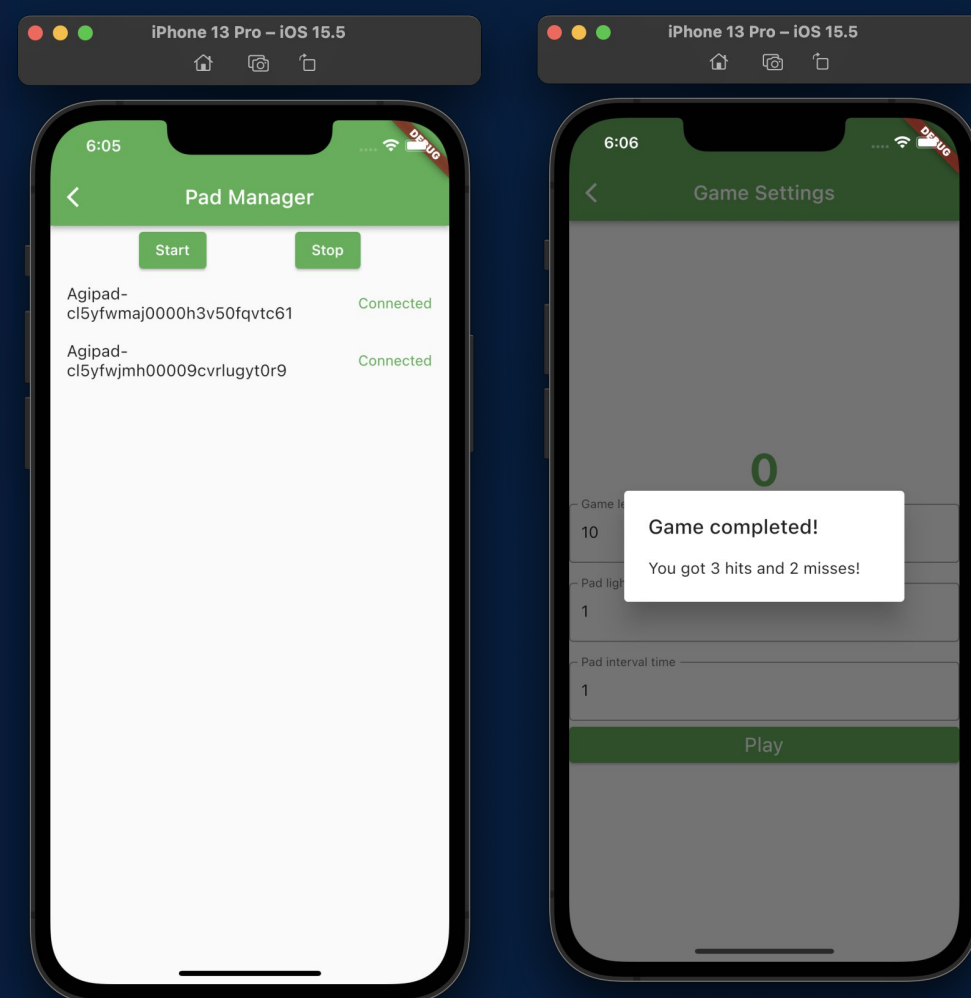


## SkillCourt

### PROBLEM

- Skillcourt allows soccer players to train and measure practical skills often developed in real matches in a “game” using LED pads. This removes the tedium of needing other players to train and allows one to grow their skills on their own.

### SCREENSHOTS



### SYSTEM DESIGN

The Skillcourt application uses network service discovery to discover services registered on the local network. The pads register themselves on the LAN and the app connects to them. Once connected, the user can configure three settings and start the random pad game which displays game stats once finished. These statistics are stored in-memory and thus cannot be accessed after the app closes.

