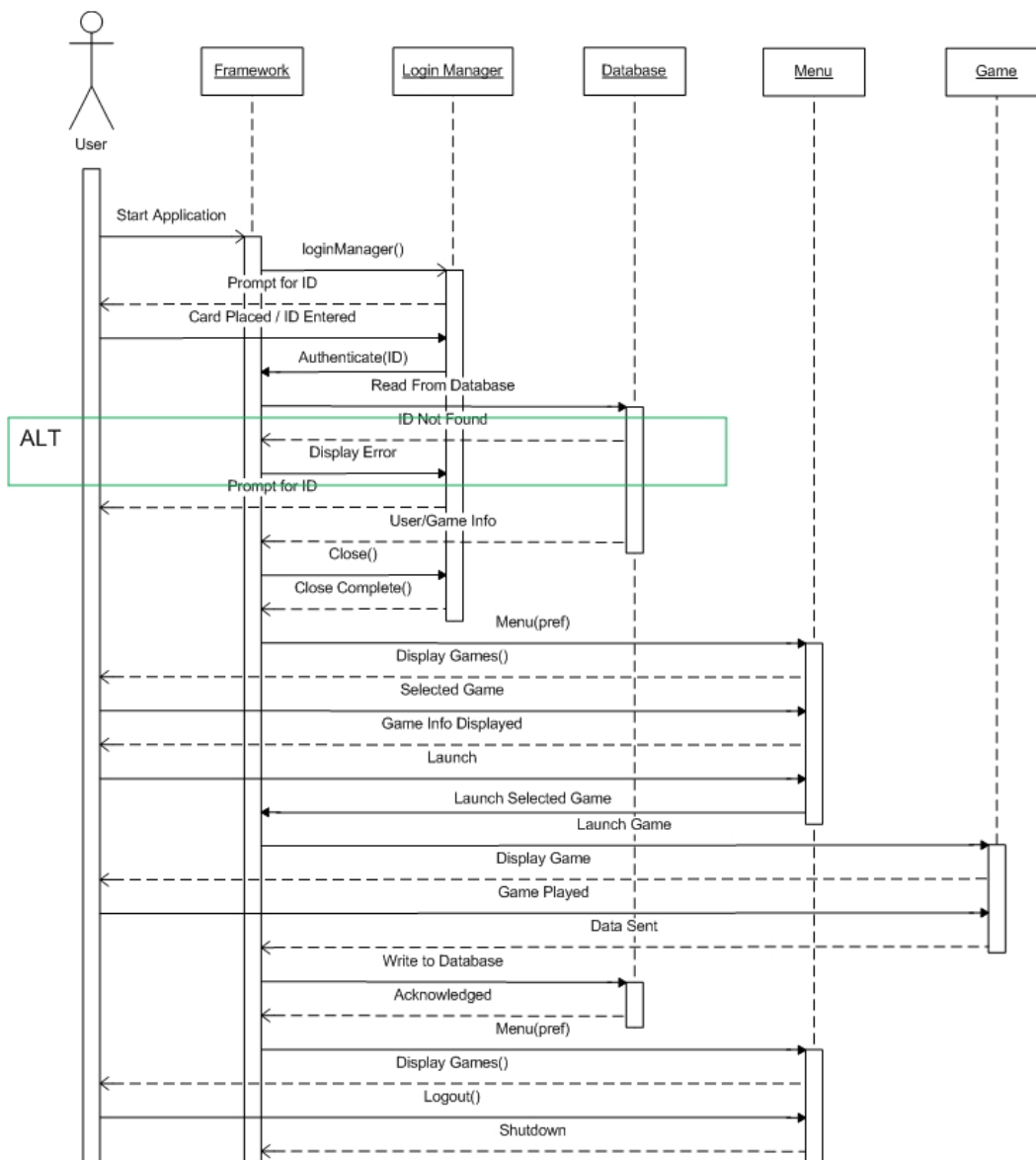


5.3 Sequence diagrams

Sequence diagrams are used to show how processes and components interact with one another during a series of events. In this document, the sequence diagrams are divided into the specific system components.

5.3.1 Framework

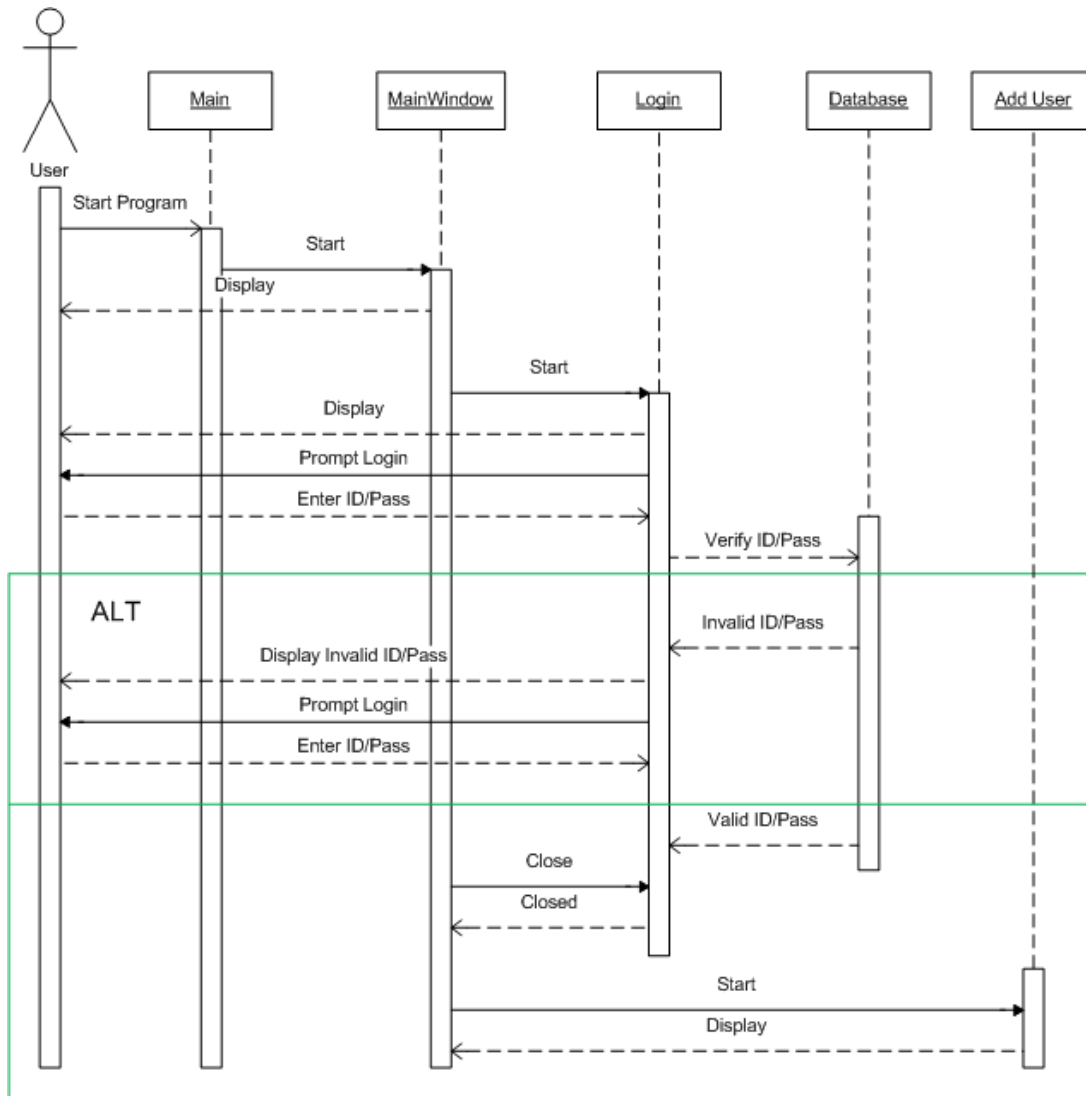
Once a user logs in, the Surface will require a connection to a server containing the relevant database. If this condition is not met, the login attempt fails. However, on the iPad, it will first synchronize with the server if there's a connection; otherwise it will use a local database.



5.3.2 Healing Vision

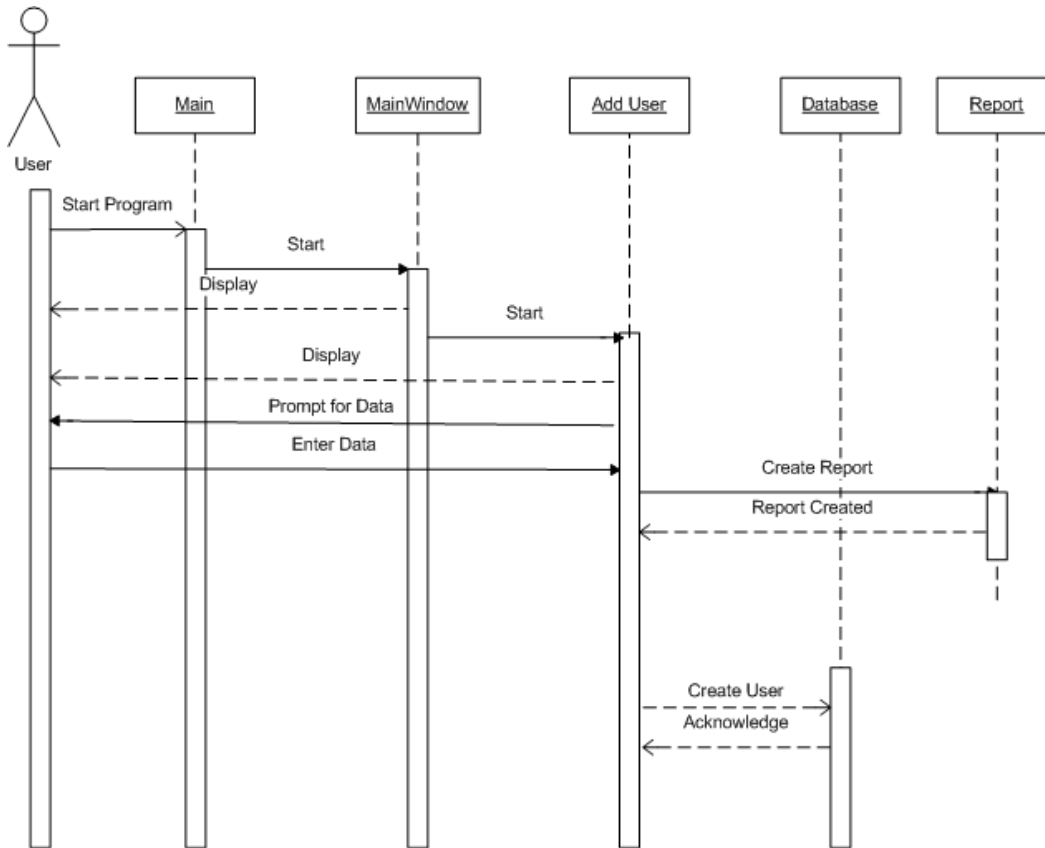
5.3.2.1 Healing Vision Login

When the user starts *Healing Vision*, the “Main” class is called which triggers the “MainWindow” class. From here, the login screen appears prompting the user to login. If the user enters incorrect credentials, they are notified that the credentials are incorrect and prompted once more. If the user enters the name and the database verifies the credentials are correct, the user is logged on and the add user screen appears.



