Second Iteration: Isshoni Sudoku

Created By **KERMit**
Emily Jin (ej2332), Meg Frenkel (mmf2171),
Riddhima Narravula (rrn2119), and Kundan Guha (kg2632)

Advanced Software Engineering - Fall 2020

Part 1: User Stories

In this iteration, we were able to complete all user stories listed in our initial proposal. We also implemented additional functionality beyond our initial user stories which are highlighted in blue below. These user stories are listed below:

1. As a player, I want to be able to fill in the sudoku so that I can solve the puzzle. My conditions of satisfaction are:
    a. I need to be able to start a new game from scratch
    b. I should be able to specify the difficulty level of a new game.
    c. I need to be able to place number and delete numbers throughout the game
    d. I need to be able to complete the sudoku puzzle and see that I have "won"
2. As a player, I want to be able to collaborate on a puzzle with another player so that I can have fun with my friends and family. My conditions of satisfaction are:
    a. I need to be able to invite another player to play in the game
    b. I need to be able to see any numbers that another player enters into the grid
    c. I should be able to delete the other player's move if I would like to change it
    d. Only one player can edit a cell at a time
    e. I should be able to see which cells other players are currently occupying and be able to identify which player is holding a cell.
    f. Once we have completed the sudoku puzzle, I need to see that we have "won"
3. As a player, I want to be able to save my current progress so that I can resume a game at a later time if need be. My conditions of satisfaction are:
    a. If I am playing the puzzle by myself and I log off, then I will see the puzzle as I left it when I log back in
    b. If I am playing the puzzle with my friends and I log off, but they are still working on it, then I will be able to see any progress they made while I was away when I log back in
    c. If I have multiple games in progress, I need to be able to select which one to resume when I log back on
    d. If I wish to delete a game from my homepage, I can do so and will not see that game on my homepage. Points earned from this game will still be reflected within my score.
4. As a player, I want to be able to see the leaderboard so that I can compare my puzzle solving scores with other players. My conditions for satisfaction are:

   a. If I complete a puzzle by myself, I am awarded the points associated with that puzzle, and my updated total puzzle solving score will be reflected on the leaderboard
   b. If I complete a puzzle with my friends, each member of our team is awarded the points associated with that puzzle, and each player's total updated puzzle solving score will be reflected on the leaderboard.
   c. I need to be able to see the leaderboard ranking with all players listed in descending order from most points to least points
5. As a player, I want to be able to chat with other players so that we can work together to solve the puzzle. My conditions of satisfaction are:
   a. At any point, I should be able to send a message via the chat box to all other players participating in the puzzle.
   b. If someone sends me a message, I should be able to see their message in the chat box section.
   c. I should be able to easily determine who sent each message in the chat box.
   d. If I logout of the puzzle application while other members are playing, I should be able to login and resume chatting. However, I will not be able to see past messages that were sent before I resumed the game.

Part 2: Equivalence Partitions and Boundary Conditions

Below are two tables outlining the equivalence partitions and corresponding test cases for the frontend and backend. Note that the equivalence partitions and corresponding test cases have been outlined for **major** methods of the project; smaller helper classes are not included, though they do have test cases. Not all methods had identifiable boundary conditions; if boundary conditions could be identified, the test case for the boundary is highlighted by the bold word **Boundary** for each case. Valid test cases are highlighted in green.

Here is an example of how to read this table. For the method `Puzzle.set_difficulty()` within the file `sudoku_puzzles.py`, we created a test file called `test_sudoku_puzzle.py` which has unit tests for the `Puzzle` class. The `set_difficulty()` method takes the difficulty level as an argument and attempts to set the difficulty level for a new `Puzzle` instance. For this method, we identified 4 equivalence classes, based on the static valid difficulty ranges set for the class:

(1) difficulty within the valid range of 0.01 <= difficulty <= 0.99,

(2) difficulty > 0.99, which is invalid,

(3) difficulty < 0.01, which is invalid, and

(4) difficulty provided as a non-float value (i.e, None or a String, etc.).

