# Unit Testing Procedures

**Product name:**

**Team name:** P A W N

**Date:** 8th December 2017

## 

Connor and Erik:

Image Recognition (PC):

* Image accuracy equivalence classes**:**
  + Included in the gitHub is some sample data
  + Run the following command: DataLabeling.py [ImageName.jpg] [Orientation]
    - The orientation should be “Top”, “Bottom”, “Left”, or “Right” which depends on the a1 to h1 row location in the picture
  + This should start running and take about one to two minutes to complete
  + On completion an output of the accuracy should be printed, along with the highest probability piece
    - Accuracy should then be compared to the actual board
    - A successful test contains accuracy of pieces higher than 75%
* Random image equivalence classes:
  + Run the following command: DataLabeling.py [RandomImage.jpg] [Random]
    - The orientation should be “Top”, “Bottom”, “Left”, or “Right” which can be random for this test
  + This should start running and take about one to two minutes to complete
  + On completion an output of the accuracy should be printed, along with the highest probability piece
    - This should output random pieces, but more importantly not crash

Jacob and David:

App-Side:

* Expected input for strings: 64 substrings of size 2, delimited by spaces
  + If input correctly: board loaded with specified positions
* String parsing equivalence classes (passed into board engine):
  + Input: String size too short (not enough substrings):
    - Output: App breaks
  + Input: String size too long (too many substrings):
    - Output: first 64 legal piece positions are put in the board as expected
  + Input: First letter of a two character substring for position is not “b” or “w”:
    - Output: App breaks (Illegal Argument Exception)
  + Input: Second letter of two character substring for position is not “r”, “b”, “n”, “k”, q”, “p”:
    - Output: App breaks (Illegal Argument Exception)
  + Input: Substrings for positions just one character:
    - Output: App breaks
  + Input: Substrings for positions more than two characters:
    - Output: Parses first two characters of each substring as expected, and ignores the remainder of the substring.
  + Input: Multiple spaces in unexpected places
    - App breaks
* Expected input for board state: Reachable board state from game of chess (not always the case from output of trained model)
  + If input correctly: game is playable where left off
* Board state equivalence classes (after string parsed):
  + Input: No Kings
    - Output: Unpredictable Behavior
  + Input: No pieces on either side
    - Output: Unplayable game
  + Input: No white pieces:
    - Unplayable game, user cannot move
  + Input: No black pieces:
    - Playable game; computer moves nonexistent pieces
  + Input: Only pawns:
    - The silliest game of chess you’ve played in your life
  + Input: Multiple kings:
    - Only one king movable
  + Input: Only Kings:
    - Game is called a draw
* Pressing application buttons
  + Play chess
    - Without entering board text, should start as starting board
  + About
    - Talks about most current version of this chess application from MIT
  + Take photo
    - Capture image
      * Go to image gallery on phone and make sure image appears
  + Load board