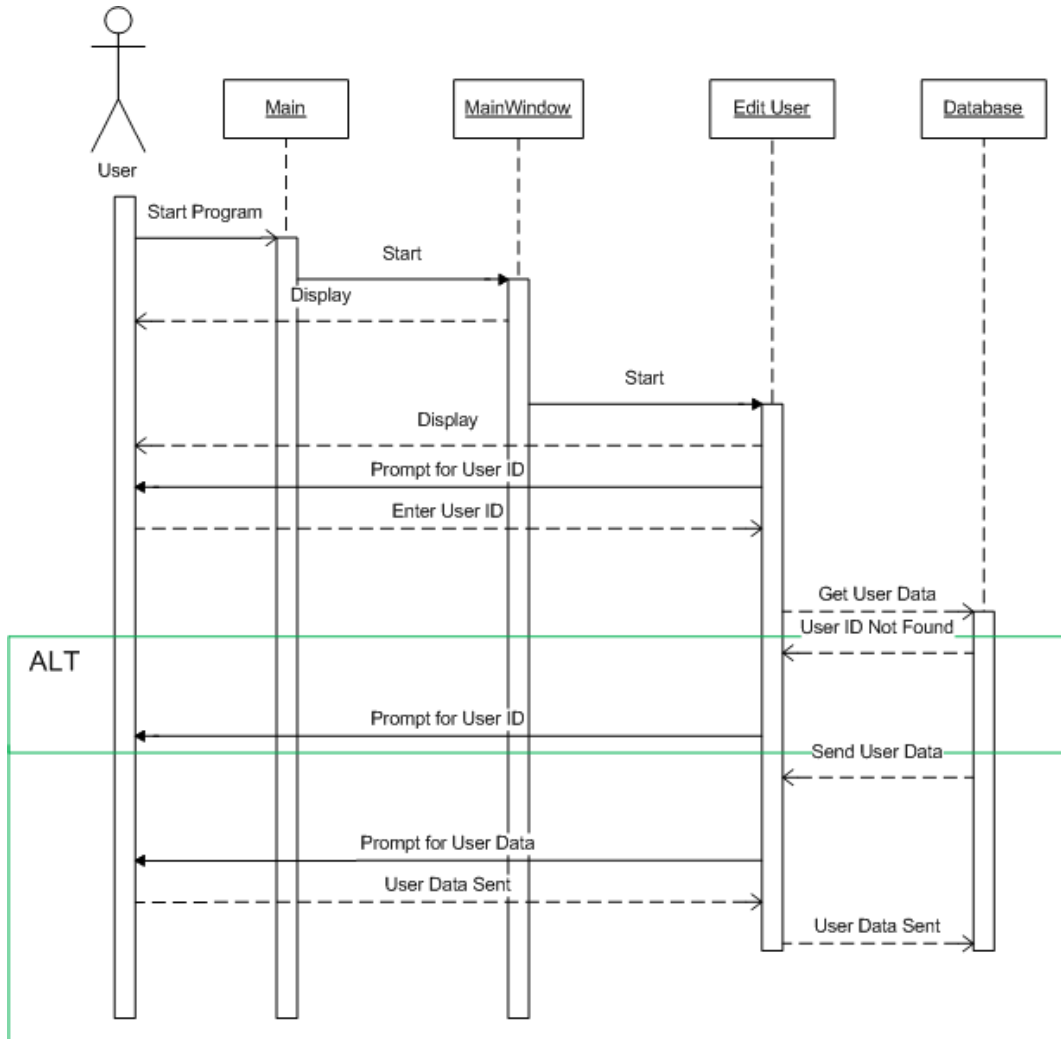
5.3.2.2 *Healing Vision Add User*

To add a patient, the user must fill out the information on the screen. If the information is filled out correctly and “add user” is clicked, the application creates a report with the user data and automatically saves the report to the desktop. Next, the data is saved in the database.



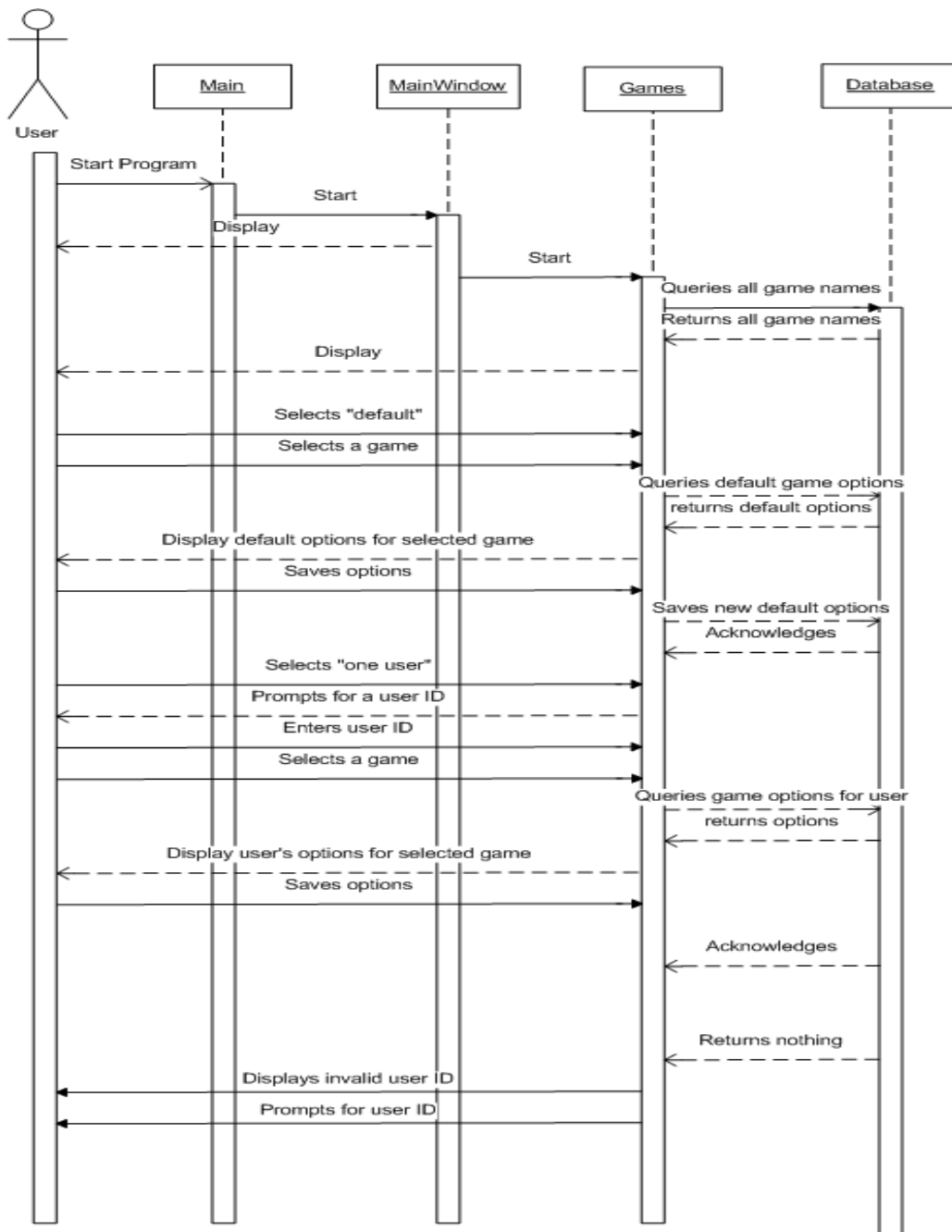
5.3.2.3 Healing Vision Edit User

Editing a patient is a very similar process to adding a patient. The exception is that the user must enter an ID to retrieve the patient information. If the ID entered does not exist, the user is prompted that the credentials are incorrect and the user must try again. If the ID does exist, the patient’s information is queried from the database and used to populate the screen. At this point the user has the option to edit any information in the UserTable and save the new information back to the database.



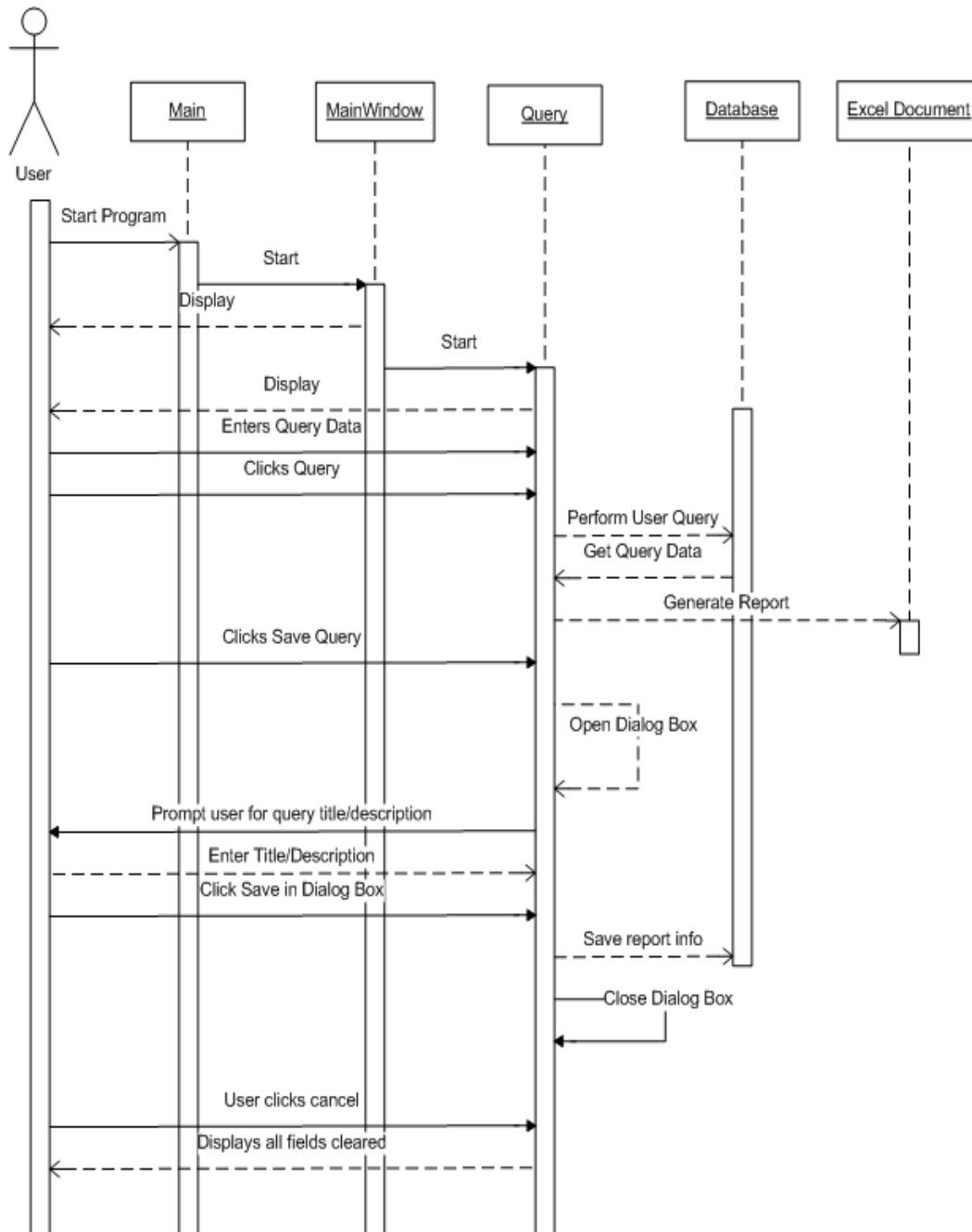
5.3.2.4 *Healing Vision Game Options*

When a user clicks on the games tab, the games screen appears. At the same time the database is queried for all the game names, and this information is used to populate the screen. Once a user clicks on a specific game and who they want the options to apply to, a box gets populated with the game options for the clicked game and selected patient group. From here, the user can edit the game options and save the changes back to the database.



