Final Project: Spades

# Introduction

Project: Implement a two-player version of Spades

Project members: Derek Luong and Jamie Bowers

Class: SWE 681- 001

Date: 2019-04-27

# Design/Architecture

# Installation Instructions

## External Dependencies

The following dependencies are required in order to install and play the spades game. This paper assumes the user knows how to install, configure, and run these dependencies.

* PostgreSQL 11
* Any Java IDE that supports Maven 3.5.2 (We used IntelliJ) with the 10.0.2 JDK
* JDK 10.0.2
* Java 1.8
* Two Separate Internet Browsers (We tested against Firefox and Google Chrome) or two separate computers.

## Initial Setup

In order to run the Spades game, the user must set up the initial database. This involves creating the necessary tables for both Spring Security as well as tables used to run and store information about Spades games.

Inside the Java project there is a folder called “sql\_scripts” and a file called “create\_database.sql”. To create the necessary tables, run the file “create\_database.sql” inside the Postgres’s pgAdmin tool using the Query tool.

# Operating Instructions

The first step in order to play this game requires that two user accounts are created.

# Game Rules

## Objective

Be the first player to reach 500 points. If both players reach 500 points in the same round the player with the higher number of points between the two wins.

## Setup

Each “Round” a 52 Card Deck is shuffled. Two players are each dealt thirteen cards from the deck randomly. This leaves 26 cards that are remaining in the deck, which will not be used in the current round. (13 cards per player \* 2 players + 26 cards left over = 52 Total Cards in Deck)

## Card Rank

Card Values are Ace (A), King (K), Queen (Q), Jack (J), 10, 9, 8, 7, 6, 5, 4, 3, and 2. Suits are Spades (S), Hearts (H), Clubs (C), and Diamonds (D). Spades beats every other suited card. E.g. 2S (2 of Spades) beats AH (Ace of Hearts). In the case of same card suits, the higher valued card will win, so 9H (9 of Hearts) beats 7H (7 of Hearts). If a Spades suit is not played but the played cards have different suits, then the leading player (the player that went first in the current trick) will win regardless of card value.

## Bidding

Each player will decide how many “Tricks” they think they can win in a round. They can also choose “nil” if they want to try to lose every trick that hand to try to get 100 points for that round (See Scoring) A “Trick” is simply a winning hand. Bidding occurs each round.

## Scoring

Each winning trick is worth 10 times the bid amount and every trick that is won that goes over the bid amount is referred to as a “Sandbag” or “bag” and is worth 1 point. **Example:** Player 1 bids 6 tricks and wins 8. They are awarded 62 points, 60 for their bid and 2 for their bag.

If a player fails to meet the number of tricks they bid the player is deducted the entire bid times 10 points for each bid. **Example:** Player 2 bids 8 tricks but only gets 6 tricks in the round, they are deducted 80 points.

Sandbags are kept track of for each player and once a player hits 10 bags or more they are deducted 100 points, and their bags are subtracted by 10.

A player can bid 0 (known as a NIL bid) if they believe they can lose every trick in order to get 100 points for the round. If they fail to do this and get even 1 trick they are deducted 100 points.

## Gameplay

Player 1 starts the game and plays a card, player 2 then follows. Spades are not allowed to be played unless they have already been “broken”. In order to “Break Spades” a player may play a spade if they do not have any cards of the suit that the opening player plays, or if the opening player only has cards of the Spades suit remaining. Once Spades is broken it can be opened with in any following round.

Example: Player 1 has 5S, 4C, 2D and 3D, player 2 has QH, 10D, 3S, and 4S. Player 1 plays 2D, player 2 must play a Diamond, in the next hand Player 1 Plays 3D because Player 2 no longer has any Diamonds they can play their 3 of Spades to win the trick, Spades are now broken and Player 1 can play their 5S in the next hand.

The winner of the trick starts the next hand. Each hand is played until the players are out of their initial 13 cards. The round is then scored and a new round is started. This process goes on until one of the players wins the game.

# Why This Application is Secure

## Confidentiality

**Requirement:** “You must create an audit trail of every game move, and these game moves must not be available to those not in the game until the game has completed. After the game is completed, the final audit record (including every move) must be publicly available to all authenticated users (but to not to unauthenticated ones). The win/loss/draw record of each user must be made available to authenticated users (but not to unauthenticated ones).”

