*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Project Title: SkillCourt 5

**Team Members:**

Sean Borland

Gajenthiran Gunasegaram

**Product Owner(s)**:

Gudmundur Orn Traustason

**Mentor(s)**:

None

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*Analytics is completely changing the way sports are being played in this modern era. SkillCourt is another tool that provides a coach or player more information on performance and progression. This document outlines the process of building such a tool and its functional and non-functional requirements specified by the product owner and exemplifies the link between hardware and software.*

**Table of Contents**

**Introduction** ……………………………………………………………………………………………………………………………….. 5

Current System ………………………………………………………………………………………………………………………... 5

Purpose of New System ……………………………………………………………………………………………………………... 5

**User Stories**

Implemented User Stories ………………………………………………………………………………………………………….. 7

Pending User Stories …………………………………………………………………………………………………………..….. 10

**Project Plan**

Hardware and Software Resources ………………………………………………………………………………………….… 12

Sprints Plan …………………………………………………………………………………………………………………………. 13

*Sprint 1*  …………………………………………………………………………………………………………………………... 13

*Sprint 2*  …………………………………………………………………………………………………………………………... 13

*Sprint 3*  …………………………………………………………………………………………………………………………... 14

*Sprint 4*  …………………………………………………………………………………………………………………………... 15

*Sprint 5*  …………………………………………………………………………………………………………………………... 16

**System Design**

Architectural Patterns ………………………………………………………………………………………………………….. 20

System and Subsystem Decomposition ……………………………………………………………………………………….… 21

Deployment Diagram ………………………………………………………………………………………………………….…... 22

Design Patterns ……………………………………………………………………………………………………………….….... 22

**System Validation**  …………………………………………………………………………………………………………………….23

**Glossary**  ………………………………………………………………………………………………………………………………….37

**Appendix**  ………………………………………………………………………………………………………………………………….38

Appendix A - UML Diagrams ……………………………………………………………………………………………………. 38

*Static UML Diagrams*  ……………………………………………………………………………………………………….38

*Dynamic UML Diagrams*  …………………………………………………………………………………………………..40

Appendix B - User Interface Design ……………………………………………………………………………………….…... 52

Appendix C - Sprint Review Reports ……………………………………………………………………………………...…… 69

Appendix D - Sprint Retrospective Reports …………………………………………………………………….…………… 74

**References** ……………………………………………………………………………………………………………………...………...80

# Introduction

SkillCourt 5.0 is the most current version of the program and has been completely rebuilt and redesigned from the ground up. A completely new pad design has been constructed and the programming language has been changed from Processing to Java. The reason for the language change is to have more functionality through the availability of more libraries, an extensible and easy to understand program and finally, to have an ide independent system as Processing also requires the use of the Processing IDE which isn’t as supported as before.

## 

## Current System

SkillCourt 4.0 and all of its previous iterations are based on the Processing programming language and the Processing IDE, Java and C. The Processing programming language was designed with the engineer in mind, so certain aspects of it are simplified and as a result, very limited.

The system required the use of a Bluetooth application on an android device to start the session and a PC with the Processing IDE to run the application. A simulator was also implemented to aid in the development, its purpose allowed the programmers to see hits register and view their values.

All previous versions of SkillCourt and the current version use the Arduino Microcontroller and communicate via the system's serial port. There is a dependency relationship in the previous versions of SkillCourt as a Master/Slave relationship was implemented so the pads needs to be aware of the existence of all other pads. This is no longer the case as each Arduino and pad is its own entity which also does not restrict the performance as it is now more scalable.

The other issue with the previous system was the lack of comments which made it incredibly difficult to add features and make edits. One main goal was instant feedback which could not be implemented using the old system. Your score was calculated and displayed via the phone app after the game was over. There was no real understanding of how the statistics in the old system worked which made it difficult to try and implement this feature in the existing code.

## Purpose of New System

SkillCourt 5.0 is a complete rebuild of the system. It’s written in an entirely different language and is IDE agnostic. What was once 12 classes and 3,244 lines of code is now 4 classes and 1,014 lines of code with the added features and comments. This was a milestone in the development process as the direction was much more clear and the program easily extensible and understandable. Since there were no comments in all the previous iterations adding to the existing code proved incredibly difficult. The new system is fully commented and completely scalable.

So, the system no longer needs a separate phone application to start the session and no simulator is needed as the program is running. The idea of having a phone app is something that will be added in the future, but without the need of a PC. The user simply loads the program, starts the application, selects three game options and can begin playing within seconds. This is a vast improvement over the previous version where 9 options were needed to run the program and roughly 25-30 seconds in delays alone were hampering the performance and slowed the whole start process down.

The system has been made completely scalable and there is no longer a need for a Master/Slave relationship between the pads. Theoretically, a user is able to connect as many pads to the system as they have available COM ports. Before, each pad had to be hard coded and given an identity in order to be active. Now the user enter the number of pads they wish to use in the session, click submit and each will be connected and ready to run in a matter of seconds. With the ability to easily add any number of pads, the options for different game modes are endless.

The biggest feature that has been introduced to the program is the ability to receive instant statistical feedback and a displayed, running countdown timer. As mentioned before, in the previous version, a user and all other participants would only know their score when the session ended and there was no sense of time during the training session. Now, the user can see their score update as they hit a pad and the timer is fully displayed so that the player will know exactly how much time is remaining in the current game.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of SkillCourt. These user stories served as the basis for the implementation of the project redesign and the new features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

**User Story #717 - Create and Manage Database**

## As a user, I should be able to store and retrieve information from a database, so that all my data is accessible.

Acceptance Criteria:

1. Database should be active while code is running.
2. Cannot have more than one of the same username.
3. Username and passwords cannot exceed 25 characters.

**User Story #718 - Design a Graphical User Interface**

As a user, I would like to have a different easy to understand screens for Logging in and Creating an account.

Acceptance Criteria:

1. Users will have a visual representation of login.
2. The design must be simple and of a standard form.
3. The Create Account page will display all important fields.

**User Story #719 - Provide Instant Statistical Feedback**

## As a user, I want to be able to instantly see the change in my score throughout a game, so that I know how I am doing as I’m playing the game.

Acceptance Criteria:

1. All statistics must start at 0 when game begins
2. Scoring must change as points are earned or deducted
3. Updates must happen as close as possible to when hits made
4. Accuracy will be measured on green pad hits over total pads hit

**User Story #724 - Connect and Send a Sequence to Multiple Arduinos/Pads**

### As a User, I should be able to see the color sequence displayed on all pads, so that I know which one needs to be hit because I want more than one option when playing the game.

* Currently the system connects to one pad and displays a particular color on that one pad. The objective of this user story is to have a sequence displayed on multiple pads as opposed to just one.

Acceptance Criteria:

1. Establish a connection with each Arduino (two in this user story).
2. Edit the code so that it is able to communicate with multiple serial ports.
3. Send a sequence to multiple Arduinos.
4. Have each Arduino’s LED light up to the corresponding color that was sent.

**User Story #727 - Add a Game Setup Menu and Session Timer**

As a User, I should be able to adjust the current game settings, so that I can customize the program to my liking.

* The user must set how long they want the game to run and how many pads they will be using during that particular game session.

Acceptance Criteria:

1. Create a menu with the time and number of pad options.
2. Introduce constraints so that the user can’t enter undesirable values.
3. The values must connect with their desired methods and fields in the program.
4. Timer needs to start and end in the user specified time.
5. Timer need to run independent of the main execution thread.
6. The game session needs to end when the time expires.

**User Story #734 - Receive data from the pad and display on Console**

As a User, I should be able to have my statistics recorded during a game, so that I can see my results because I would like to record my progress.

Acceptance Criteria:

1. Establish contact with the Arduino
2. Send it a particular sequence
3. Record a hit of the pad
4. Send that hit result, in Newtons, to Java
5. Display that result in the console window

**User Story #735 - Create a Results Screen**

As a User, I should have scoring statistics so that I can gauge and improve my own personal skill as well as improved based on other player’s scores.

Acceptance Criteria:

1. Each player must have the same statistics being measured.
2. Scoring will involve gaining a point for green pad hits and losing one for red pad hits
3. Accuracy will be a measure of successful green pad hits over total hits.

**User Story #739 - Send Discrete Sequences to Each Pad**

As a User, I should be able to see different colors displayed on each pad so that a random sequence of color changes can be made on each pad.