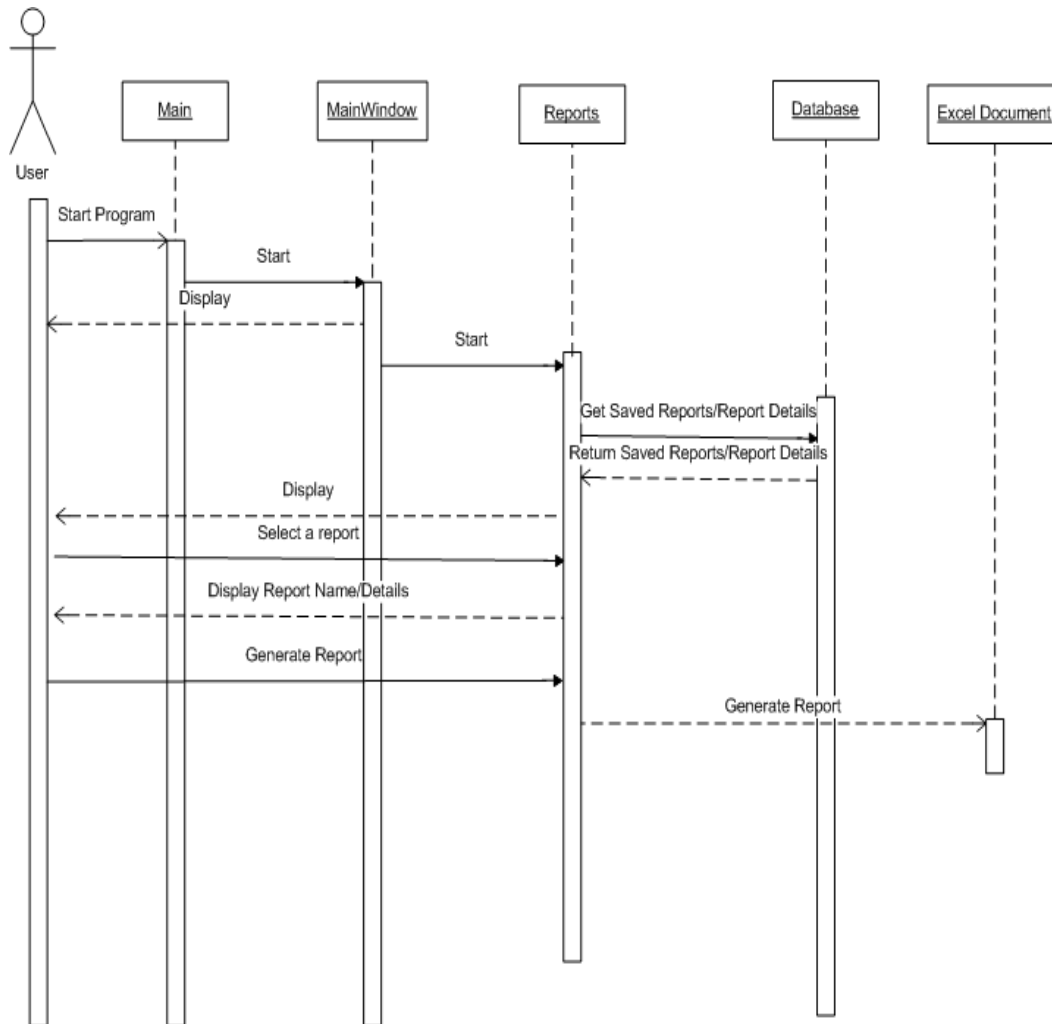
5.3.2.5 Healing Vision Query Data

Using a series of GUI components, the clinician can select attributes, patient groups, and range values for each attribute in order to generate a report. Also, in addition to generating the report, the user is able to save the Query as a report. In this case, the user clicks “Save Query” and a dialog box appears prompting the user for the query title and description. When this query is saved, it will appear in the “Reports” tab of *Healing Vision*.



5.3.2.6 *Healing Vision Generate Reports*

To generate a report, the user clicks one of the reports from a list on the screen. The title of this report and a description of this report will appear. If this is in fact the report the user wants to generate, the user clicks “Generate Report.” Finally, an excel document with the data gathered from the report query is generated.