The boundary conditions for a valid input would be 0.01 and 0.99 and the boundary conditions for invalid input would be 0.0 and 1.0. Hence, we created tests corresponding to each of the different equivalence classes and boundary conditions (see test cases outlined below).

Automated test suite & script for the backend component of this project can be found here:

https://github.com/mmfrenkel/KERMit/tree/main/server/tests

https://github.com/mmfrenkel/KERMit/blob/main/server/bin/run_backend_tests.sh

Backend Equivalence Partitions and Test Cases:

Function/Method	Equivalence Partitions	Test Cases
`authentication.py` `verify_token()`		**Tests**: `test_authentication.py`
	Auth token is header is valid (valid case)	`test_authorize_token_validation_success()`
	Auth header entirely missing (invalid case)	`test_authorize_token_missing_header()` & `test_authorize_token_missing_header2()`
	Auth header is malformed; cannot parse token (invalid case)	`test_authorize_token_malformed_header()`
	Auth token in header is invalid (invalid case)	`test_authorize_token_validation_error()`
`authentication.py` `registration.post()`		Tests: `test_authentication.py`
	User not already registered in system (valid)	`test_register_user_not_registered()`
	User already registered in system (invalid)	`test_register_already_registered()`
	User has not released Google information (email and/or id are not accessible) (invalid attempt)	`test_register_missing_info_email()` `test_register_missing_info_id()`
`sudoku_puzzle.py` `Puzzle.set_difficulty()`		**Tests:** `test_sudoku_puzzle.py`
	Difficulty [0.01, 0.99] (valid cases)	`test_create_puzzle_valid_specification()`
	Difficulty is a non-float value (invalid)	`test_create_sudoku_puzzle_invalid_difficulty_str()` `test_create_sudoku_puzzle_invalid_difficulty_null()`
	Difficulty < 0.01 (invalid)	`test_create_sudoku_puzzle_invalid_difficulty_`

		`too_low()`
	Difficulty > 0.99 (invalid)	`test_create_sudoku_puzzle_invalid_difficulty_too_high()`
	Difficulty = 0.99 (Upper **Boundary** Condition)	`test_create_puzzle_valid_difficulty_upper_bound()`
	Difficulty = 0.01 (Lower **Boundary** Condition)	`test_create_puzzle_valid_difficulty_lower_bound()`
	Difficulty = 0.00 (Just Outside Lower **Boundary** - invalid)	`test_create_sudoku_puzzle_invalid_difficulty_too_low()`
	Difficulty = 1 (Just Outside Upper **Boundary** - invalid)	`test_create_sudoku_puzzle_invalid_difficulty_too_high()`
`sudoku_puzzle.py` `Puzzle.set_size()`		**Tests:** `test_sudoku_puzzle.py`
	Size within range [2, 5] (valid case)	`test_create_puzzle_valid_specification()`
	Size is a non-integer (invalid)	`test_create_sudoku_puzzle_invalid_size_str()` `test_create_sudoku_puzzle_invalid_size_float()`
	Size < 2 (invalid case)	`test_create_sudoku_puzzle_invalid_size_too_small()`
	Size > 5 (invalid case)	`test_create_sudoku_puzzle_invalid_size_too_large()`
	Size = 2 (Lower **Boundary** Condition - valid)	`test_create_puzzle_valid_size_lower_bound()`
	Size = 5 (Upper **Boundary** Condition - valid)	`test_create_puzzle_valid_size_upper_bound()`
	Size = 1 (Just Outside Lower **Boundary** - invalid)	`test_create_sudoku_puzzle_invalid_size_too_small_boundary()`
	Size = 6 (Just Outside Upper **Boundary** - Invalid)	`test_create_sudoku_puzzle_invalid_size_too_large_boundary()`
`Sudoku_puzzle.py` `is_complete_puzzle()`		**Tests:** `test_sudoku_puzzle.py`
	Puzzle is incomplete has null pieces (valid)	`test_check_for_completion_null_values()` `test_check_discrepancies_incomplete_puzzle()`
	Puzzle is incomplete and has incorrect values (valid)	`test_check_for_completion_incorrect_values()` `test_check_discrepancies_many_incorrect()`
	Puzzle is complete (valid)	`test_check_for_completion_complete()` `test_check_discrepancies_none()`
`Sudoku_puzzle.py` `set_point_value()`		**Tests:** `test_sudoku_puzzle.py`