**Students:** Yasaar Kadery

**Mentor:** Joseph Quinn, Mentor; Gudmundur Oxndal, Product Owner

**Instructor/Faculty:** Dr. Masoud Sadjadi, Florida International University

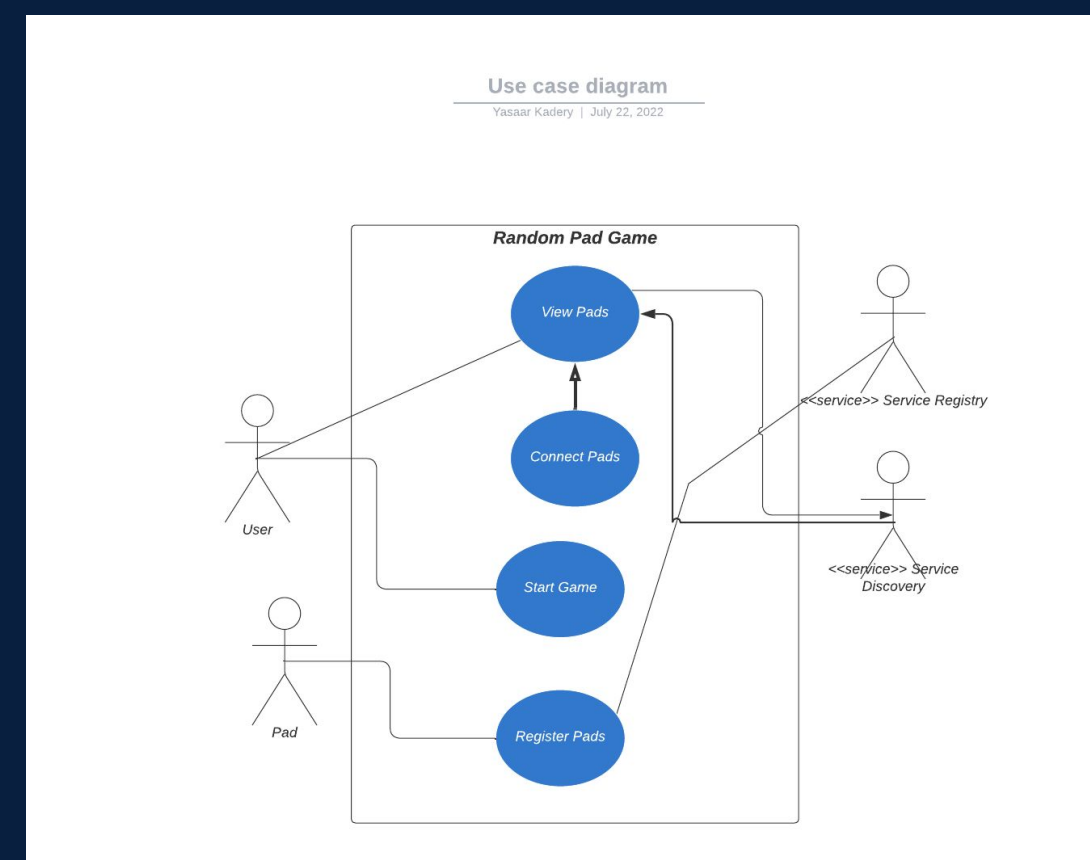
### CURRENT SYSTEM

The current Skillcourt app supports the ability to connect to multiple “pads” simultaneously. The goal for this semester was to develop a game that randomly lights up pads for a user configurable number of seconds. We also had a few goals that we could not complete:

- Create GraphQL mutations for the backend
- Implement user sign up and login functionality
- Implement persistent storage of game statistics for users

### OBJECT DESIGN

Figure 1: Use Case Diagram for SkillCourt Pad Pugin



### IMPLEMENTATION



### REQUIREMENTS

In order to work on this project, the proper environment is needed. The following is necessary:

- Android Studio (for windows) or XCode (for MacOS)
- git
- git-flow-avh or another git-flow plugin for Git.
- A proper emulator for Android or iOS
- Flutter SDK along with Dart
- Java JRE version 1.8 or greater
- Python 3 (for the pad emulator)

### SUMMARY

My main contribution to the project was developing the algorithm for the randomized pad game and the UI for the page. Last semester’s students had already implemented the ‘threading’ to deal with connecting to the pads using Flutter isolates. For this semester, I had to make some modifications to the current logic to support multiple pads being connected simultaneously. There were some changes to the naming convention of the pad service on the local network which made the application unable to discover these services as they were looking for the wrong name. The algorithm was fairly simple. Once connected, the pads were added to a Map data structure with their respective hostnames and socket information allowing us to send and receive information with each pad. Using the flutter Random package it was simple enough to create a pseudo-random sequence of pads to light up.

### ACKNOWLEDGEMENT

Special thanks to Gudmundur Oxndal and Joseph Quinn for the mentorship they provided throughout the semester.