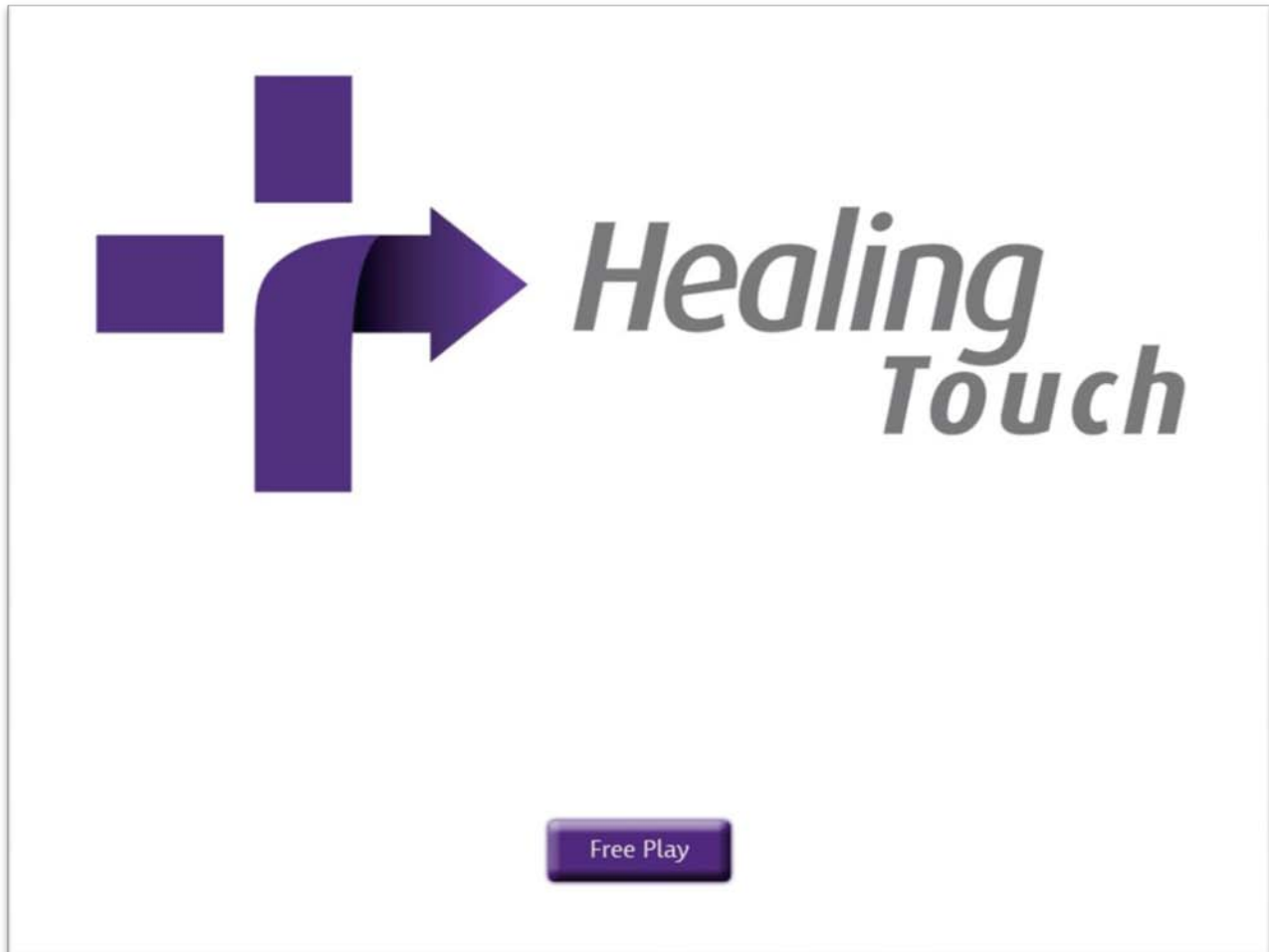


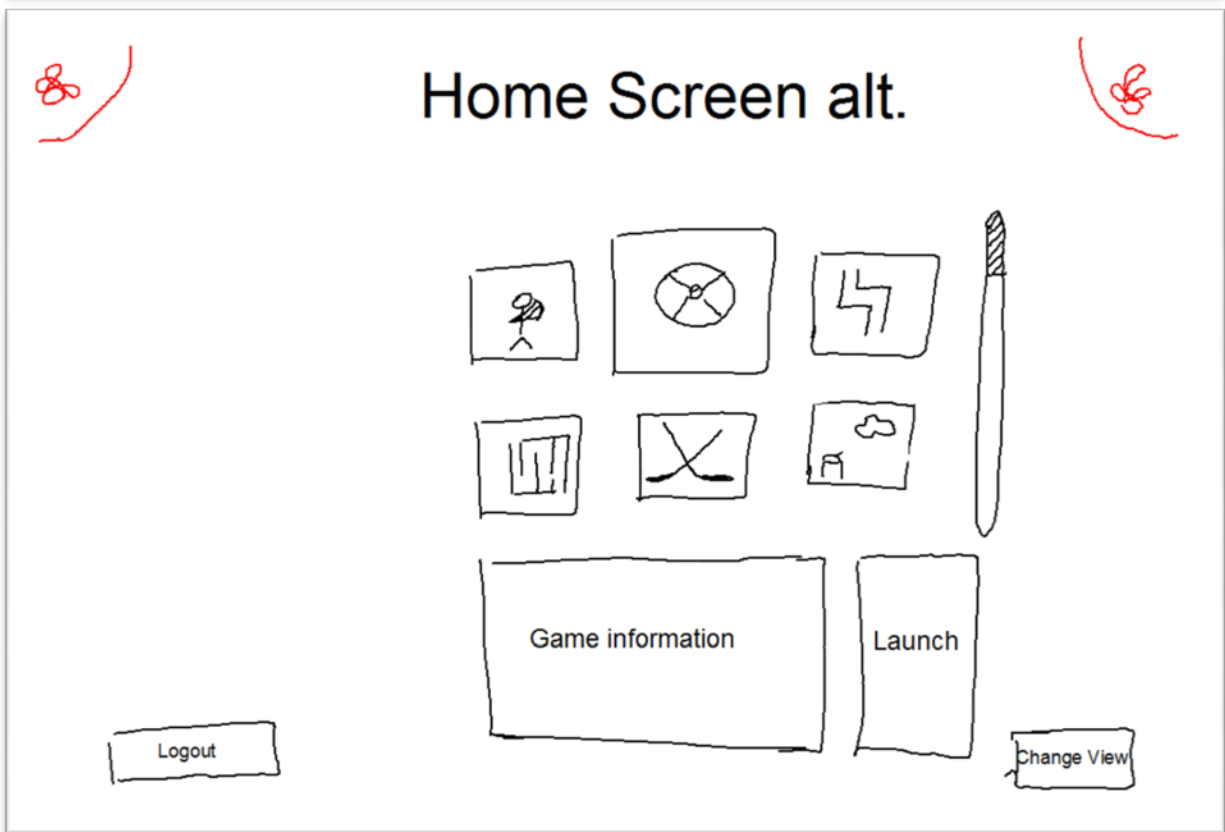
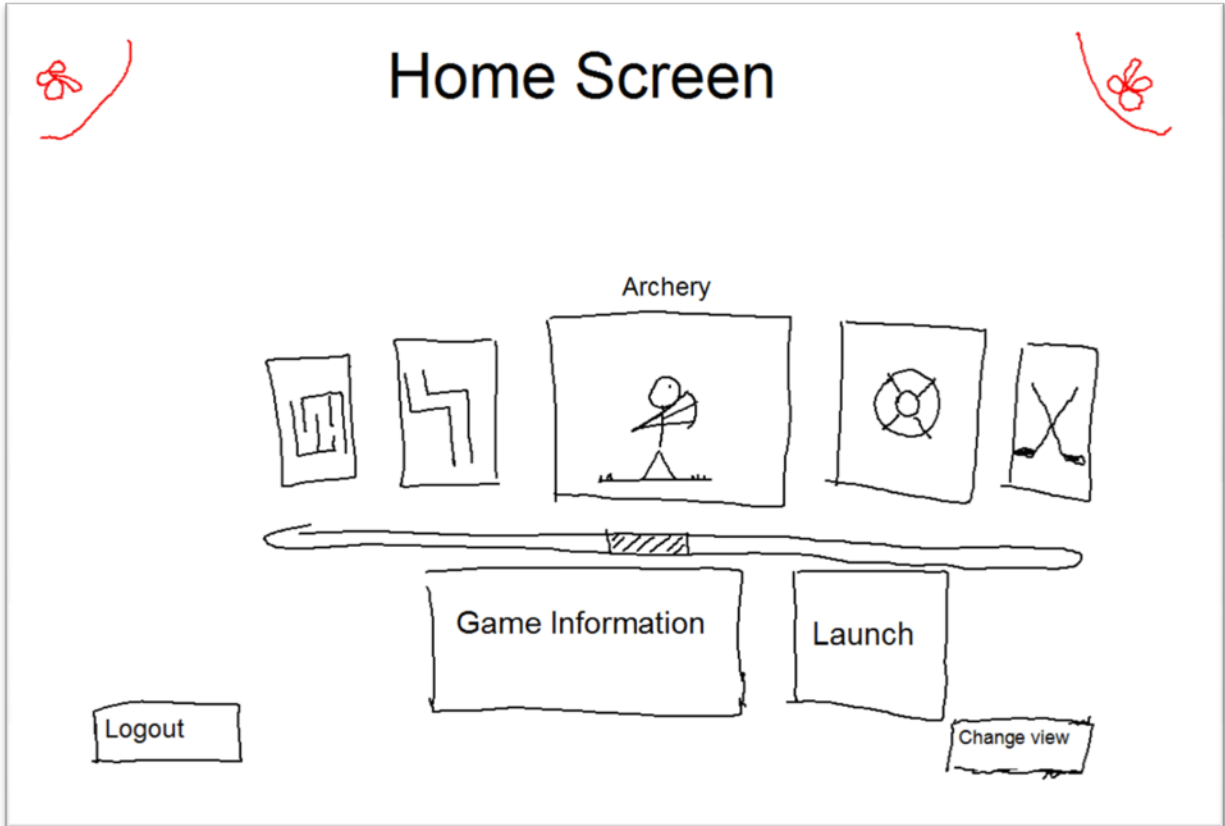
6. External Interface Architecture

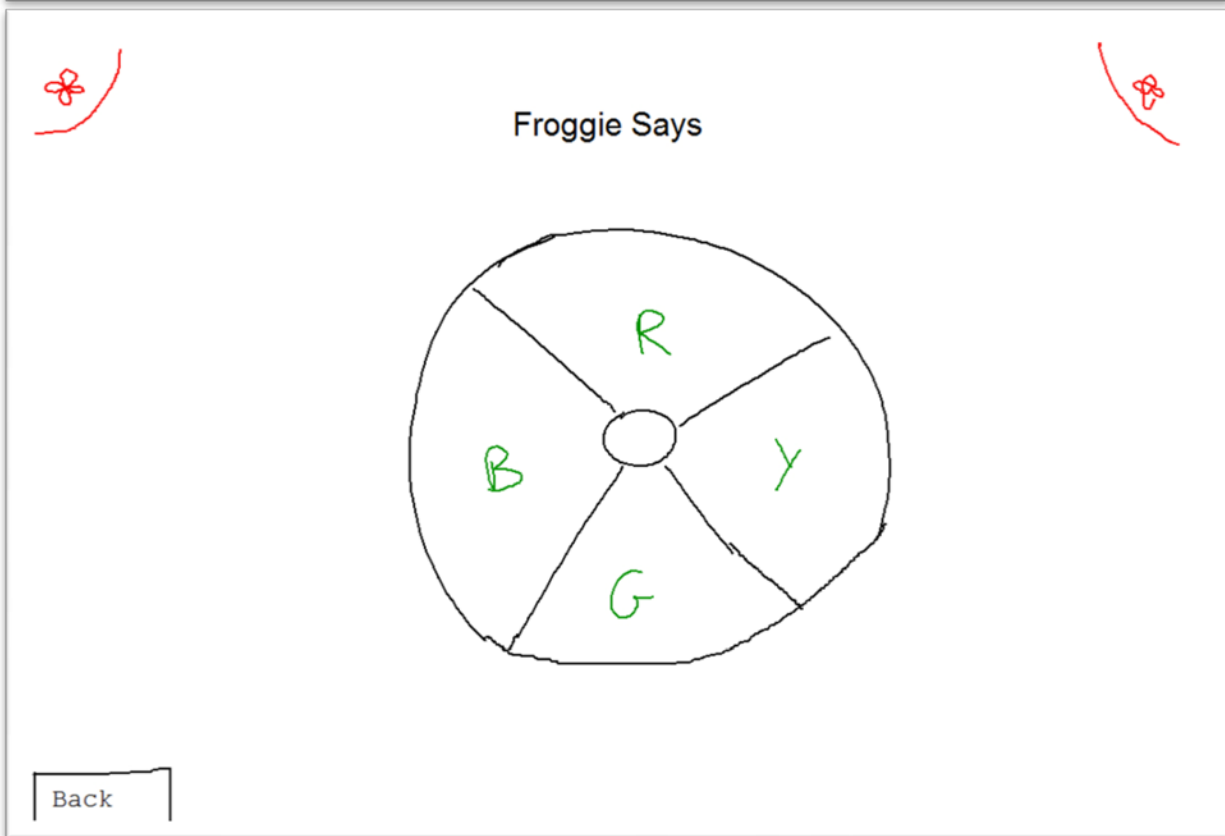
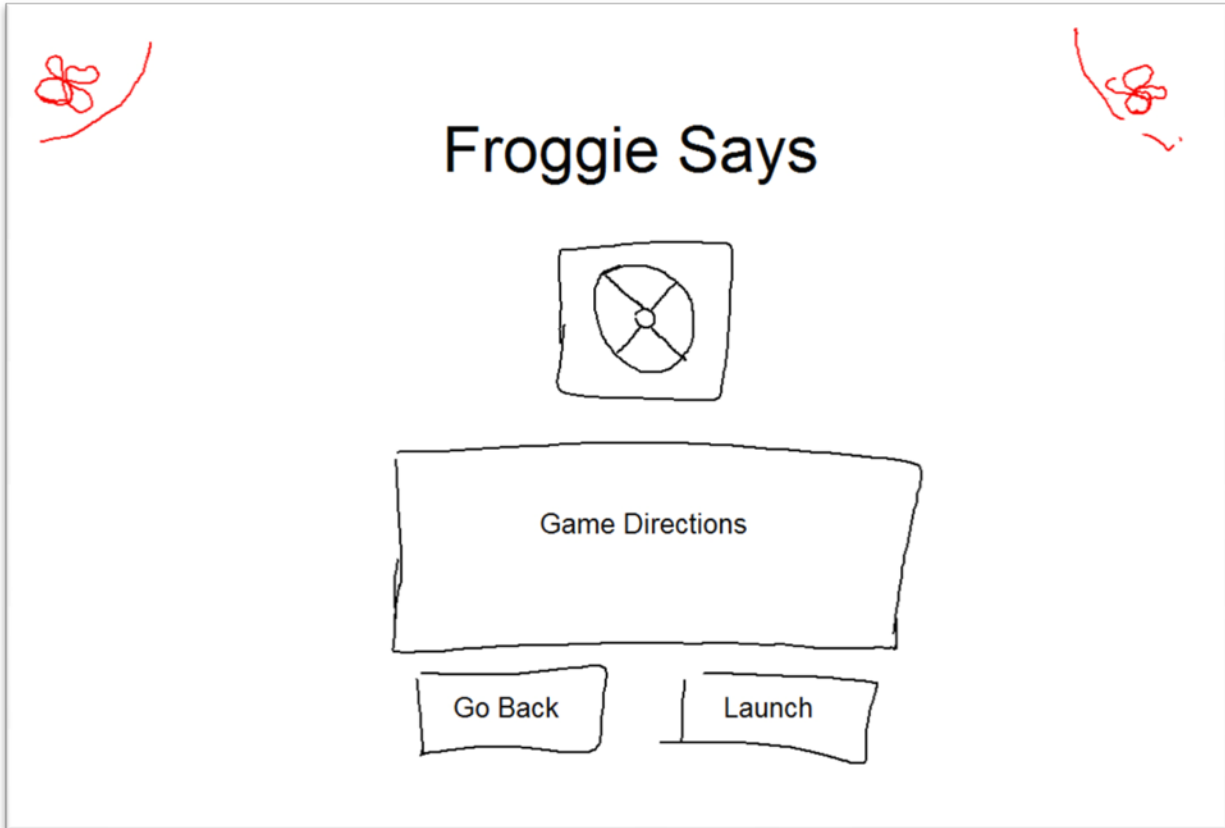
The *Healing Touch* system is designed to accommodate the addition of new games to the framework. Every game should possess its own package. A game's main class needs to extend the "Game" class in the Framework package to inherit several properties such as: a reference to the framework, an icon, a brief description, and appropriate methods.