* Last sprint the program was able to send a sequence to multiple pads, however it was the same sequence of colors displayed on each pad. The goal this time is to have each pad light up a different color.

Acceptance Criteria:

1. Each pad has it’s own communication setup.
2. A thread of control is established between each Arduino and Java.
3. The color of each pad change over time or after a pad is hit.
4. One pad must always be green in each step of the sequence.

**User Story #740 - Receive the Data from all Pads and Display it**

As a user, I should be able to have all of my pad hits recorded and stored so that I can see my final result and compare it to previous games played.

Acceptance Criteria:

1. Arduinos successfully send data to Java
2. Display results of all pads

**User Story #770 - Display a Countdown Timer**

As a Player, I should have the ability to see how much time I have remaining in my current game session.

Acceptance Criteria:

1. Timer must be synchronized with the game session
2. The timer must close when the game ends.

## Pending User Stories

**User Story #843 - Filter A Job**

As a staff user, I need to be able to filter the list of available jobs based on a search criteria so that I can better assess which job is best for me.

Acceptance Criteria:

1. Be able to enter a search criteria.
2. Have the list of jobs narrowed down to match that search criteria.

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

**Arduino**

Arduino is a hardware and software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

**AdaFruit LED strips**

A series of LED strips are used which lights up the pad a specific color. Each LED strip is fitted with 120 LED lights and is roughly 4 feet long

**FlexiForce Force Sensor**

A force sensor is attached to the center of each pad which current detects whether or not a pad has been it. In the future, the sensor will have the ability to detect the exact amount of force exerted and the velocity in which the ball traveled.

**Java**

Java is the core language used. Some featured classes and libraries to highlight would be SerialPort, rxtxComm, Runnable(Threads), and Sockets will be used in the future for wireless communication.

**C**

The Arduino MicroController runs completely on C with it’s own Serial port package, SPI.h and LED adjuster AdaFruit.h

**Mingle**

Mingle was used as a planning and management tool for the various agile development processes.

**MySQL**

MySQL was chosen as the relational database because of the open source nature and the group’s familiarity with it.

**Github**

Github was used to store and manage the source code.

**Gmail**

Gmail was used for communication.

**Google Drive**

Google Drive was used to store project documents and to transfer data between the group members.

## 

## 

## Sprints Plan

### Sprint 1

(05/23/2016 - 06/03/2016)

**User Story #717 - Create and Manage Database**

***Tasks***

* Connect to database
* Verify user info
* Store user info in database

***Acceptance Criteria***

1. Database should be active while code is running.
2. Cannot have more than one of the same username.
3. Username and passwords cannot exceed 25 characters.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

**User Story #718 - Design a Graphical User Interface**

***Tasks***

* Design login page
* Design Create Account page
* Design main game menu

***Acceptance Criteria***

1. Users will have a visual representation of login.
2. The design must be simple and of a standard form.
3. The Create Account page will display all important fields.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

### Sprint 2

(06/06/2016 - 06/17/2016)

**User Story #724 - Connect and Send Sequence to Multiple Arduinos**

***Tasks***

* Add button with navigation for creating a job in the employer home screen.
* Create a form for the user to enter job information.
* Create and upload the job on form submission.

***Acceptance Criteria***

1. Understand how COM ports are seen through Java
2. Initialize method changed to see more pads

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

**User Story #734 - Receive data from the pad and display on console**

***Tasks***

* Research Serial Port communication
* Capture feedback from Arduino Code
* Display feedback on Java Console

***Acceptance Criteria***

1. Establish contact with the Arduino
2. Send it a particular sequence
3. Record a hit of the pad
4. Send that hit result, in Newtons, to Java
5. Display that result in the console window

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

### Sprint 3

(06/20/2016 - 07/01/2016)

**User Story #739 - Send discrete sequences to each pad**

***Tasks***

* Alternative way to send colors without Sequence files.
* Connect to multiple Arduinos using an array.
* Create way to randomize sequence.
* Light effects to show game starting and ending.
* Add a timed loop for colors to change.

***Acceptance Criteria***

1. Each pad has it’s own communication setup.
2. A thread of control is established between each Arduino and Java.
3. The color of each pad change over time or after a pad is hit.
4. One pad must always be green in each step of the sequence.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

**User Story #727 - Add a game setup menu and session timer**

***Tasks***

* Alternative way to send colors without Sequence files.
* Connect to multiple Arduinos using an array.
* Create way to randomize sequence.
* Light effects to show game starting and ending.
* Add a timed loop for colors to change.

***Acceptance Criteria***

1. Each pad has it’s own communication setup.
2. A thread of control is established between each Arduino and Java.
3. The color of each pad change over time or after a pad is hit.
4. One pad must always be green in each step of the sequence.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

### 

### Sprint 4

(07/04/2016 - 07/15/2016)

**User Story #740 - Receive the data from all pads and display it**

***Tasks***

* Establish individual communication between each Arduino and Java.
* Record a hit on each pad/Arduino.
* Each Arduino sends it result to Java.
* Store the results in a thread-safe array/arraylist.
* The final results are tallied and displayed on the console.

***Acceptance Criteria***

1. Arduinos successfully send data to Java
2. Display results of all pads

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

**User Story #719 - Provide instant statistical feedback**

***Tasks***

* Research threading concepts
* List solutions for threading problems
* Create Scoreboard Jframe
* Implement threading to scoring
* Fix threading issues

***Acceptance Criteria***

1. All statistics must start at 0 when game begins
2. Scoring must change as points are earned or deducted
3. Updates must happen as close as possible to when hits made
4. Accuracy will be measured on green pad hits over total pads hit

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

### Sprint 5

(07/18/2016 - 07/29/2016)

**User Story #770 - Display a countdown timer**

***Tasks***

* Research how to have a running and changing timer.
* Synchronize the timer and the current game session.
* Create a separate JFrame for the timer.
* Add some design for visual pleasure.

***Acceptance Criteria***

1. Timer must be synchronized with the game session
2. The timer must close when the game ends.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

**User Story #735 - Create Results Screen**

***Tasks***

* Create scoring class
* Come up with statistics to measure
* Create play again functionality
* Create new game functionality
* Create upload score functionality
* Create View Previous score functionality

***Acceptance Criteria***

1. Each player must have the same statistics being measured.
2. Scoring will involve gaining a point for green pad hits and losing one for red pad hits
3. Accuracy will be a measure of successful green pad hits over total hits.

***Modeling***

Refer to UML diagrams in Appendix A that were created or modified to model the functionality that will be implemented in this sprint.

# 

# 

# System Design

SkillCourt is heavily influenced by the hardware as well as the needs of the modern day soccer player and coach. With a series of fast, high interval training sessions, the devices must run fast and accurate and the code must be efficient so not to affect performance and scalability. SkillCourt is divided into different sub-systems. Modularly developed to allow the re-use and preprogramming of needed details in later development.

## 

## 

## Architectural Patterns

The main architectural pattern used for this project is Model View Controller (MVC) and Layered design approach. This involved the user interacting with a start menu which sets all the necessary initial variable and also physical interaction with the pad which is layered in its approach from recording the hit to transmitting the data to and from the systems.

## 

## System and Subsystem Decomposition

There are three systems at work, each divided further into their own subsystems. The three main systems are the Microcontroller, the Java controller and the Database.

The Arduino microcontroller is subdivided into a section that handles the communication of the pads to and from the system, a section in charge of regulating the LEDs and a section that records the readings from the force sensors. At the moment, the communication is handled exclusively through the use of serial ports. Next, the LEDs are controlled by methods that send readings to the individual pins on the boards that correspond to the instructions received from the Java controller. Finally, the force sensor is controlled by methods that record the various voltages received by each pin on the arduino and this information is reported back to the Java controller to be stored.

The Java controller is divided into 4 subclasses, each handling a specific aspect of the program in a highly cohesive manner. The first is the SkillCourt.java class which contains the main method and runs the entire system. Here the colors change after each successive hit on a pad and the information is communicated to all the necessary classes. Next is the Statistics.java class, this subsystem is in charge displaying a live screen with various statistical data and the timer. All the necessary values needed to record and update the score are in this class and communicate that information to all other classes. Next is the Start\_menu.java class, this class consists of a view, which gives the user the ability to set the number of pads and the desired game time. Once submitted, the necessary fields and variable are set and the game can begin. Finally, the Arduino.java class, which is the most complex of the four classes, is where all the communication between the Arduino and the java system take places. The communication is handled via a serial port and all incoming and outgoing data is controlled by an independent thread that monitors the serial port, polling for all incoming transmissions.

