\*Just a first attempt, looking for feedback before finishing or adjusting this plan/format. Needs to be updated based on scope, finalized requirements, and use cases.

Baker’s Dozen Solitaire Card Game Testing Plan – SWE 3313 Term Project

This test plan covers the different aspects of a game of baker’s dozen solitaire, how those software pieces interact, and how each must be tested individually and as a whole to meet the functional requirements of our client Kennesaw State University via Dr. Chao Mei.

Test Policy……………………………………………………………………………………… 1

Test Strategy and Project Test Plan………………………………………….............................. 2

Components Tests Plan……………………………………………………………………………

Integration Test Plan…….………………………………………………………………………

System Test Plan…………………………………………………………………………………

Test Policy

# The overall test policy for the project should focus on the design requirements laid out in the SWE\_3313\_Term\_Project\_Apring2020 document along with additional requirements laid out by the development team. These are as follows

# \*Subject to change as Scope document is finalized

# Backend

# A standard deck of 52 cards (no Jokers)

# Randomized into 13 piles (stacks)

# Four piles must have kings as the bottom item in the stack

# Four foundations that each accept cards of one suit and in order from lowest to highest

# Ability to reset game to new random start state

# Empty tableaus and foundations, then refill randomly

# Only the top card of each tableau can be moved

# Cards can only be placed on cards that are one value higher

# Win condition of all foundations filled and the tableaus being empty

# Timer that resets on game start/reset

# Move counter that resets on game start/reset

# Frontend

# Website for users to access

# GUI with graphics for the cards, tableaus, foundations, reset button

# Dynamic timer graphic

# Dynamic move counter graphic

# Each of these features will be tested for functionality to the development teams, client’s, and end user’s desires and requirements. Test results that do not meet requirements will be fixed by the development team and results that call in to question the aspects of requirements will be relayed to the client for review and any requirement changes will be passed back to the development team.

# Test Strategy and Project Test Plan

# 

# Testing will occur at each milestone. All features completed upon the reaching of the milestone will be tested to show they are functional and designed with both the client’s intent and efficiently in mind. Tests scripts will identify and indicate fixes for any defects and be able to retest components or systems post fixes.

# During final system testing the product will also be tested for user acceptance.

# …(incomplete)

Components Test Plan

The Deck of cards

* + - Test should increment through the data structure the cards are stored in and display the cards value, suit, and color to confirm the existence of all 52 unique cards
      * The four kings can be removed and checked in a separate data structure making a check of 4 cards and a check of 48

Randomizing the Deck

* + - The cards data structure should be randomized then incremented through and display the cards value, suit, and color to confirm all cards are present and their order. Then the data structure should be randomized and analyzed again. Repeat this process a number of times and record results
      * The four kings if separate should be tested in the same way

Putting the Deck into the Four Tableaus

* + - Test the ability to empty tableaus
      * This test script should empty the tableau data structures and increment through each of them to confirm they are empty
    - Test the ability to randomly insert the four kings into the 13 tableau data structures.
      * Increment through each of the tableaus to confirm the placement of the kings, then reset and repeat until satisfied each of the kings is being randomly and uniquely placed into one of the 13 tableaus
    - Test the ability to randomly insert the remaining cards into the tableaus
      * This should be done after testing the four kings and each instance of this test should start from an end of case of the king test. Confirm the cards are being put into the tableaus and accounts for the already placed kings. The test should increment through all 13 tableaus showing that each has four cards and four have kings as their lowest item

Card Movement

* + - * + Cards should be put into the tableau in a not random order to test movement interaction

cards can be placed on a card of any suit that is one value higher

nothing can be placed into an empty tableau

* + - * + Scripts must be written that call individual tableaus and attempt to move card objects to other tableaus
        + The movement of cards to the foundations must also be tested

Only one suit per foundation, stacks from lowest to highest card value, with ace acting as lowest.

* + - * + If a user-friendly interface has been built client/user testing can begin to be done here. This could help cover any input conditions that test scripts fail to.