SPIT

Design Inspection, Code Inspection and Unit Testing

Design Inspection Defect Log

Product	Front-End Design Inspection		
Date	2/3/17		
Author	Marty		
Inspectors	Byron, Nicky, Nick		
Defect #	Description	Severity	Correction Method
1	Poor design of Custom React Components for Front End made code cluttered and hard to understand.	3	Made less React Components that had more consolidated code in each.
2	The first iteration of the game board layout was very confusing and hard to understand.	2	We redesigned the layout of the board and asked people not involved in the project what they thought
3	The landing page was not aesthetically pleasing.	3	We took inspiration from a currently popular web pages to recreate the page.
4	The landing page did not have intuitive guide to play a match or look at the leaderboard.	1	We pivoted to a minimalistic approach by having all actions in the center of the page.

Product	Architecture Design Inspection		
Date	2/3/17		
Author	Byron		
Inspectors	Nicky, Nick, Austin, Marty		
Defect #	Description	Severity	Correction Method
1	Originally used strings to represent	1	Changed to numbers

	cards (ex: “2Spades”). This proved to be difficult when writing logic for moving cards.		(1-52 to represent cards, and 0 to represent an empty hand)

Product	Database Design Inspection		
Date	2/3/17		
Author	Nick		
Inspectors	Byron, Nicky, Austin, Marty		
Defect #	Description	Severity	Correction Method
1	Originally we created a separate database separate from the website with a web API that the site called. This caused the read/writes to the database to be noticeably slow on the client side.	3	We rebuilt the MongoDB interface so it connected directly to the website using mongoose.

Product	Backend Design Inspection		
Date	2/3/17		
Author	Nick		
Inspectors	Byron, Nicky, Austin, Marty		
Defect #	Description	Severity	Correction Method
1	Module-izing the card number for King and Queen resulted in 0 and 12, so it seemed like these cards were not adjacent and therefore could not be played on one another.	3	Add an edge case in our areCardsSequential method to account for this wraparound.
2	Originally we were going to send a password with every socket connection, but that is insecure.	2	We now authenticate using JWT when the socket connection is opened, making our communication secure.

Code Inspection Defect Log

Product	Backend Code Inspection		
Date	2/6/17		
Author	Nicky		
Inspectors	Marty, Byron, Nick		
Defect #	Description	Severity	Correction Method
1	The test button caused the backend server to crash.	1	Front end was calling a method that didn't exist anymore.
2	The config file did not load, and everything crashed.	3	Correctly copy the default config file to the config file location.

Product	Game Front End Code Inspection		
Date	2/6/17		
Author	Marty		
Inspectors	Byron, Austin, Nick		
Defect #	Description	Severity	Correction Method
1	Failed to properly refactor a variable causing runtime errors.	1	Undid refactor and renamed manually.
2	Used a number variable in a String, {${variable}.png}, to grab the correct Card image to display in game. We only have 52 cards so this number should never go above 52 or below 0. Sometimes they did causing broken image links to display on on	4	Made a method to validate that the value is between [0,52].

	the site.		

Product	Database Code Inspection		
Date	2/6/17		
Author	Nick		
Inspectors	Byron, Austin, Marty, Nicky		
Defect #	Description	Severity	Correction Method
1	The connection to MongoDB did not work.	1	Change the mongoose connection method.
2	The code threw an exception when trying to create a mongoose item based off of our game schema.	2	Mongoose has different type names based on the version of mongoose and we were using depreciated types.

Product	Game Logic Code Inspection		
Date	2/6/17		
Author	Byron		
Inspectors	Nicky		
Defect #	Description	Severity	Correction Method
1	King could be “combined” with an empty hand.	3	Fixed the conditions in which cards in hand can be combined
2	Miscounted the number of cards left for each player.	1	Subtracted an additional 4 cards, one for each “0” card in each hand
3	Allowed movement of the “0” card.	1	Added checks to prevent movement of the “0” card

Product	Landing Page Code Inspection		
Date	2/6/17		
Author	Austin		
Inspectors	Marty, Austin, Nicky		
Defect #	Description	Severity	Correction Method
1	No requirements or checks on the nicknames being given.	3	iring a toForced user to give an appropriate nickname before playing (not null).
2	The code did not account for resizing of the window.	2	Ensured that the components interact appropriately with each other when resizing.

Testing Defect Log

When commits are pushed to Github, or a Pull Request is open, TravisCI automatically runs our test suite. Additionally, these tests, powered by Mocha and Karma, are automatically run whenever changes are made locally, so that no one will ever push broken code. Our tests cover the frontend, backend, as well as the socket communication layer.

Product	Backend Unit Tests		
Date	2/8/17		
Author	Nicky		
Inspectors	Nick, Byron, Marty		
Defect #	Description	Severity	Correction Method
1	The test cases found situations where a king could not be placed on top queen because of our algorithm. Input: `AreCardsSequential(queen-hearts, king-spades)` Output: false	1	We built a helper function so that a Queen and king would be seen as adjacent cards. Input: `AreCardsSequential(queen-hearts, king-spades)` Output: true
2	Clients were able to join games without being authenticated. Input: `JoinGame(currentClient)` Output: join games; false	2	When we were creating games, we were not calling the authentication method, we added that function call Output: “you cannot join this game”

Product	Database Unit Tests
Date	2/7/17
Author	Nick

Inspectors	Nicky, Byron, Marty		
Defect #	Description	Severity	Correction Method
1	The mongoDB was requiring a total amount of wins and games played to be fields when the user was trying to create a account. This prevented users from being able to create an account. Test: Trying to create any game Input: create game Output: crash	1	We removed the flag from the DB schema that required that field to be entered when the JSON file was stored. Test: Trying to create any game Input: create game Output: game starts
2	There were issues in storing the game data for replays. The schema was set up to store every move and some of the moves were being lost or incorrectly stored. Test: game-saving-test-3 Input: a seeded game Output: stored to DB acknowledgement and a copy of the move	2	Instead of storing each move in the DB we decided to store a snapshot of the board after every move and that fixed the problem Test: game-saving-test-3 Input: a seeded game Output: stored to DB acknowledgement and a snapshot copy
3	All of the data was being stored under a test Collection in the MongoDB versus a game or user collection. Test: game-saving-test-1	3	Fixed the DB connection so that it corresponded with the correct collection Test: game-saving-test-1

	Input: new Blank Game Output: stored to DB acknowledgement and a table in Test Collection		Input: new Blank Game Output: stored to DB acknowledgement and a table in Game Collection
4	During testing we realized that the game ID and the URL were the same field so we were doubling storing data Test: game-saving-test-1 Input: new Blank Game Output: stored to DB acknowledgement and a table in Test Collection	3	Removed the URL field from the DB schema Test: game-saving-test-1 Input: new Blank Game Output: stored to DB acknowledgement and a table in Test Collection