The final system is the Model, it is in charge of all communication to and from the database. At present, the database is local to perform all the test needed but eventually, next semester, a fully remote database will be implemented in order to easily deploy the system and store all relevant data.

Subsystems.pngFigure 1 - Subsystem Decomposition

## 

## 

## 

## Deployment Diagram

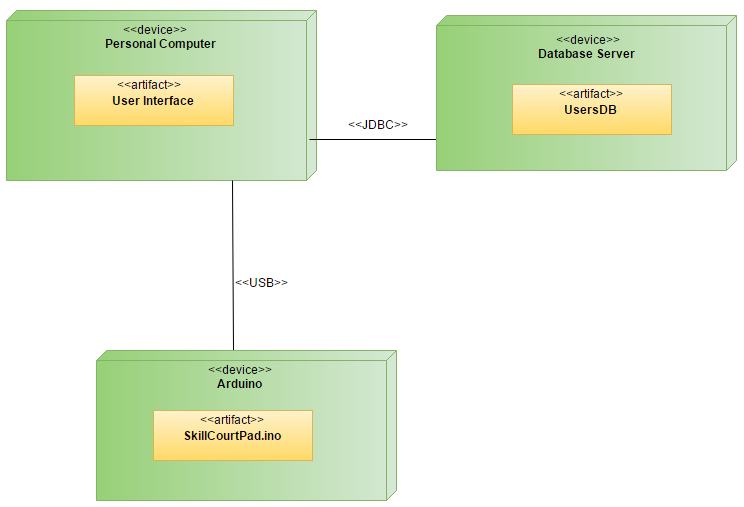


Figure 2 - Deployment Diagram

## 

## 

## Design Patterns

The design pattern used in the system is the Adapter pattern. The Adapter pattern is used mainly to encapsulate existing components and reuse them as needed. The SerialEventListener interface is a good example. This interface has methods to specifically handle traffic to and from the serial ports. So rather than create a method from scratch to monitor the serial port for incoming data or creating a method that sends bytes of data to a microcontroller, the SerialEvent class handles each of these situations. With the introduction of new components, the various serial classes provided in the java library aid in handling of various operations between software and hardware. Creating specific methods or classes to handle each and every device in its own individual way is time consuming and inefficient, so using the Adapter pattern will expedite the development process when dealing with software to hardware communication.

# System Validation

In this section, all of the test cases that were down to validate our system are outlined. All of the testing was done manually, following the input requirements of the test cases.

**User Story #717 - Create and Manage Database**

System Tests

## Test Case ID - CreateAccount\_001(Sunny Day)

Purpose:

* To test that an account is properly added to the database once the user clicks create account and submits the right information

Precondition:

* The user has the application active
* The user has no account in the database
* The user has clicked on the create account button

Input:

* The user fills out all required fields.
* The user clicks the done button.

Expected Output:

* The system will add the account to the database

## Test Case ID - CreateAccount\_002(Rainy Day)

Purpose:

* To test that the system rejects the account creation if required/correct information is not entered during the inputting of account information.

Precondition:

* The user has the application active
* The user has no account in the database
* The user has clicked the create account button.

Input:

* The user does not fill out one or more of the required fields.
* The user clicks the done button.

Expected Output:

* The system will not add the account and instead return an error stating that the operation cannot be completed.

## Test Case ID - Login\_001(Sunny Day)

Purpose:

* To test that a person who has created an account can successfully login to their account.

Precondition:

* The user has the application running
* The user has an account in the database
* The user has has clicked the login button

Input:

* The user fills out their username and password
* The user clicks the done button.

Expected Output:

* The system will display a successfully logged in message and take user to the create game screen

## Test Case ID - Login\_002(Rainy Day)

Purpose:

* To test that a person who has not created an account cannot login to their account.

Precondition:

* The user has the application running
* The user has no account in the database
* The user has has clicked the login button

Input:

* The user fills out their username and password
* The user clicks the done button.

Expected Output:

* The system will display a message that the account does not exist in the database

## Test Case ID - ForgotPassword\_001(Sunny Day)

Purpose:

* To test that a person who owns an account can successfully retrieve a forgotten password.

Precondition:

* The user has the application running
* The user has an account in the database
* The user has has clicked the forgot password button

Input:

* The fills out the text field with their username

Expected Output:

* The system will display a message that with the password associated with the given username

**User Story #724 - Connect and send a sequence to multiple pads/arduinos**

## Test Case ID - SendColorChange\_001(Sunny Day)

Purpose:

* To test if the Java code can send an instruction for the arduino to change colors.

Precondition:

* The Arduinos are connected and the proper source code uploaded to them

Input:

* The Java code sends an instruction using an int parameter matching to the int associated with the color in the Arduino code.

Expected Output:

* The Arduino LED matches the color instruction given by Java

## 

## Test Case ID - ConnectArduinos\_001(Sunny Day)

Purpose:

* To test that the Java code can successfully connect to each Arduino that is plugged into a valid USB port

Precondition:

* The Arduino code is uploaded to each Arduino
* Each Arduino is plugged into a USB port recognized by Java

Input:

* Array of Arduinos that are currently connected

Expected Output:

* The system will display a console message for each successful connection.

**User Story #734 - Receive data from pad and display on Console**

## Test Case ID - ReceiveData\_001(Sunny Day)

Purpose:

* To test that the Arduino is sending data that can be interpreted and utilized by Java.

Precondition:

* The Arduino code is uploaded to each Arduino
* Some sort of data or instruction is sent to Arduino

Input:

* Send a color instruction to Arduino

Expected Output:

* Returns a valid result such as a print statement of the value returned from Arduino.

**User Story #739 - Send discrete sequences to each pad**

## Test Case ID - SendIndividualData\_001(Sunny Day)

Purpose:

* To test that each Arduino can receive a separate instruction and react accordingly at the same time

Precondition:

* The Arduino code is uploaded to each Arduino
* The Java code is running and sending data to the Arduino

Input:

* The send color change instruction is sent using different colors to different Arduinos

Expected Output:

* Each Arduino displays the color associated with its own instruction

**User Story #727 - Add a game setup menu and session timer**

## Test Case ID - CreateGame\_001(Sunny Day)

Purpose:

* To test that a game can be successfully started using the desired amount of pads and game duration

Precondition:

* The Arduino code is uploaded to each Arduino
* The Java code is running

Input:

* Integer representing number of pads and game duration

Expected Output:

* The system will display the scoreboard and begin the game after a brief time

## Test Case ID - CreateGame\_002(Rainy Day)

Purpose:

* To test that the game will not start unless the timer and number of pads are selected

Precondition:

* The Arduino code is uploaded to each Arduino
* The Java code is running

Input:

* The user does not select any values for number of pads and game duration, passing 0 as the int for the method call

Expected Output:

* The JFrame will return an error message to select a number of pads and game duration

**User Story #740 - Receive the data from all pads and display it**

## Test Case ID - ReceiveHitData\_001(Sunny Day)

Purpose:

* To test that the force sensors attached to each Arduino sends the appropriate data back to Java

Precondition:

* The Arduino code is uploaded to each Arduino
* The game is started and running

Input:

* The force sensor is hit to cause activity

Expected Output:

* The sensor returns an integer value representing force of the hit received to Java

## Test Case ID - InstantColorChange\_001(Sunny Day)

Purpose:

* To test that the colors swap instantly when one of the pads is hit

Precondition:

* The Arduino code is uploaded to each arduino
* The game is started and running

Input:

* The force sensor registers a hit on one of the pads

Expected Output:

* The colors change to a new set of colors where only one of the pads is green

**User Story #719 - Provide instant statistical feedback**

## Test Case ID - UpdateScore\_001(Sunny Day)

Purpose:

* To test that the scores being measured in the statistics class are updating properly during a game.

Precondition:

* The game is started and running
* The scoreboard is loaded and appeared

Input:

* A pad is hit

Expected Output:

* The scores are updated based on which pad is hit

**User Story #770 - Display a Countdown Timer**

## Test Case ID - CountdownClock\_001(Sunny Day)

Purpose:

* To test that a clock with the time remaining is properly displayed throughout a game

Precondition:

* The game is started and running

Input:

* The game time selected during game creation

Expected Output:

* A display of the time clocking down in real time on the scoreboard

**User Story #735 - Create a results screen**

Test Case ID - ShowResultsAfterGame\_001(Sunny Day)

Purpose:

* To test that accurate data is represented of the player’s performance after each game

Precondition:

* A game has been completed

Input:

* The values for each statistic after a game has ended

Expected Output:

* Display all scores measured from the game

Test Case ID - UploadScore\_001(Sunny Day)

Purpose:

* To test that scores can be uploaded to the database after a game is played.