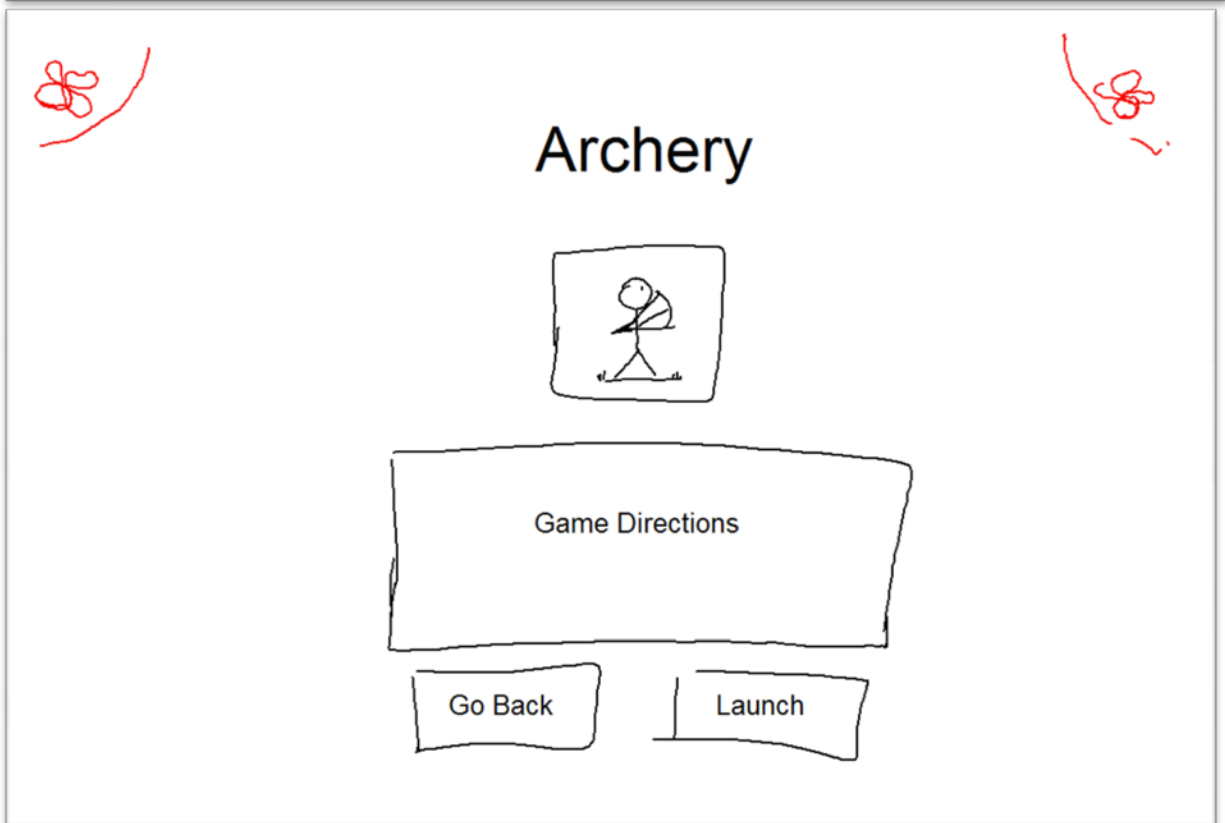
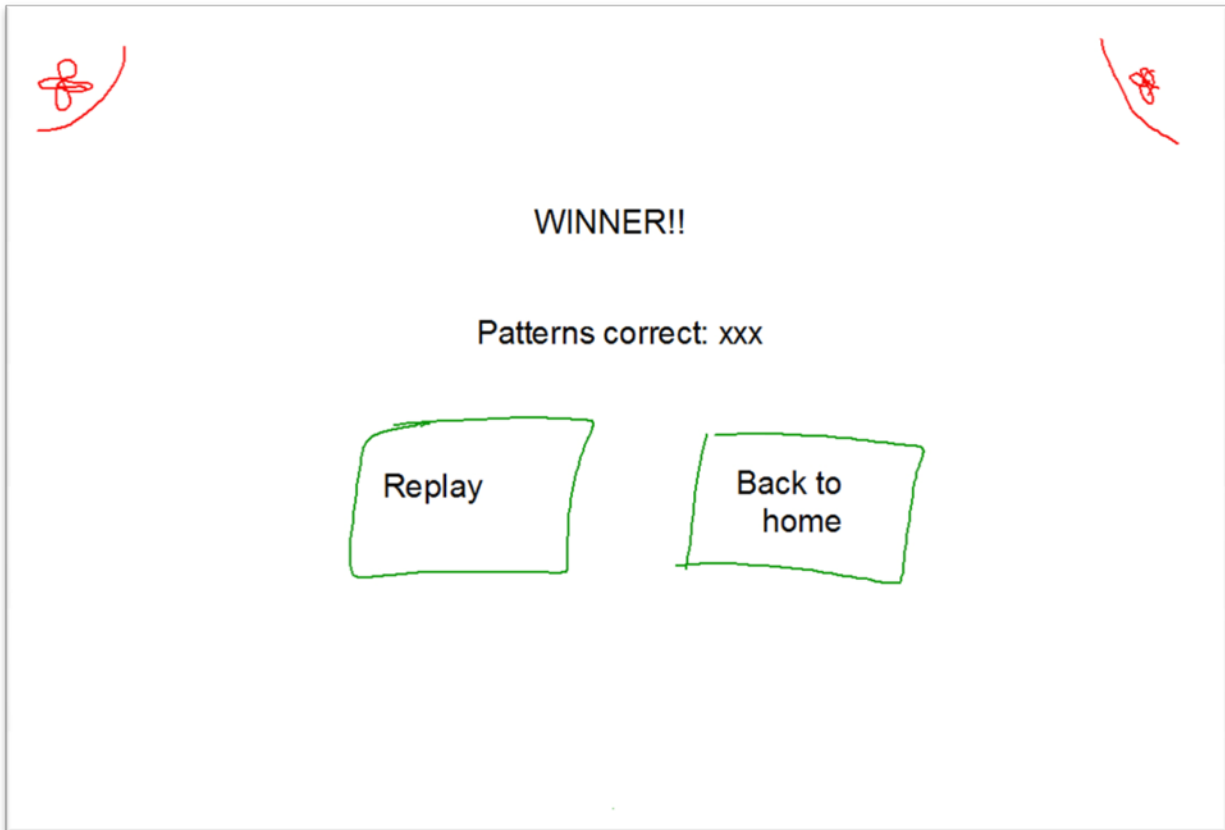
7. Human Machine Interface

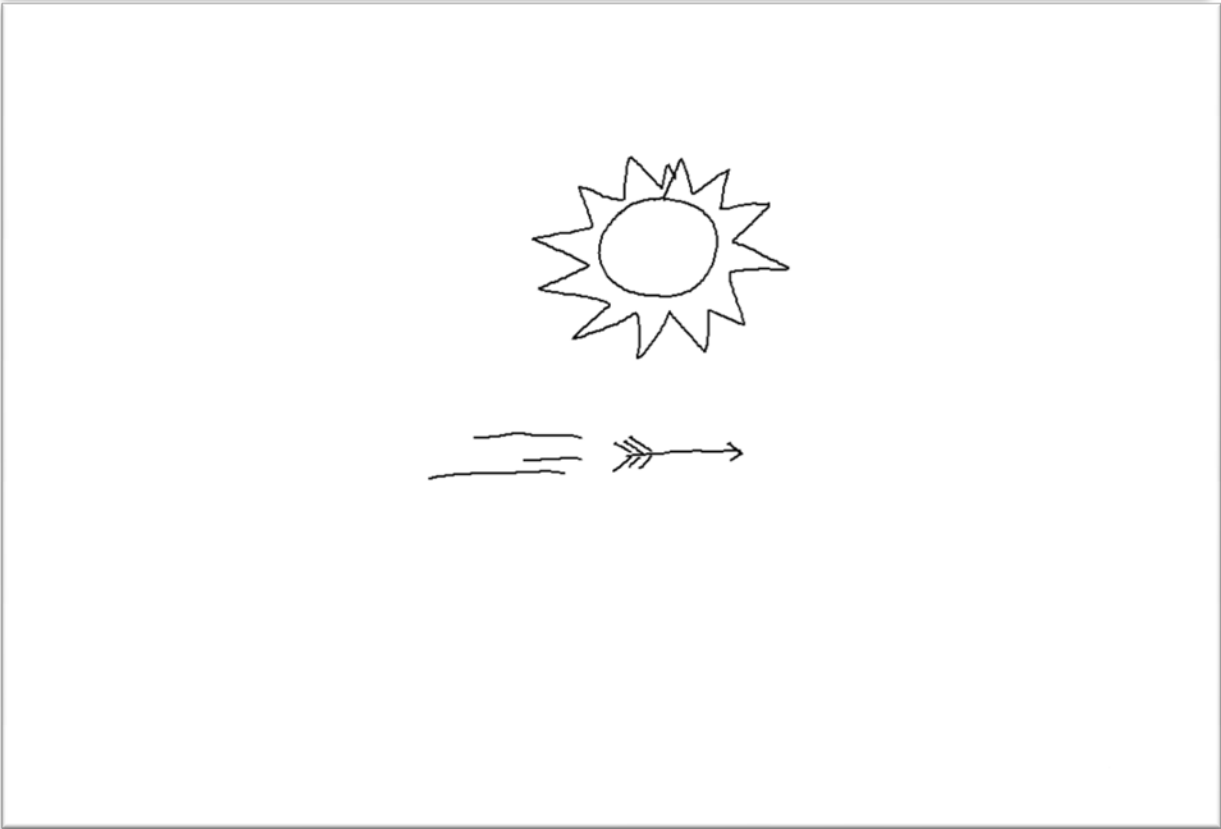
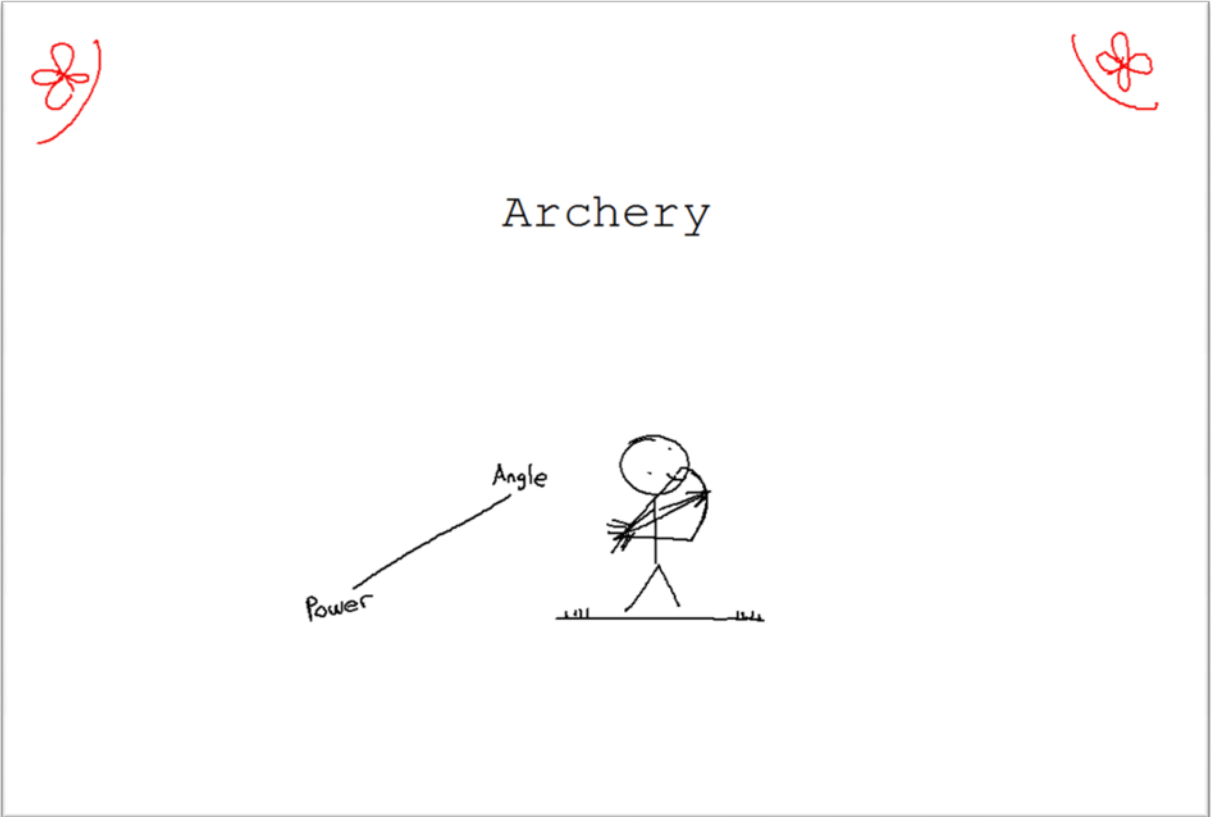
The following prototypes walk the user through the *Healing Touch* and *Healing Vision* interface. Two uses of the prototypes are to illicit system requirements from THR, *Healing Touch*'s client, and to outline the application's work flow.