	Difficulty and size correctly set (note that this function relies on set_size() and set_difficulty() to have valid parameters for the calculation. As a result, this method was tested for two sets of valid difficulty and size levels.	`test_set_point_value1()` `test_set_point_value2()`
`sudoku_puzzle.py` `Puzzle.update()`		**Tests:** `test_sudoku_puzzle.py`
	Valid board coordinates (0<=x<=4, 0<=y<=4) and within accepted value range [1-4] for a 4x4 puzzle where the piece is not static and puzzle is not completed.	`test_update_valid()`
	X-coordinate < 0 (invalid)	`test_update_invalid_coordinate_x2()`
	X-coordinate > 4 (invalid)	`test_update_invalid_coordinate_x1()`
	Y-coordinate < 0 (invalid)	`test_update_invalid_coordinate_y2()`
	Y-coordinate > 4 (invalid)	`test_update_invalid_coordinate_y1()`
	New value < 1 (invalid)	`test_update_invalid_value_too_small()`
	New value > 4 (invalid	`test_update_invalid_value_too_large()`
	X-coordinate = 0, Y-coordinate = 0, value = 1 (Lower **Boundary** All conditions - valid)	`test_update_valid_lower_bounds()`
	X-coordinate = 3, Y-coordinate = 3, value = 4 (Upper **Boundary** All conditions - valid)	`test_update_valid_upper_bounds()`
	X-coordinate = -1 (Outside Lower **Boundary** - invalid)	`test_update_invalid_coordinate_x2()`
	X-coordinate = 5 (Outside Upper **Boundary** - invalid)	`test_update_invalid_coordinate_x1()`
	Y-coordinate = -1 (Outside Lower **Boundary** - invalid)	`test_update_invalid_coordinate_y2`
	Y-coordinate = 5 (Outside Upper **Boundary** - invalid)	`test_update_invalid_coordinate_y1`
	New Value = 0 (Outside Lower **Boundary -** invalid)	`test_update_invalid_value_too_small()`
	New Value = 5 (Outside Upper **Boundary** - invalid)	`test_update_invalid_value_too_large()`
	Update attempted on static puzzle piece (cannot be updated - invalid)	`test_update_static_piece()`