Precondition:

* A game has been completed
* Results screen is displayed

Input:

* The user clicks Upload Score button

Expected Output:

* Displays a message saying successfully uploaded scores

Test Case ID - ViewPreviousScore\_001(Sunny Day)

Purpose:

* To test that a player can view their previous best score in the database

Precondition:

* A game has been completed
* Results screen is displayed

Input:

* The user clicks the View Score button

Expected Output:

* Displays the previous scores of the user stored in database

# 

# 

# 

# 

# Glossary

**Arduino:** Arduino is a hardware and software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

**Force Sensor:** FlexiForce force sensors can measure force between almost any two surfaces and are durable enough to stand up to most environments. Our sensors are available off-the-shelf for prototyping or can be customized to meet the specific needs of your product design and application requirements.

**LED:** is a light-emitting diode (LED) product which is assembled into a lamp (or light bulb) for use in lighting fixtures. LED lamps have a lifespan and electrical efficiency which are several times longer than incandescent lamps, and significantly more efficient than most fluorescent lamps, with some chips able to emit more than 300 lumens per watt.

**Microcontroller:** (or MCU, short for microcontroller unit) is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

**Processing:** is an open source computer programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks.

**Serial Port:** is a serial communication interface through which information transfers in or out one bit at a time (in contrast to a parallel port).[1] Throughout most of the history of personal computers, data was transferred through serial ports to devices such as modems, terminals and various peripherals. While such interfaces as Ethernet, FireWire, and USB all send data as a serial stream, the term "serial port" usually identifies hardware more or less compliant to the RS-232 standard, intended to interface with a modem or with a similar communication device.

**View Controllers**: view controllers are the classes that interact with the elements on a storyboard.

Appendix

Appendix A - UML Diagrams

### Static UML Diagrams

Story718_Class_Diagram.png

Figure - Login and Create account

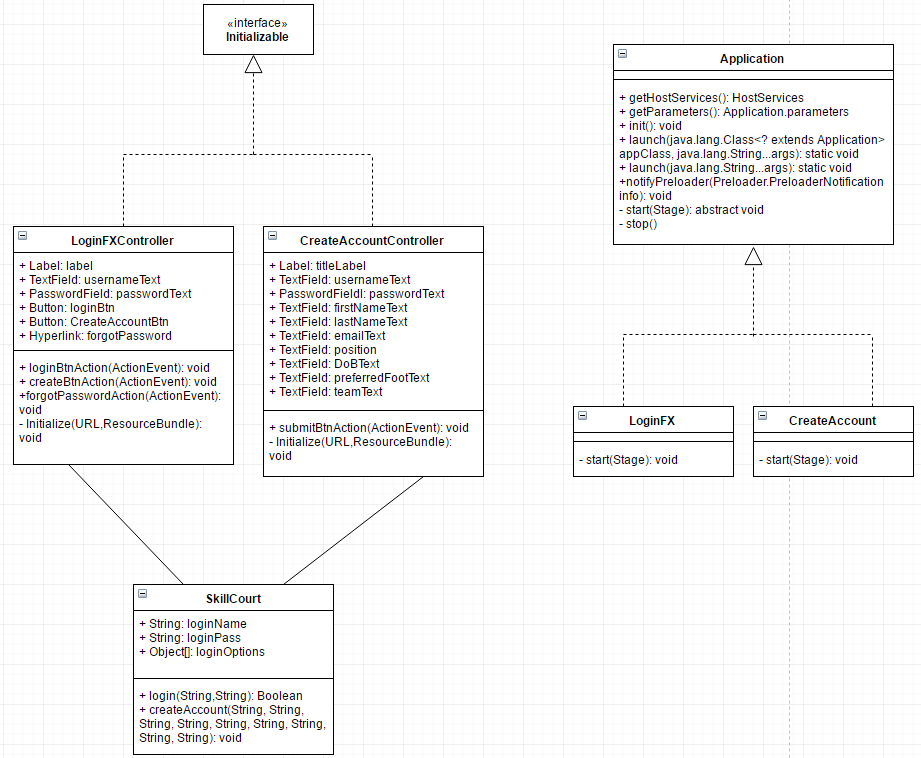


Figure - Database

### 

### 

Story734_Class_Diagram.png

Figure - Send and Receive data

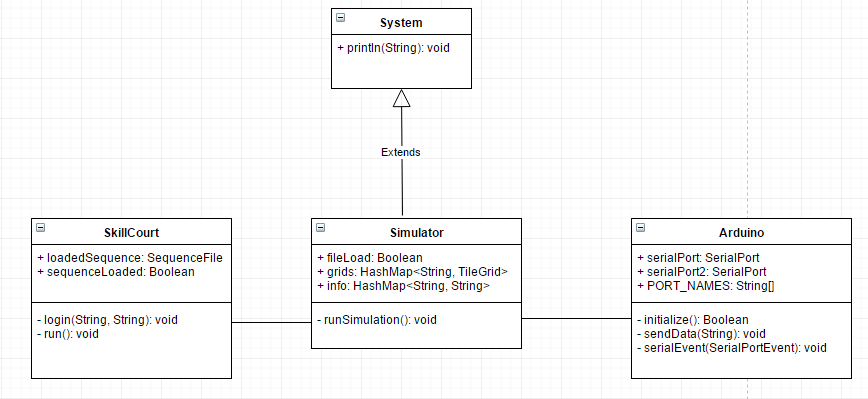


Figure - Communication with multiple pads

Sprint 3 class Diagram.png

Figure - Start Menu

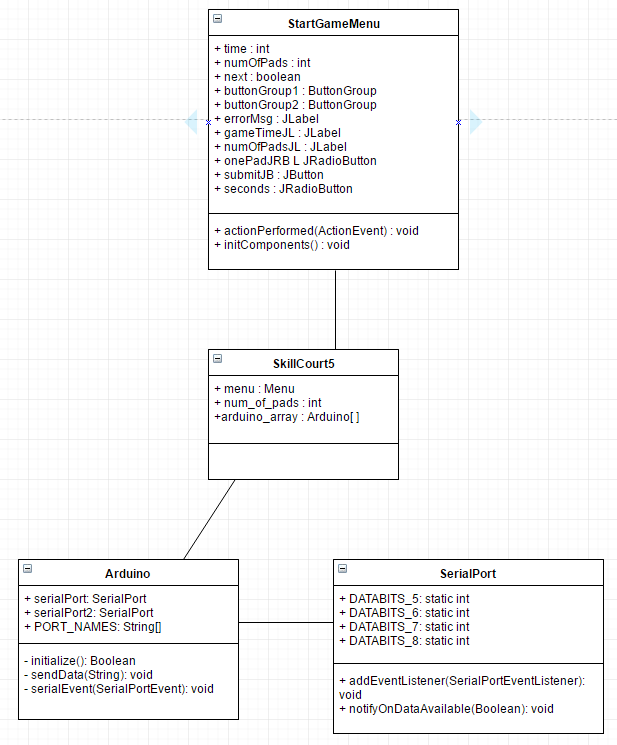


Figure - Send Random Sequence to Pads.