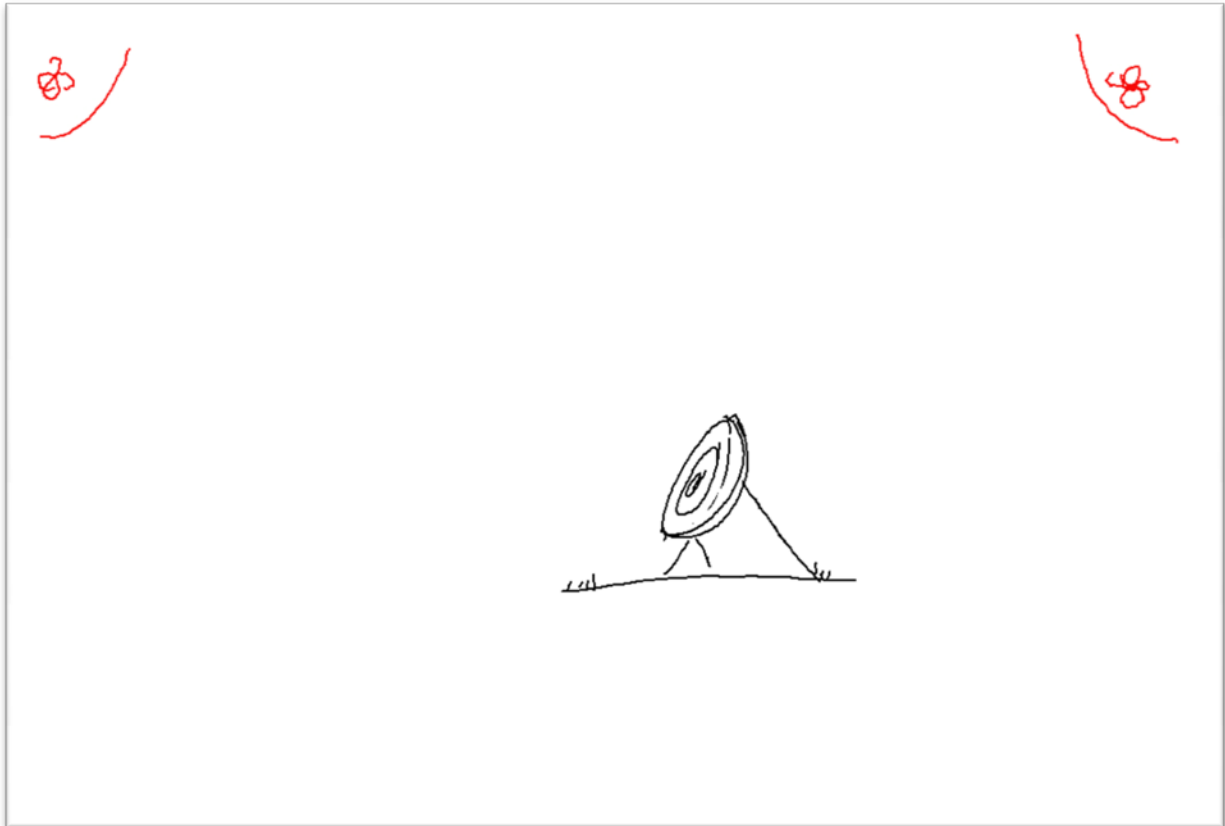


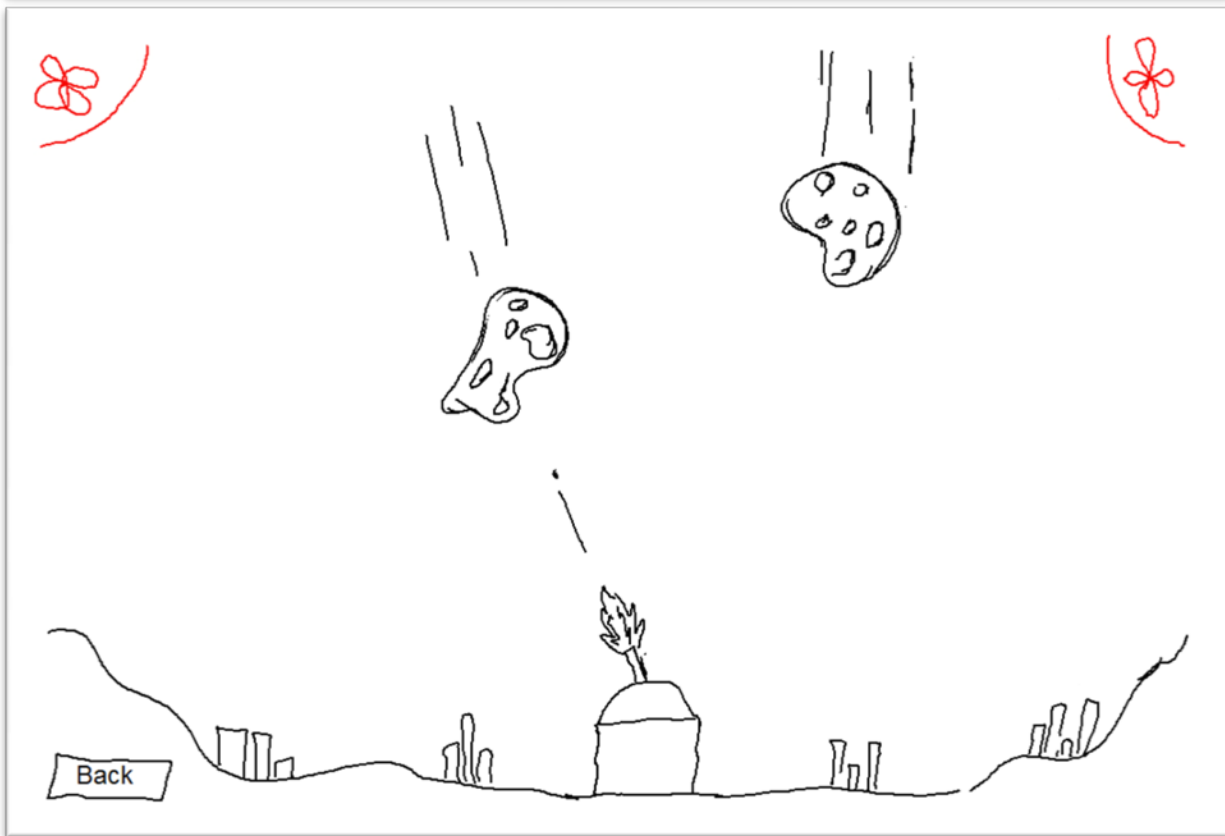
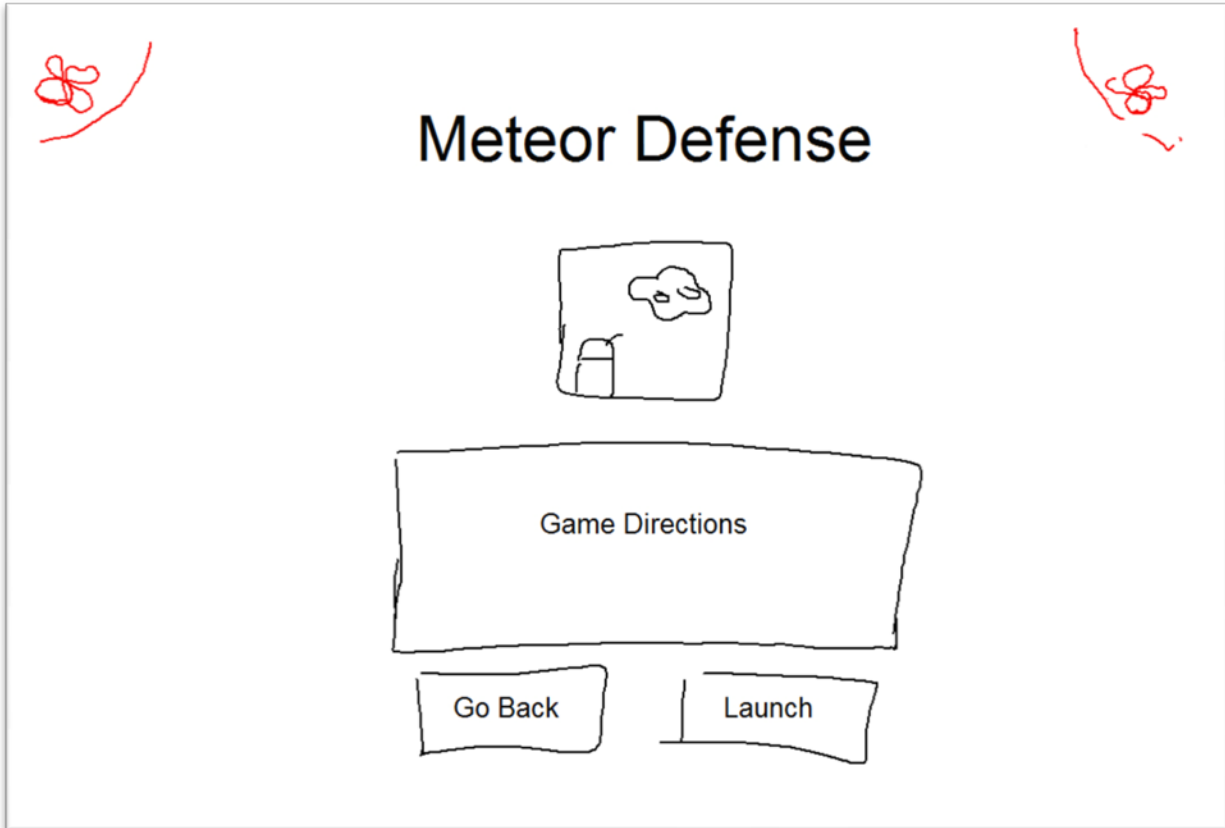