	Update attempted on completed puzzle (invalid)	`test_attempt_update_complete_puzzle()`
`player.py` `PuzzlePlayer.add_player_to_puzzle()`		**Tests:** `test_player.py`
	Puzzle ID provided is for a puzzle with > 1 and < 4 players	`test_add_player_to_puzzle_ok_lower_bound()`
	Puzzle ID is for puzzle that has 1 player on it (**Boundary Condition - **valid)	`test_add_player_to_puzzle_ok_lower_bound()`
	Puzzle ID is for puzzle that has 3 players on it (**Boundary Condition - **valid)	`test_add_player_to_puzzle_ok_upper_bound()`
	Puzzle ID is for a puzzle with 4 players (Upper **Boundary** Condition) (invalid)	`test_add_player_to_puzzle_already_too_many_players()`
	Puzzle ID is for puzzle that does not exist; there are currently 0 players (Lower **Boundary Condition** - invalid)	`test_add_player_to_puzzle_player_doesnt_exist()`
`player.py` `PuzzlePlayer.get_top_players`		**Tests** `test_player.py` `test_leaderboard.py`
	Player limit is provided as an integer value > 0 (valid)	`test_get_leaders_completed_puzzles_limit2()`
	Limit is not an integer (invalid)	`test_get_top_players_str()` `test_get_top_players_none()`
	Limit is <= 0 (invalid)	`test_get_top_players_negative()`
	Limit = 0 (**Boundary** Condition - invalid)	`test_get_top_players_boundary()`
	Limit = 1 (**Boundary** Condition - valid)	`test_get_leaders_completed_puzzles_invalid_limit_1()`
`puzzle_pieces.py` `PuzzlePiece.update()`		**Tests:** `test_puzzle_pieces.py`
	Attempt to update a non-static piece with an integer value (valid)	`test_update_success()`
	Attempt to update a non-static piece with None (valid)	
	Attempt to update a static piece (invalid)	`test_update_fail_static()`
	Attempt to update a piece with a non-numeric value (invalid)	`test_update_fail_str()` `test_update_fail_float()`

`resources/ sudoku_puzzles.py` `SudokuPuzzles.get()`		**Tests:** `Test_sudoku_resources.py` **Note that there are no “invalid” cases here; either a player has puzzles or they do not.
	User has no puzzles (Lower **Boundary Condition** - valid)	`test_get_sudoku_puzzles_none()`
	User has one puzzle (valid)	`test_get_sudoku_puzzles_one()`
	User has multiple puzzles (valid)	`test_get_sudoku_puzzles_many()`
`resources/ sudoku_puzzle.py` `SudokuPuzzle.get()`		**Tests:** `Test_sudoku_resources.py`
	User asks for puzzle that is not associated with them (invalid)	`test_get_sudoku_puzzle_none_associated()`
	User asks for a puzzle that does not exist (invalid)	`test_get_sudoku_puzzle_none_retrieved()`
	User asks for a puzzle that they are associated with (valid)	`test_get_sudoku_puzzle_found()`
`resources/ sudoku_puzzles.py` `SudokuPuzzles.post()`		**Tests:** `test_sudoku_resources.py (unit tests)` `test_sudoku_endpoints.py (integration)`
	Zero additional players are specified (valid)	`test_create_puzzle_no_others()` `test_save_new_puzzle_valid()`
	< 4 additional distinct players are specified by email, none are registered (valid)	`test_create_puzzle_unregistered_other()` `test_save_new_puzzle_valid_unregistered_participant()`
	< 4 additional distinct players are specified by email, all are registered (valid)	`test_create_puzzle_registered_others()` `test_save_new_puzzle_valid_registered_participant()`
	< 4 additional players are specified by email, but emails within the list are duplicated (valid)	`test_create_puzzle_registered_duplicates()` `test_save_new_puzzle_duplicate_additional_players()`
	Size specification missing (invalid)	`test_save_new_puzzle_missing_size()`
	Difficulty specification missing (invalid)	`test_save_new_puzzle_missing_difficulty()`
	<4 additional players are specified by email, but list includes player himself (invalid)	`test_create_puzzle_self_added_as_additional_player()` `test_save_new_puzzle_user_added_themselves_as_additional_player()`
	> 4 additional players by email (invalid)	`test_sudoku_puzzles_too_many_additional()`
	0 additional players are attempted	`test_create_puzzle_no_others()`