Story740_ClassDiagram.png

Figure - Change Pad Color

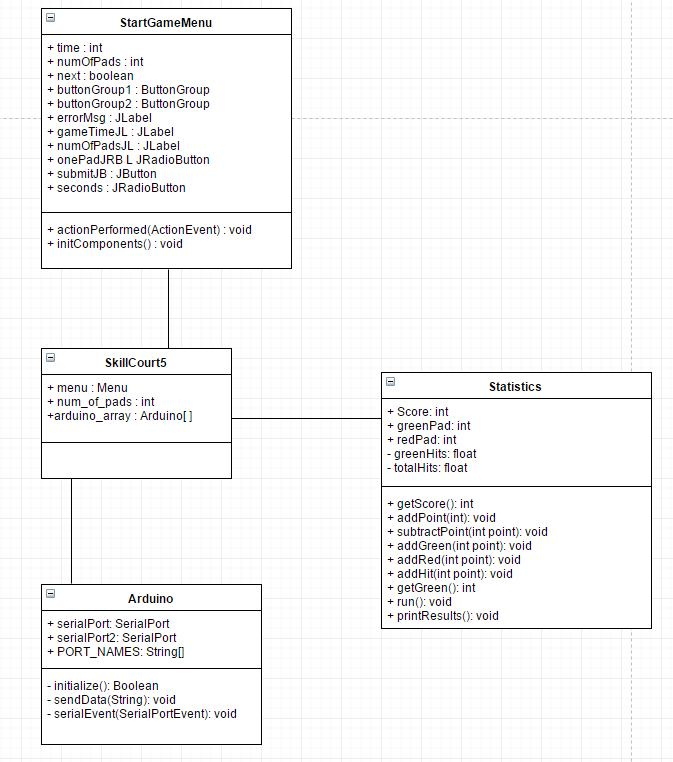


Figure - Instant Feedback

**Dynamic UML Diagrams**

Login_Usecase_Diagram.png

Login_Sequence_Diagram.png

CreateAccount_Usecase_Diagram.png

CreateAccount_Sequence_Diagram.png

ForgotPassword_Usecase_Diagram.png

ForgotPassword_Sequence_Diagram.png

Send_data_Usecase_Diagram.png

Send_Data_Sequence_Diagram.png

Receive_Data_Usecase_Diagram.png

Receive_Data_Sequence_Diagram.png

Setup Menu Use case diagram.png

Setup Menu Sequence Diagram.png

Session timer use case Diagram.png

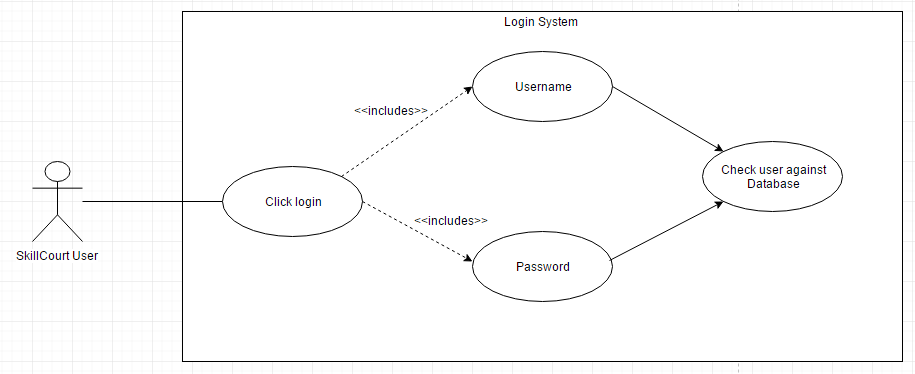
Timer Sequence Diagram.png

FS_Listen_UseCase_Diagram.png

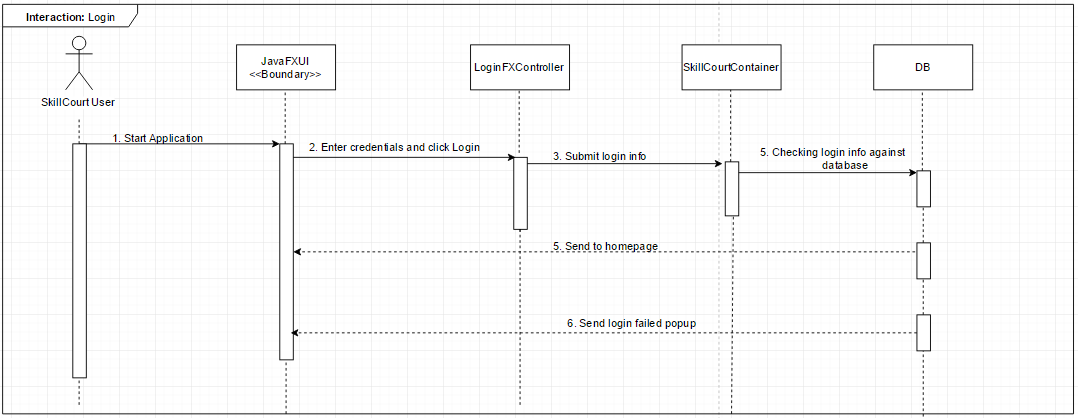
FS_Receive_Sequence_Diagram.png

Receive_GreenRed_UseCase.png

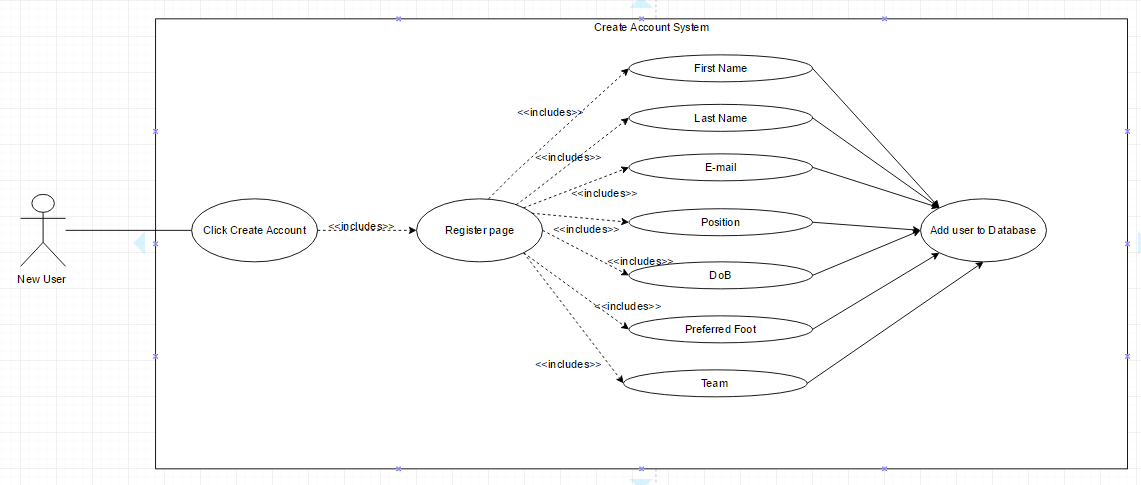