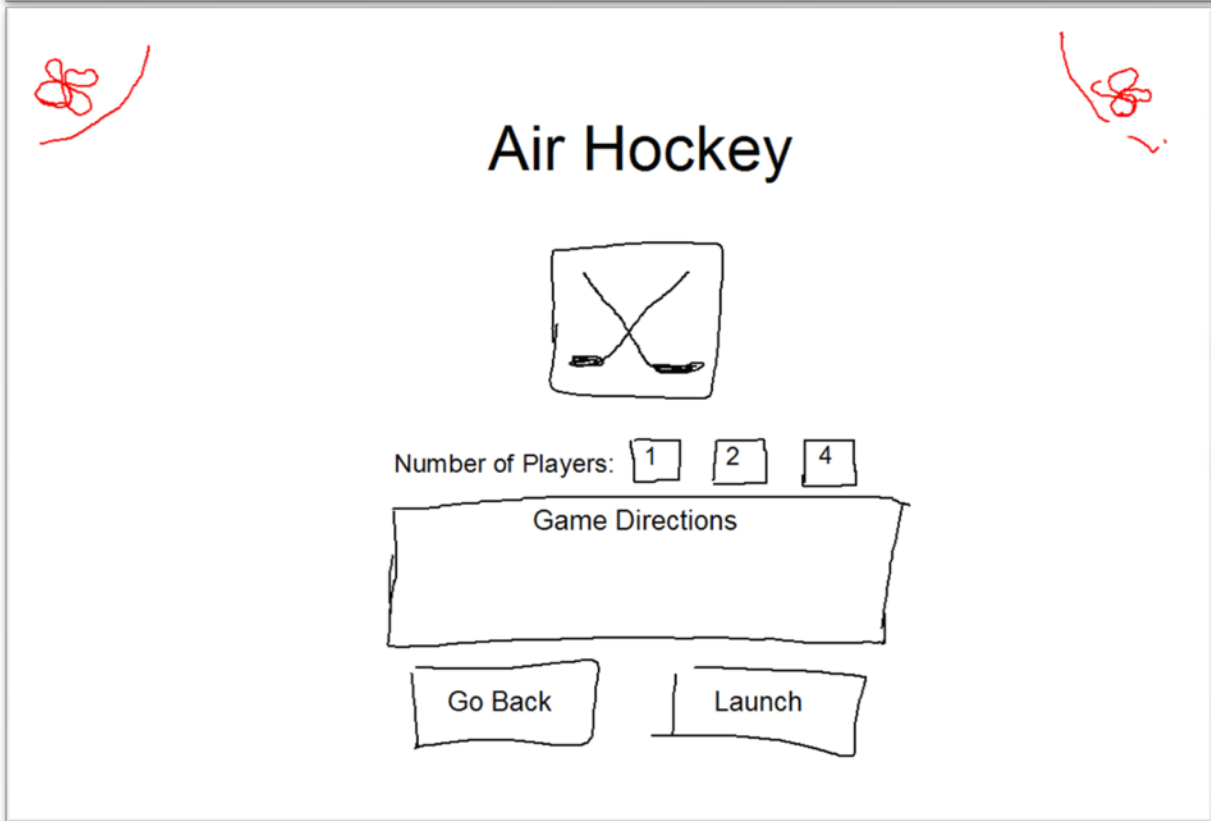
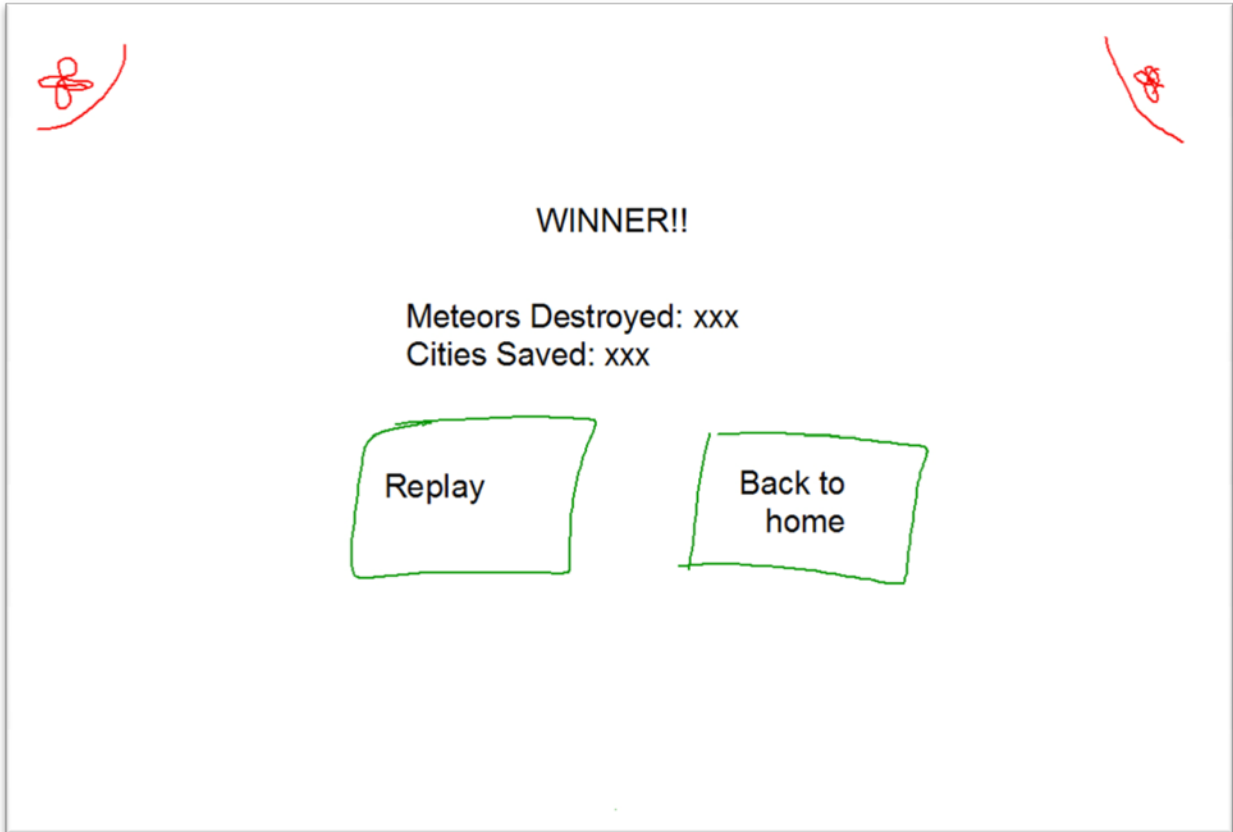


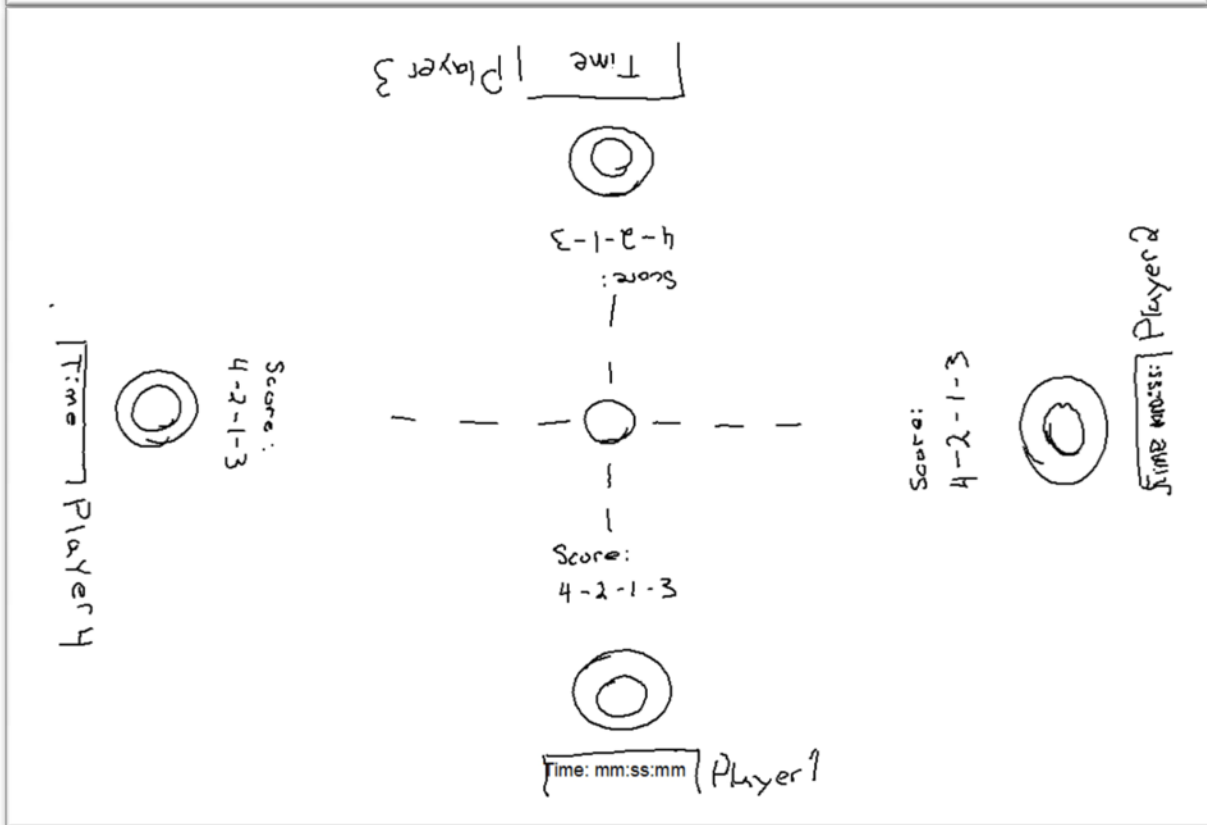
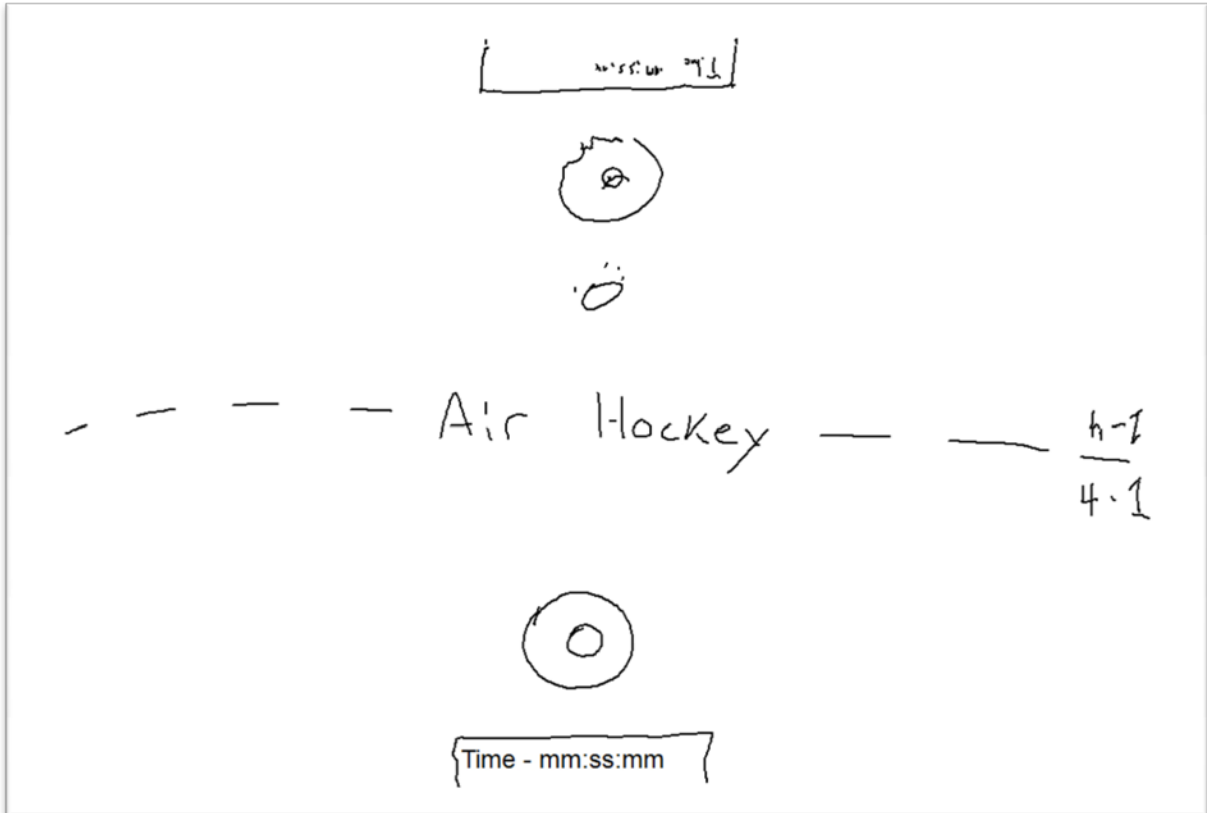