	to be added to new game (Lower **Boundary Condition** - valid)	
	3 additional players are attempted to be added to new game (Upper **Boundary Condition** - valid)	`test_create_puzzle_registered_others()`
	4 additional players are attempted to be added to new game (Outer **Upper Boundary Condition** - invalid)	`test_sudoku_puzzles_too_many_additional()`
`resources/ sudoku_player.py` `SudokuPlayer.post()`		**Tests**: `test_sudoku_resources.py (unit tests)` `test_sudoku_endpoints.py (integration)`
	User attempts to join a puzzle that they have not joined and that does not have max number of players (valid)	`test_join_sudoku_puzzle()` `test_attempt_to_add_player_to_puzzle_valid()`
	User attempts to join a puzzle that they have already joined (invalid)	`test_join_sudoku_puzzle_already_joined()` `test_attempt_to_add_player_to_puzzle_already_in_puzzle()`
	User attempts to join a puzzle that they have not joined but that is already full (invalid)	`test_join_sudoku_puzzle_full()` `test_attempt_to_join_puzzle_max_players_reached()`
	User attempts to join a puzzle that does not exist (invalid)	`test_attempt_to_add_player_to_puzzle_that_doesnt_exist()`
`resources/ sudoku_puzzle.py` `SudokuPuzzlePiece.post()`		**Tests**: `test_sudoku_resources.py (unit tests)` `test_sudoku_endpoints.py (integration)`
	Valid move is attempted on a puzzle user is associated with (valid)	`test_sudoku_puzzles_add_move_success()`
	Move is attempted on a puzzle user is not associated with (invalid)	`test_sudoku_puzzles_add_move_not_associated()` `test_attempt_add_piece_player_is_not_affiliated_with()`
	Move is attempted on a puzzle user is associated with, but at an invalid location (see the `PuzzlePiece.update()` testing plan above for more granular boundary conditions, etc.)	`test_get_sudoku_puzzles_add_move_invalid()` `test_attempt_add_piece_invalid_position_high()` `test_attempt_add_piece_invalid_position_low()`
	Move is attempted on a puzzle user is associated with, but with an invalid value (see the `PuzzlePiece.update()` testing plan above for more granular boundary conditions, etc.)	`test_get_sudoku_puzzles_add_value_invalid()` `test_attempt_add_piece_invalid_piece_low()` `test_attempt_add_piece_invalid_piece_high()`
`resources/`		**Tests**:

sudoku_solution.py SudokuSolution.get()		test_sudoku_resources.py (unit tests) test_sudoku_endpoints.py (integration)
	User attempts to get a solution for a puzzle that they are not associated with (invalid)	test_sudoku_puzzles_get_solution_not_associated()
	User attempts to get a solution for a puzzle that does not exist (invalid)	test_sudoku_puzzles_get_solution_not_associated_2()
	User attempts to get a solution for a puzzle that they are associated with (valid)	test_sudoku_puzzles_get_solution() test_get_puzzle_solution_complete()
resources/ leaderboard.py Leaderboard.get()		**Tests**: test_leaderboard.py (unit tests) test_leaderboard.py (integration)
	Limit is unspecified, no one is on the leaderboard yet (valid)	test_get_leaderboard_no_leaders() test_get_leaders_no_completed_puzzle()
	Limit is unspecified, there are leaders on the leaderboard yet (valid)	test_get_leaderboard_leaders_exist() test_get_leaders_completed_puzzles()
	Limit is specified as integer > 0, there are leaders on the leaderboard yet (valid)	test_get_leaders_completed_puzzles_limit2
	Limit is specified as integer <= 0, there are leaders on the leaderboard yet (invalid)	test_get_leaders_completed_puzzles_invalid_limit_negative()
	Limit = 0 (Beyond Lower **Boundary Condition** - invalid)	test_get_leaders_completed_puzzles_invalid_limit_0()
	Limit = 1 (Lower **Boundary Condition** - valid)	test_get_leaders_completed_puzzles_invalid_limit_1()
	Limit is not an integer value. (invalid)	test_get_leaders_completed_puzzles_invalid_value()
player.py find_all_puzzles_for_player()		**Tests:** test_player.py
	Find all hidden puzzles (valid)	test_find_all_puzzles_for_player_hidden_only()
	Find all puzzles, hidden and visible specified (valid)	test_find_all_puzzles_for_player_visible_and_hidden()
	Find all puzzles, no setting specified (valid)	test_find_all_puzzles_for_player
resources/ sudoku_puzzle.py SudokuPuzzle.post()		**Tests**: test_sudoku_resources.py (unit tests) test_sudoku_endpoints.py (integration)