SerialPort_Receive_Sequence_Diagram.png

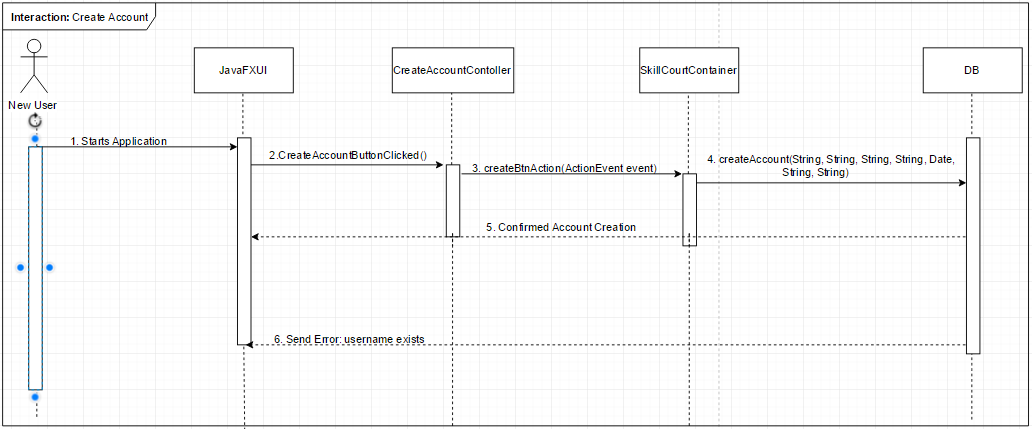


## 

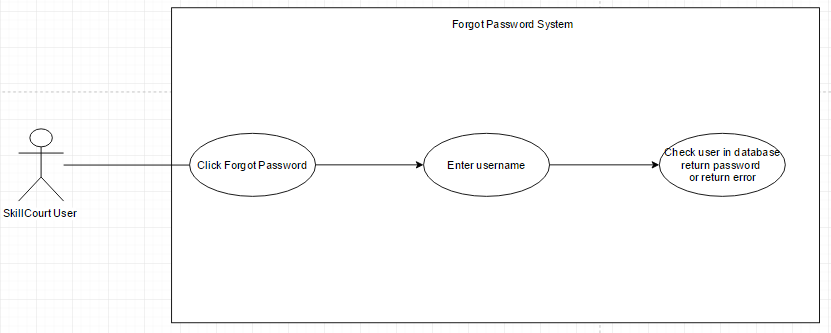


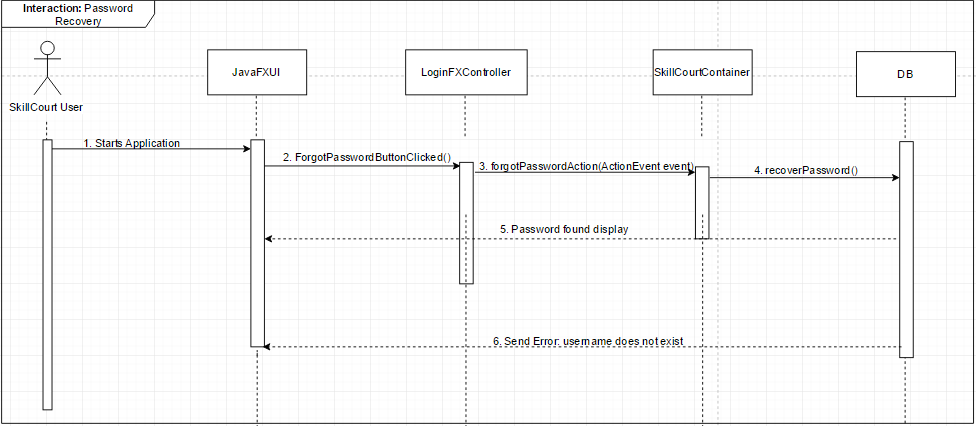
## 

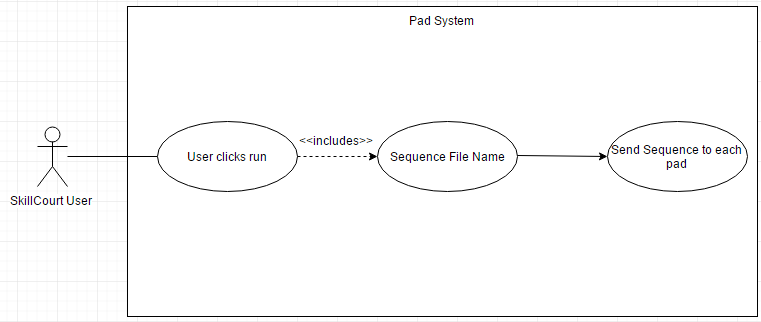


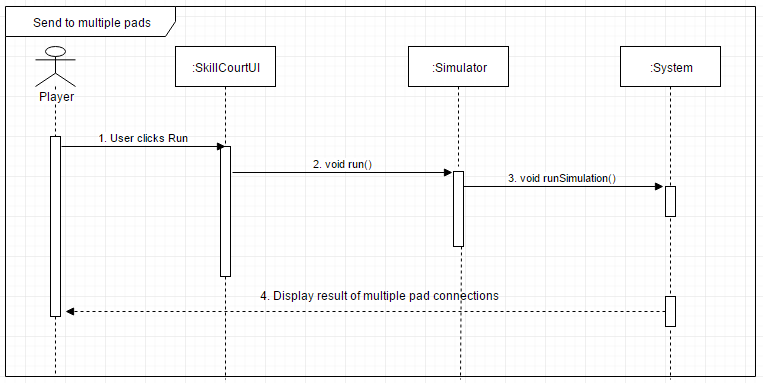


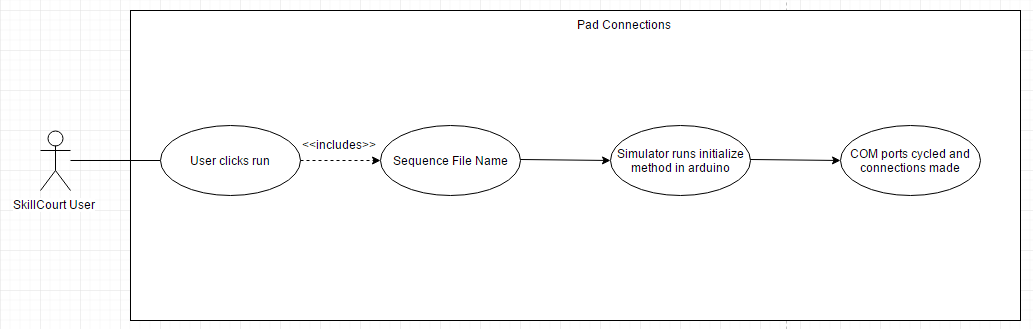
## 

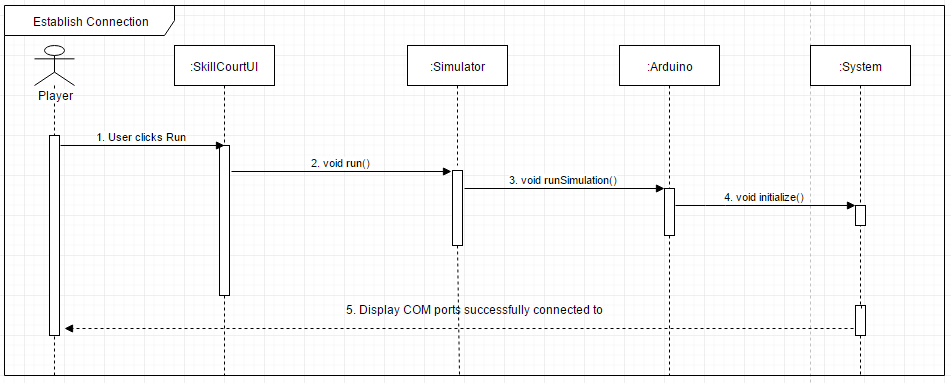


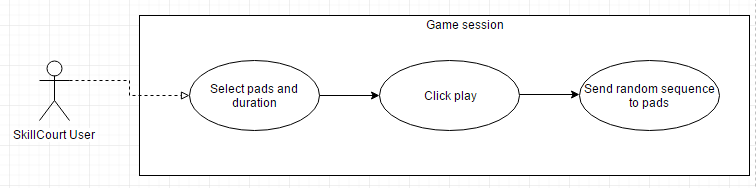


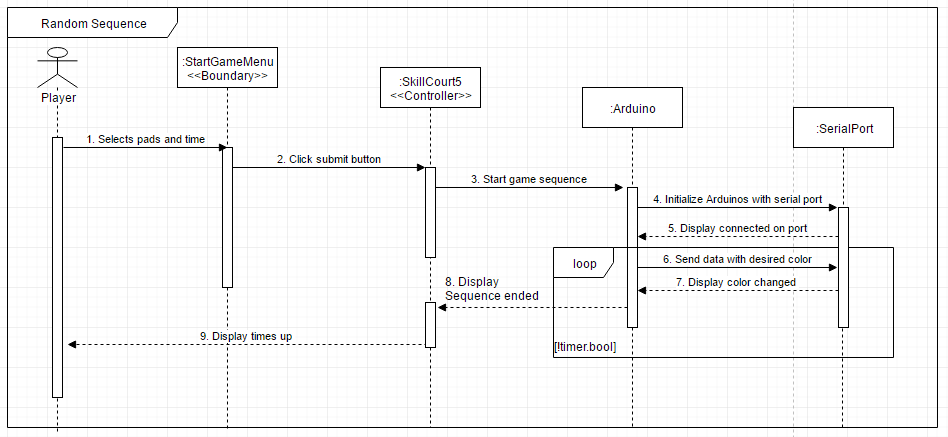


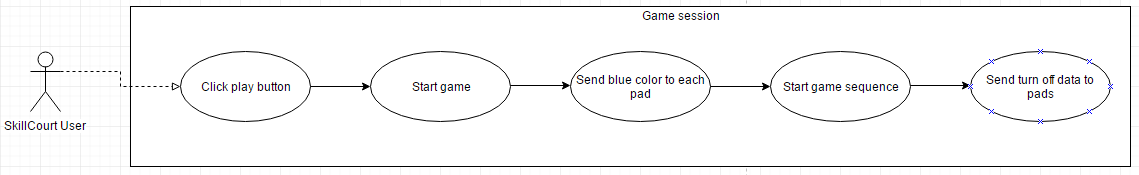


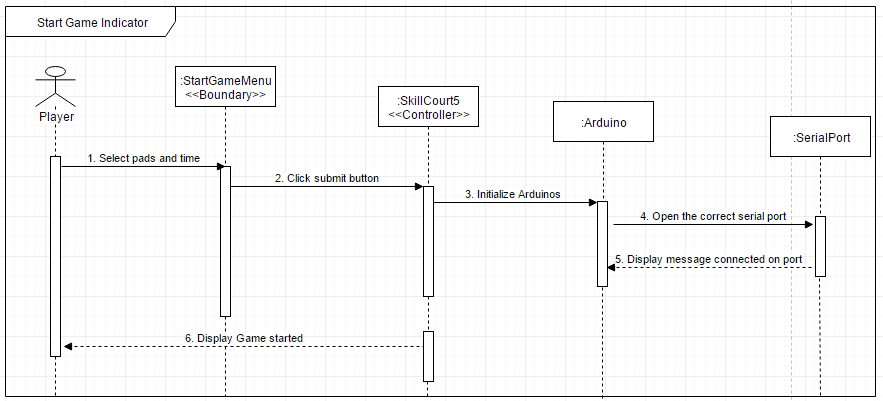


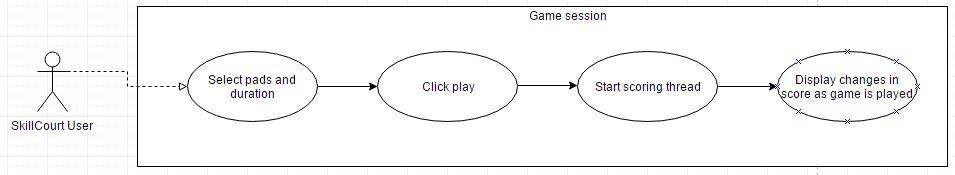


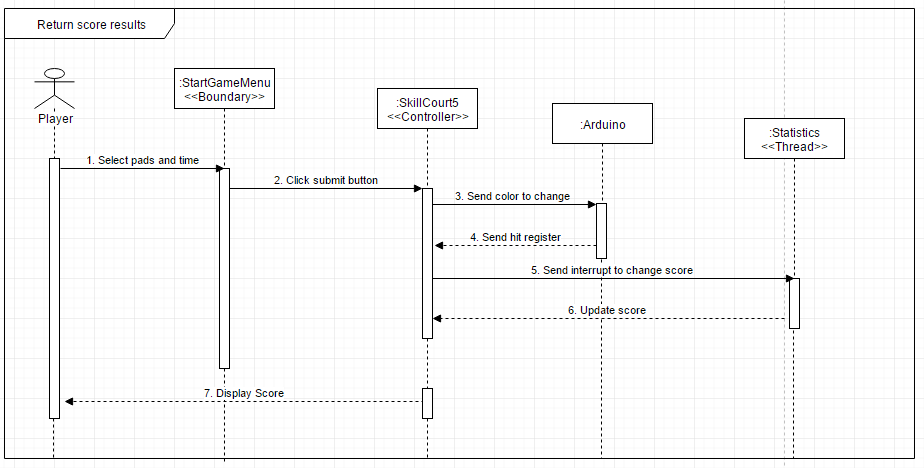












## 

## 

## Appendix B - User Interface Design



Figure - Login & Create Account screen

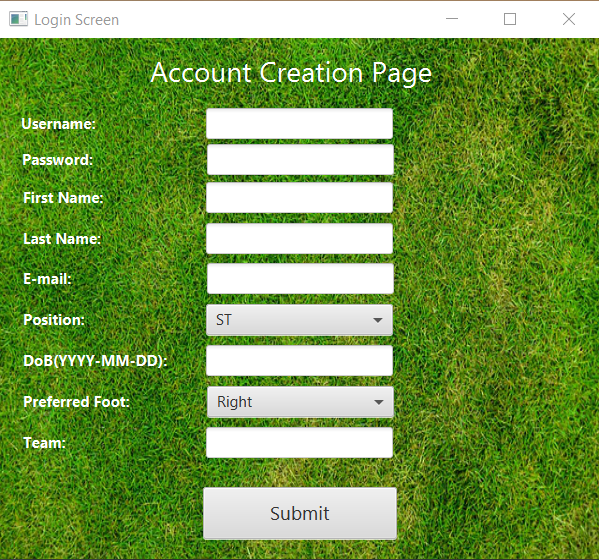


Figure - Create Account page