We have created an audit trail of every game move using a database table called “moves” each time a player has played a card it records the card they played in the moves table, if a game crashes the round is reset and moves that we’re made before the completion of the round are deleted from that table, this is done because of the randomness of dealing a deck each round does not let us persist who had what cards if your game has crashed. The overall game moves are not shown until the game is over and the final audit report of who won that game, the rounds played during that game, the moves of that game, and points for the game. A win/loss statistics page is available when you are not playing a game, which shows wins and losses for every player and is only available to authorized players. We consider an authorized player any player that has an account and is logged in.

## Integrity

**Requirement: “**Only the current player can make a move in a given game, and it must be a legal move (syntactically and given the current situation). Make sure that the audit trail of game moves can’t be changed (except by legal moves during a game!), and that the win/loss/draw record of each user can only be changed by actually winning, losing, or drawing.”

In order to meet the requirement that only the current player can make a move in a given game we implement a check on whose turn it currently is and if the player whose turn it is makes a move it is allowed, if they player whose turn it is not tries to play a card the game does not allow that card to be played and the game continues to wait for the player whose turn it is. The same audit trail of moves we implemented for the “moves” table only records moves if the move was allowed as such it is impossible for players to change that table themselves, only the game records moves if the move was allowed. The “Win, lose, and draw” statistics is calculated each time a game is completed and is calculated from the games table in the database. Users can not make changes to these statistics themselves in any malicious way, if they want to change the statistics, they must complete playing a game.

## Availability

**Requirement:** “The player must not be able to make any game (including one he’s playing) pause forever. A timeout eventually means forfeit, and sending 1 byte should not cause a stall forever. A simple packet or command shouldn’t kill or stall a game. You can’t prevent everything, so in the case of DDoS attacks, try to relatively quickly recover once the attack ceases or is filtered out upstream. If a player gets logged out for some reason (e.g. wireless failure), they must be able to reconnect to the game and continue to play (as long as the timeout hasn’t expired). Your system must support multiple simultaneous games by different users (threads are permitted, but not necessary). It’s fine if a given user can only play one game at a time.”

We met the requirement to have a timeout in two ways. One we have a game time out in which after a user has created a game we set this timer to 5 minutes. We also have a hand time out that will forfeit the game for the user whose turn it is if they do not play a card in the 3 minutes we set for the hand timeout. For the second part of this requirement if a game for some reason crashes we allow the users to rejoin the game once they log back into their account. Our system also allows for multiple simultaneous games to be played however a player can only play one game at a time.