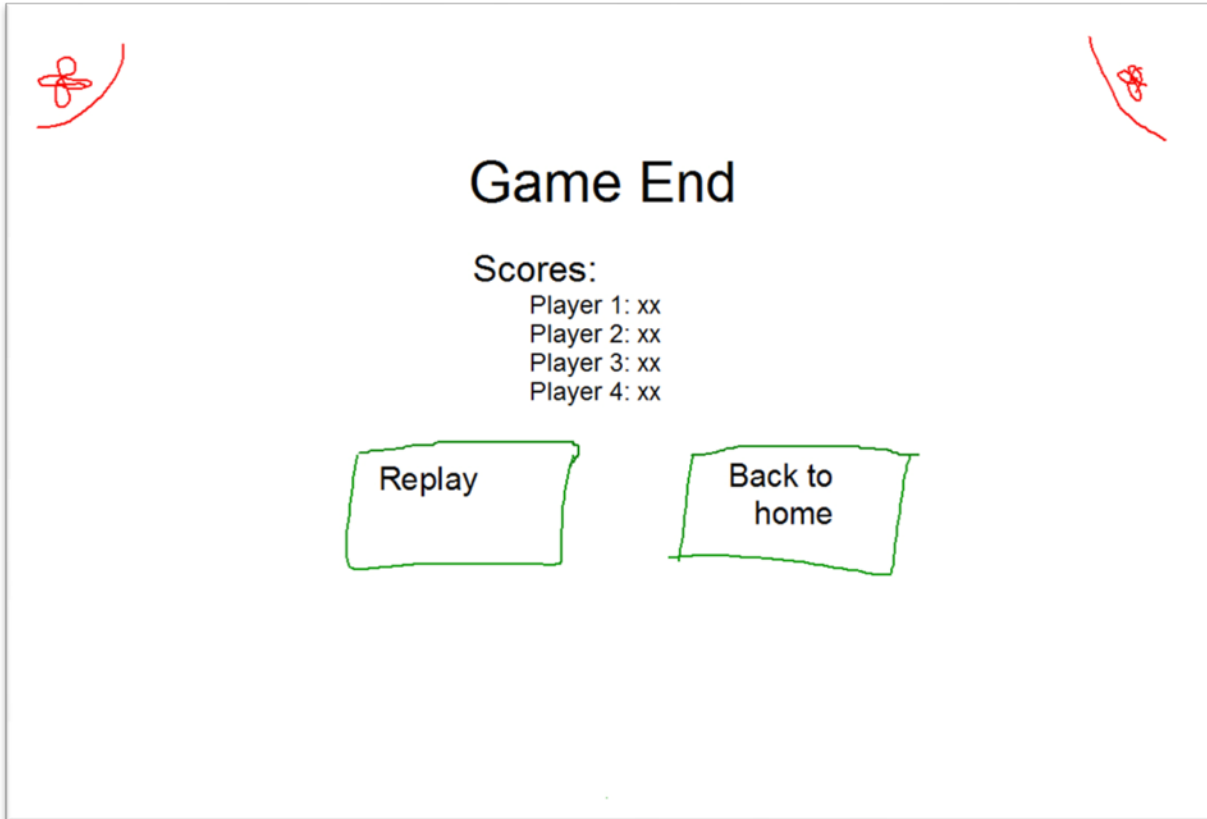


















 Healing Vision				
Add User	Edit User	Reports	Query	Games
				Login
<div style="border: 1px solid black; padding: 10px; width: fit-content; margin: auto;"><p>User Name: <input type="text"/></p><p>Password: <input type="password"/></p></div>				


 Healing Vision				
Add User	Edit User	Reports	Query	Games
				Add User
<div style="border: 1px solid black; padding: 10px; width: fit-content; margin: auto;"><p>User First Name: <input type="text"/></p><p>User Last Name: <input type="text"/></p><p>Date of Birth: <input type="text" value="04"/> <input type="text" value="2010"/></p><p>Left Handed: <input checked="" type="checkbox"/> Right Handed: <input type="checkbox"/></p><p>Color Blind: <input type="checkbox"/></p><p>Deaf: <input checked="" type="checkbox"/></p></div> <div style="border: 1px solid black; padding: 10px; width: fit-content; margin: auto; margin-top: 10px;"><p>Additional Information:</p><div style="border: 1px solid black; height: 80px; width: 100%;"></div><div style="text-align: right; margin-top: 10px;"><input type="button" value="Add User"/></div></div>				

 Healing Vision				
Add User	Edit User	Reports	Query	Games
User ID: <input type="text"/>				Edit User
				<input type="button" value="Search"/>


 Healing Vision				
Add User	Edit User	Reports	Query	Games
User ID: <input type="text" value="12355"/>				Edit User
Invalid User ID				
				<input type="button" value="Search"/>

 Healing Vision				
Add User	Edit User	Reports	Query	Games
<p>User ID: <input type="text" value="12345"/></p> <p>Date of Birth: <input type="text" value="04"/> <input type="text" value="2010"/></p> <p>Left Handed: <input checked="" type="checkbox"/> Right Handed: <input type="checkbox"/></p> <p>Color Blind: <input type="checkbox"/></p> <p>Deaf: <input checked="" type="checkbox"/></p>			<h2>Edit User</h2> <p>Additional Information:</p> <div style="border: 1px solid black; padding: 5px; min-height: 80px;">This is additional information that will help the clinician.</div> <p><input type="button" value="Update"/> <input type="button" value="Clear"/></p>	

 Healing Vision									
Add User	Edit User	Reports	Query	Games					
<table border="1"><tr><td><u>Froggie Says</u></td></tr><tr><td>Maze</td></tr><tr><td>Meteor Defense</td></tr><tr><td>Air Hockey</td></tr><tr><td>Archery</td></tr></table>		<u>Froggie Says</u>	Maze	Meteor Defense	Air Hockey	Archery	<h2>Reports</h2> <div style="border: 1px solid black; padding: 10px;"><h3>Froggie Says</h3><p>This is a description of the Froggie Says report.</p><p style="text-align: right;"><input type="button" value="Generate Report"/></p></div>		
<u>Froggie Says</u>									
Maze									
Meteor Defense									
Air Hockey									
Archery									

							
Add User	Edit User	Reports	Query	Games			
<h1>Query</h1>							
<table border="1"><tr><td>Query</td><td>Save Query</td><td>Clear</td></tr></table>					Query	Save Query	Clear
Query	Save Query	Clear					

Query Name: <input type="text"/>	
Query Description: <input type="text"/>	
<input type="button" value="Save"/>	<input type="button" value="Cancel"/>

																				
Add User	Edit User	Reports	Query	Games																
				<h2>Games</h2>																
Default Options: <input type="checkbox"/>		User Options: <input checked="" type="checkbox"/>																		
User ID: <input type="text"/>																				
<table border="1"><tr><td>Froggie Says</td><td><input type="checkbox"/></td></tr><tr><td>Maze</td><td><input type="checkbox"/></td></tr><tr><td>Meteor Defense</td><td><input type="checkbox"/></td></tr><tr><td>Air Hockey</td><td><input type="checkbox"/></td></tr></table>		Froggie Says	<input type="checkbox"/>	Maze	<input type="checkbox"/>	Meteor Defense	<input type="checkbox"/>	Air Hockey	<input type="checkbox"/>	<table border="1"><tr><td colspan="2">Game Options:</td></tr><tr><td>Sound: <input type="checkbox"/></td><td><input type="checkbox"/></td></tr><tr><td>Colors: <input type="checkbox"/></td><td><input type="checkbox"/></td></tr><tr><td>Speed: <input type="text"/></td><td><input type="text"/></td></tr></table>			Game Options:		Sound: <input type="checkbox"/>	<input type="checkbox"/>	Colors: <input type="checkbox"/>	<input type="checkbox"/>	Speed: <input type="text"/>	<input type="text"/>
Froggie Says	<input type="checkbox"/>																			
Maze	<input type="checkbox"/>																			
Meteor Defense	<input type="checkbox"/>																			
Air Hockey	<input type="checkbox"/>																			
Game Options:																				
Sound: <input type="checkbox"/>	<input type="checkbox"/>																			
Colors: <input type="checkbox"/>	<input type="checkbox"/>																			
Speed: <input type="text"/>	<input type="text"/>																			
		<input type="button" value="Save"/> <input type="button" value="Cancel"/>																		