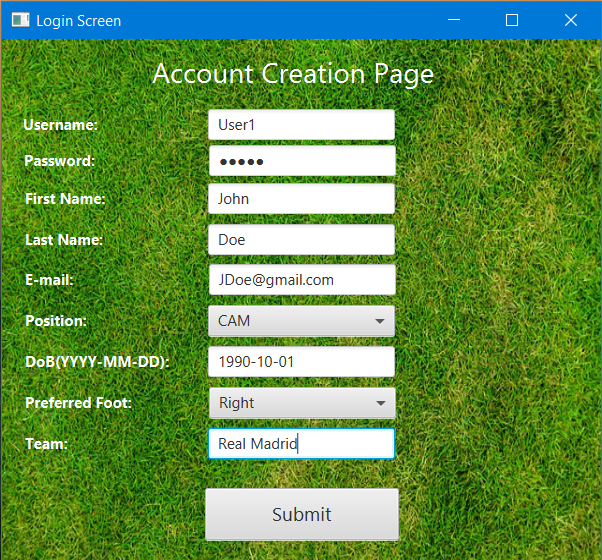


Figure - Sample filled in Create Account page

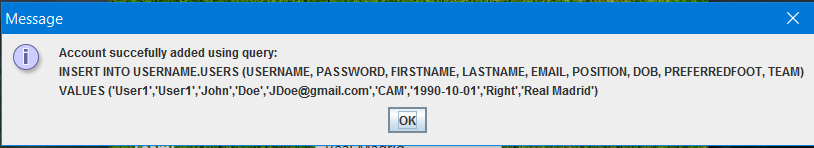


Figure - Successful account creation

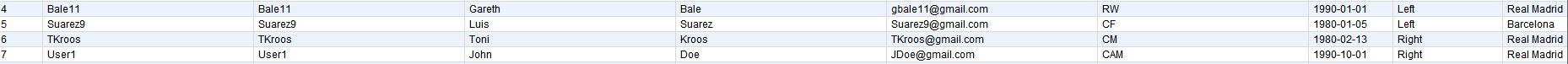


Figure - Sample stored information in database

## 



Figure B6 - Sample Username and PW input

## 

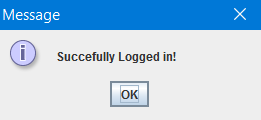


Figure - Successful login into the system

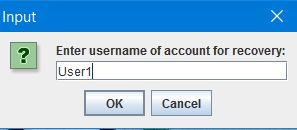


Figure - Password Recovery selected

## 

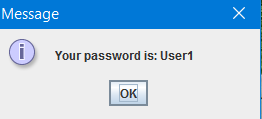


Figure - Message displayed when credentials match

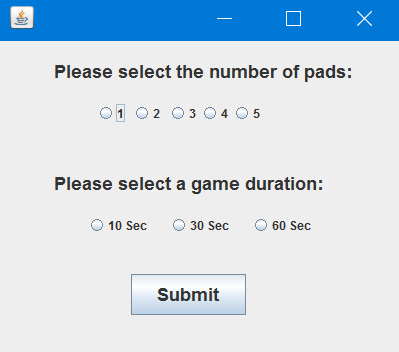


Figure - Start Menu

## 

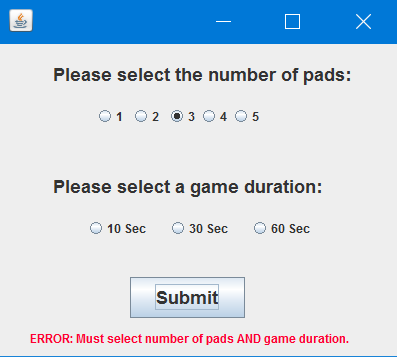


Figure - Error msg when pad number selected but not time.

## 

## 

## 

## 

## 

Figure - Error msg when time selected but not pad number.