	Request to change puzzle visibility is made to an unassociated puzzle (invalid)	`test_set_puzzle_visibility_not_available()`
	Request to change puzzle visibility is made to an associated puzzle -- puzzle is set to hidden (valid)	`test_set_puzzle_visibility_hidden()`
	Request to change puzzle visibility is made to an associated puzzle -- puzzle is set to visible (valid)	`test_set_puzzle_visibility_visible()`

Frontend Equivalence Partitions and Test Cases:

The table below is in a similar format to the backend equivalence partitions and test case table. In general, it was more difficult to define boundary conditions/equivalence partitions since our input is often clicks or no input at all (ie. just viewing the page). Note that the tests file listed is located in the same directory as the component that it is testing. Automated test suite and script for the frontend component of this project can be run using commands specified in the package.json file:

https://github.com/mmfrenkel/KERMit/blob/main/app/package.json

Component	Equivalence Partitions	Test Cases
`app/src/components/Header/Header.js`		**Tests**: `Header.test.js`
View cases (no invalid cases since user can only be logged in or logged out):	When logged in, header shows all navigation options (e.g. nav item to home page and nav item to leaderboard)	`test('header shows all options when logged in')`
	When logged out, header shows limited options (can only view title and log in button)	`test('header shows less options when not logged in')`
Click cases (when user clicks on nav items or on component)	Clicking on homepage nav-item results in correct redirect to home page	`test('clicking on my puzzles redirects to homepage')`
	Clicking on leaderboard nav-item results in correct redirect to home page	`test('clicking on leaderboard redirects to leaderboard')`
	Clicking on items other than the two nav-items does not result in a redirect (invalid case)	`test('clicking outside nav elems does not redirect')`
`app/src/components/HidePuzzleModal/HidePuzzleModal.js`		**Tests**: `HidePuzzleModal.test.js`
Click cases (only two options, no invalid cases or boundary conditions)	Clicking on 'yes' button triggers correct prop functions	`test('clicking yes calls hidePuzzle')`
	Clicking on 'no' button triggers correct prop functions	`test('clicking no calls setModalOpenStatus')`
`app/src/components/PuzzleCard/PuzzleCard.js`		**Tests**: `PuzzleCard.test.js`
View cases (puzzle can renders with different params passed in, no valid cases since params are not inputted directly by	When puzzle passed in is complete, renders complete message on puzzle card	`test('renders correct text according to params')`

user)		
	When puzzle passed in is incomplete, renders in progress message on puzzle card	`test('renders incomplete message')`
	Renders other params correctly as passed in	`test('renders correct text according to params')`
Click cases (user can click on entire card or on hide button, no valid cases since there are only two options)	When user clicks on entire card, passed in onClick function is called	`test('puzzle card click calls function')`
	When user clicks on hide button, passed in onClick function is **not** called but setHideModalPuzzleStatus and setHidePuzzleId are called	`test('hide button click calls functions')`
`app/src/components/SudokuBoard/SudokuBoard.js`		**Tests**: `SudokuBoard.test.js`
	When the client is waiting on getting the gridstate response from the server, the board will render a “Loading puzzle” message (valid edge case)	`test(‘shows loading text when the puzzle is loading’)`
	When the puzzle pieces have successfully been returned to the client, the board will render the puzzle grid	`test(‘shows puzzle grid when the puzzle has loaded’)`
	When a user clicks on a cell in the grid, it emits locking event telling the backend to lock the cell at the location specified	`test(‘emits add_lock event when nonempty number is entered’)`
	When a user clicks the “Check Solution” button, the solution endpoint is called	`test(‘calls the solution endpoint when then Check Solution button is clicked’)`
	When a user clicks the “Check Solution” endpoint and there are mistakes in the grid, the user sees a “Something’s not right...” message	`test(‘shows that the puzzle has errors when the check solution button is clicked, but the grid contains mistakes’)`
	When a user wins a game, a “You win!” message is displayed	`test(‘shows that you win when the puzzle is solved’)`
	When no puzzle number (puzzle id) is specified, the page will not render anything (invalid case)	`test(‘shows empty page when the puzzle number is undefined’)`
`app/src/components/SudokuCell/SudokuCe`		**Tests**: `SudokuCell.test.js`