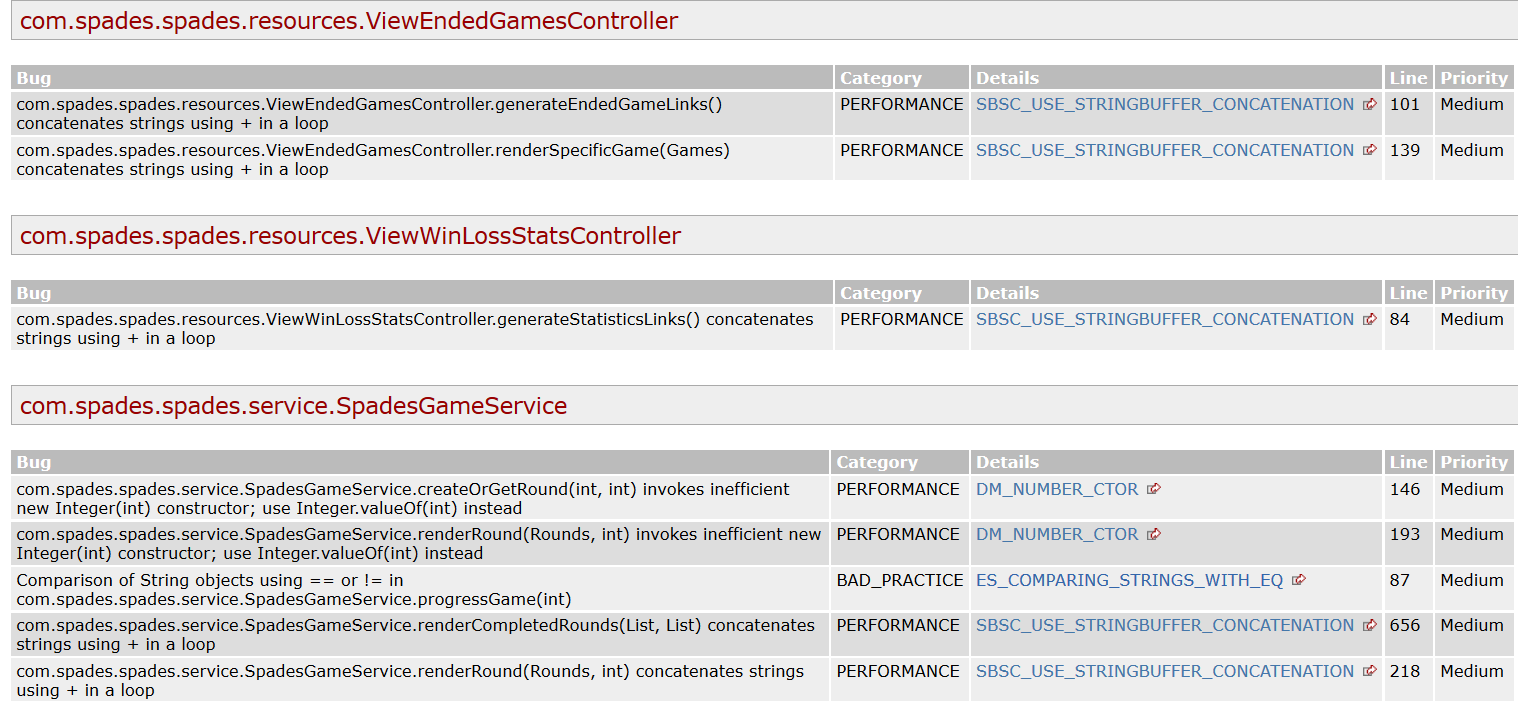
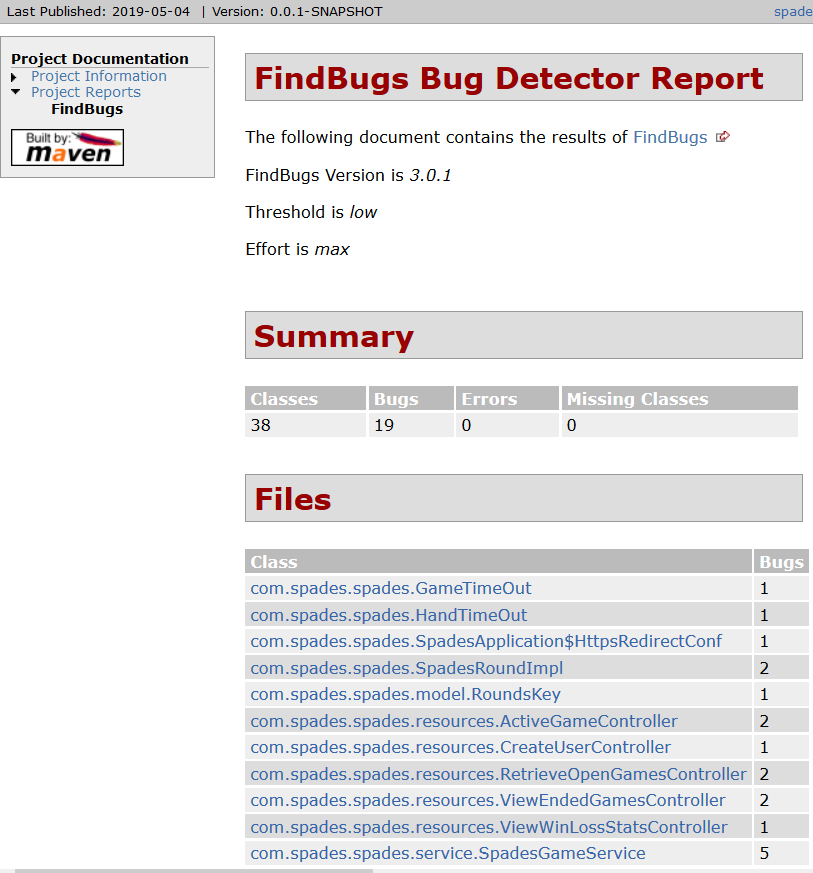
## Information protection

The entire application uses HTTPS to encrypt information that is exchanged. This reduces the risk that a third-party could retrieve sensitive data from web traffic that could be used to inflict damage.

Additionally, apart from the initial home page and the create user page, all other points in the application requires user authentication in order to access it. This is done by extending Spring Security’s default configuration and requiring that all requests to a certain path (containing the functionality of the application) requires user authentication, and will otherwise prompt the user with a login page.

## Using Analysis Tools

The static analysis tool FindBugs was used to scan for any potential issues. An initial use of the tool revealed 19 different issues. However, most of these initial issues were related to performance or bad practice, and didn’t reveal any severe problems. After using this tool, the issues were inspected and addressed. Below is the generated summary and a sample excerpt of the issues report.



After these initial issues were resolved, another use of the FindBugs report indicated the following: