For my senior project, I will be creating a 2-player implementation of the board game Kismet, playable against an intelligent computer agent. Kismet is a board game similar to Yahtzee and is most notable for having colored die faces. This introduces a variation to the scoring categories by simulating poker hands. There are many research papers and optimal strategy implementations in existence for solitaire Yahtzee, but few to none for 2-player Kismet. The main focus of this senior project will be to develop a flushed-out GUI that allows the user to smoothly play an implementation of the game. Creating components of Kismet place emphasis on good object-oriented design, an attractive, user-friendly interface, and robust mobile development. Developing an intelligent computer agent will require training a neural network with data gathered from running the optimal, heuristic-based strategy. One of the unique aspects about Kismet (and Yahtzee) is that the optimal strategy is complex enough to be unrealistically calculable for a human but simple enough for a machine to compute (somewhat) inexpensively. Computer intelligence and mobile development are two areas I have little experience in while past internships and projects have introduced me to frontend development and object-oriented design. This will provide a good opportunity learn new skills by exercising existing ones. Additionally, Professor Glenn is an expert in the field of computer intelligence and stochastic games: his expertise will certainly be helpful over the course of the project. Outlined below are the planned deliverables.
Starting Steps

The initial steps will consist of getting a rudimentary text-based form of solitaire Kismet working. This involves running a simple Python script, where at the beginning of every turn, the current score of the 13 categories is output and 5 dice are rolled. The player can choose to reroll certain dice by inputting the respective dice numbers or to simply select a category to score the current dice in. Once a category is selected or dice have been rerolled up to 2 extra times, the game moves to the next turn and this process repeats. At the end of 13 turns, the program will compute the players total score. This text-based implementation will be a good start to fleshing out the matching of dice rolls to respective categories, rerolling certain dice, keeping track of additional scoring bonuses (rolling a “Kismet”: 5 of a kind), and disallowing the player from making illegal moves, such as scoring into the wrong category, scoring into a category that has already been scored into, or rerolling an incorrect number of times.

Mobile App

The next step will be creating a playable mobile application for Kismet. The framework utilized will be React Native, which synthesizes JavaScript and Facebook’s React to employ a cross-platform experience. The main focuses of this iteration will be to develop the ability to play against a second player (which can later be selected as either human or CPU) and to design certain GUI components. Effectively, the main GUI can be divided into two sections: the current score table and the dice. A player should have the ability to select one of the 13 table cells to either score into that category (assuming a legal dice roll) or to view the rules for scoring into that category. Additionally, after an initial roll, the user should be able to select certain die to reroll again should they choose. Creating dice animations for rolling would be an interesting exploration into the React Native framework and may require actual Android/iOS embedding in
order to truly function cross platform. Lastly, certain aspects such as a global leaderboard, personal score tracker, game tutorial, and hint giver will be developed should time permit.

**Intelligent Computer Agent**

In terms of solitaire Yahtzee, both an optimal solution to maximize the expected score and to beat a certain score have already been solved. Jacob Pawlewicz delves into 2-player optimal strategies. His findings will serve as a guiding to create the skeleton behind the intelligent computer agent. In the solitaire version, one is attempting to maximize their average score while with another player, one simply needs to beat the score of the opponent. In his paper, Pawlewicz argues a near-optimal strategy. Building this near-optimal strategy will require first coding an implementation of the solitaire optimal strategy. This can then be fine tuned from achieving maximum expected value to beating a score (one’s opponent). This can then be used to gather “training” data, which can be fed into a neural network in order to ultimately create the heuristic-based near-optimal player Pawlewicz describes. The computer agent will likely be developed in Java, which should conceive considerably shorter runtimes than a Python alternative.

**Final Product**

In summary, the final product should be a playable, cross platform Kismet mobile game where the user can either play against a friend or against a non-optimal but highly challenging CPU. The main UI components will be a score chart with categories, tutorial/ruleset, and die roller. The computer agent will involve training a neural net with data from the brute force optimal strategy.