Figure - Scoreboard

## 

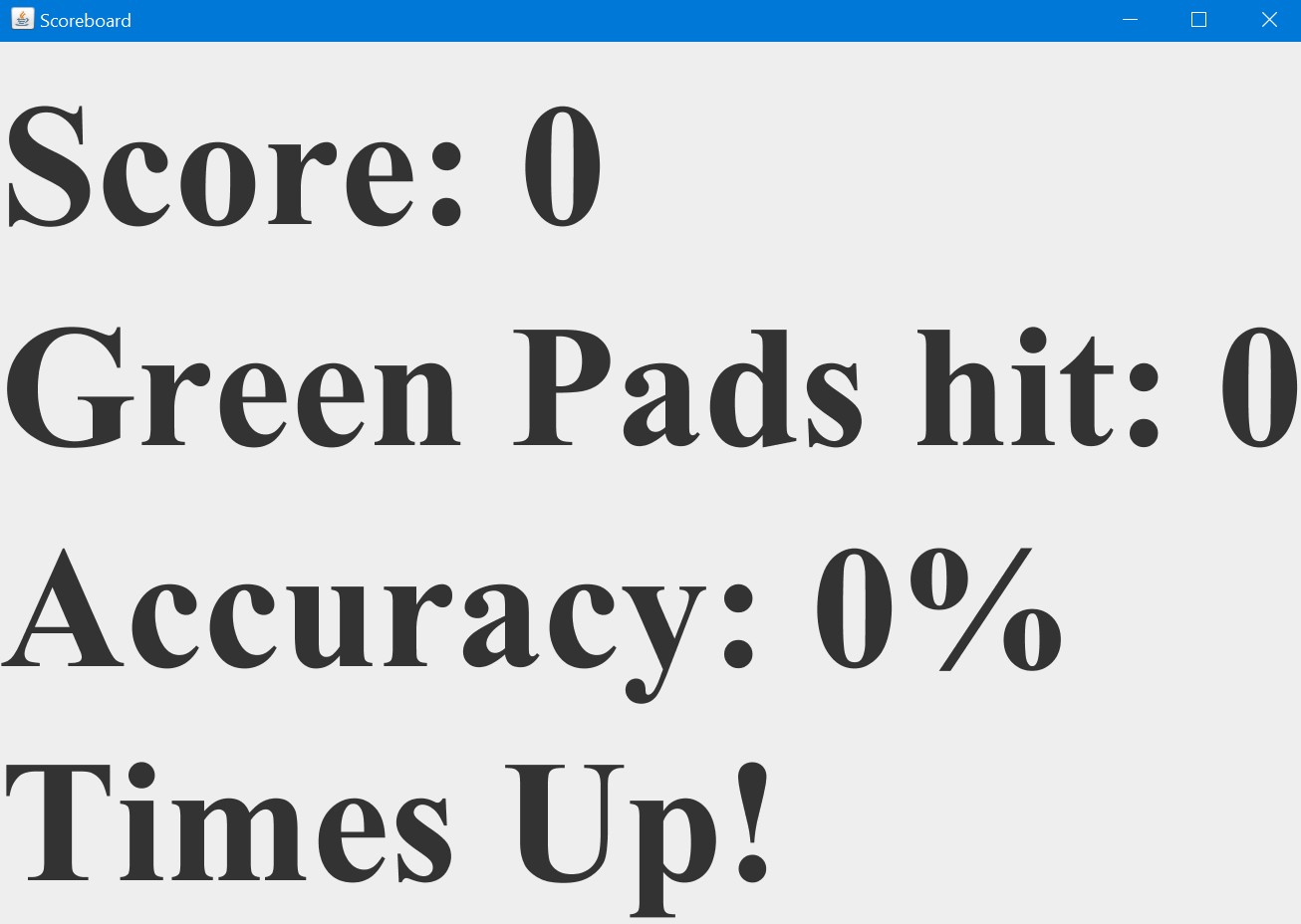


Figure - Scoreboard when time has expired

## 

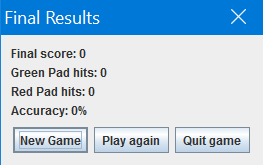


Figure - Final Results Menu

## 

## 

## Appendix C - Sprint Review Reports

**Sprint 1:**

Attendees:Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 9:15PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 718 - Create GUI for the PC version
* User Story 719 - Database management

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* Initially there were only two user stories and both were completed on time.

**Sprint 2:**

Attendees:Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 7:15pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 724 - Sending data to multiple pads
* User Story 734 - Receive data from pad

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* Two user stories were worked on and completed for this sprint.
* Porting the rest of the processing code is taking a lot longer than we initially projected, it is going to take a few sprints to complete this task as we need work from previous stories to be completed in order to progress through the rest of the processing code and port it over.

**Sprint 3:**

Attendees:Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 7:15pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 727 - Add a game setup menu and session timer
* User Story 739 - Send discrete sequences to each pad

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* Two user stories were worked on and completed for this sprint.

**Sprint 4:**

Attendees:Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 6:00pm

End time: 9:30pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 740 - Receive data from all pads and display it.
* User Story 719 - Provide instant statistical feedback.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* Two user stories were worked on and completed for this sprint.

**Sprint 5:**

Attendees:Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 9:30pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story 770 – Display a countdown timer.
* User Story 735 – Create a Results screen.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* Two user stories were worked on and completed for this sprint.

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

Attendees: Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 9:15

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Our velocity was slower than expected
* Did we do a good job estimating the points (time required) for each user story?
  + Our estimation as a bit high which resulted in a slower velocity
* Did each team member work as scheduled?
  + Each member followed the schedule but was slower than expected.

What went right?

* The final design and functionality of the login/create account page and database were a success.

How to address the issues in the next sprint?

* How to improve the process?
  + Better sense of velocity.
* How to improve the product?
  + Continue to work as has been done and continue to follow the plan set out.

**Sprint 2 Retrospective**

Attendees: Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 7:30pm

What went wrong?

* Did we do a good job estimating our team's velocity?
  + The focus so far have been one user story between the two of us, and that has led to good results of completion of the stories on time including documentation.
* Did we do a good job estimating the points (time required) for each user story?
  + We found the two user stories to each take about the entire sprint’s duration to complete, which is why they were both given 8 points each.
* Did each team member work as scheduled?
  + Yes, another great week of work for each of us.

What went right?

* We were successfully able to connect to multiple Arduinos using Java and are now able to receive data from the Arduino.

How to address the issues in the next sprint?

* How to improve the process?
  + We need the engineering room made available to have better access to the pads.
* How to improve the product?
  + Have each Arduino communicate on its own with Java.

**Sprint 3 Retrospective**

Attendees: Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 7:00pm

End time: 7:30pm

What went wrong?

* Did we do a good job estimating our team's velocity?
  + The focus so far have been one user story between the two of us, and that has led to good results of completion of the stories on time including documentation.
* Did we do a good job estimating the points (time required) for each user story?
  + We found the two user stories to each take about the entire sprint’s duration to complete, which is why they were both given 8 points each.
* Did each team member work as scheduled?
  + Yes, another great week of work for each of us.

What went right?

* We were successfully able to send discrete sequence to each Arduino and implement a timer and set-up menu.

How to address the issues in the next sprint?

* How to improve the process?
  + We need access to the force sensors in order to begin the next phase of development.
* How to improve the product?
  + The system needs to record data received from each force sensor.

**Sprint 4 Retrospective**

Attendees: Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 6:00pm

End time: 9:30pm

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes, everything was completed on time.
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, we figured the work required would be out double from the previous sprint which turned out to be the case.
* Did each team member work as scheduled?
  + Yes, another great week of work for each of us.

What went right?

* We were able to receive data from all the pads and have a working instant feedback feature which was the main feature the PO wanted and he was very pleased with the outcome.

How to address the issues in the next sprint?

* How to improve the process?
  + We have more meetings planned with the PO and the engineering teams.
* How to improve the product?
  + Now that the functionality is working, we will focus on improving the visuals of the menus created as well as add more options to the menus.

**Sprint 5 Retrospective**

Attendees: Sean Borland, Gajen Gunasegaram, Gudmundur Orn Traustason

Start time: 6:00pm

End time: 9:30pm

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Yes, everything was completed on time.
* Did we do a good job estimating the points (time required) for each user story?
  + Yes, we figured the work required would be out double from the previous sprint which turned out to be the case.
* Did each team member work as scheduled?
  + Yes, another great week of work for each of us.

What went right?

* We were able to display a detailed final results on a final results screen with capabilities for play again, new game, upload score, and view previous score.

How to address the issues in the next sprint?

* How to improve the process?
  + We have more meetings planned with the PO and the engineering teams.
* How to improve the product?
  + Now that the functionality is working, we will focus on improving the visuals of the menus created as well as add more options to the menus.

**References**

<https://www.arduino.cc/>