ll.js		
	When a user clicks on a cell, it “locks” the cell so that other players cannot edit it	test(‘adds lock when cell is focused’)
	When a user leaves a cell, it unlocks the cell so that other players can now edit it	test(‘removes lock when cell is blurred’)
	The cell renders with a slightly thicker left edge to differentiate smaller 3x3 grids in the board	test(‘has thicker line styling on the left edge for every third column’)
	The cell renders with a slightly thicker top edge to differentiate smaller 3x3 grids in the board	test('has thicker line styling on the top edge for every third row')
`app/src/pages/Home/HomePage.js`		**Tests**: `HomePage.test.js`
View cases	When the user has no puzzles, the page renders with a “You do not currently have any puzzles” message (**Boundary condition** - valid)	test('renders no puzzles message')
	When the user has more than one puzzle, the page renders with one puzzle card per puzzle	test('renders puzzle cards for user with >0 puzzles')
	When the response to the get puzzles endpoint returns with an error, the page renders with a “You do not currently have any puzzles” message (Invalid case)	test('renders empty page if response undefined')
Click cases (no invalid cases)	Clicking on the delete button for a specific puzzle opens the hide puzzle modal	test('clicking delete button on puzzle opens modal')
	Clicking yes on the hide puzzle modal triggers a POST request to change the visibility of the puzzle	test('clicking yes on hide modal triggers fetch')
	Clicking yes or no on the hide puzzle modal closes the modal	test('clicking yes closes hide modal') test('clicking no closes hide modal')
	Clicking no on the hide puzzle modal does not trigger a POST request	test('clicking no on hide modal does not call fetch')
	Clicking the create game button opens the create puzzle modal	test('create game button opens create modal')
	Creating a game using the modal triggers a POST request to create a puzzle	test('creating game with modal calls fetch')
	Creating a game using the model	test('creating game with modal redirects')

	triggers a redirect to the puzzle page for the created puzzle	
`app/src/pages/Leaderboard/LeaderboardPage.js`		**Tests**: `LeaderboardPage.test.js`
	If there are no players that have completed a game, then the leaderboard page renders an empty leaderboard message (Boundary condition - valid)	`test('renders with empty leaderboard message')`
	If there is more than one player who has completed a game, the leaderboard page renders a grid containing those players	`test('renders with DataGrid table’)`
	If request to get top players returns with an error, renders with a empty leaderboard message (Invalid case)	`test('renders empty with undefined response')`
`app/src/pages/Puzzle/PuzzlePage.js`		**Tests**: `PuzzlePage.test.js`
	When there are multiple players in a game, the page will render the chat along with the puzzle	`test(‘shows the chat in multiplayer mode’)`
`app/src/components/Chat/Chat.js`		**Tests**: `Chat.test.js`
View cases	Renders no messages if the message list is empty (boundary case - valid)	`test('renders no messages')`
	Renders one message	`test('renders one message')`
	Renders >1 messages	`test('renders list of messages')`
Text entry cases	When non-empty text is entered and the user presses the enter key, a socket emit event is triggered	`test('emits socket event when non-empty message entered')`
	When the textbox does **not** currently have any text and the user presses the enter key, a socket emit event is not triggered (invalid case)	`test('does not emit socket event when empty message entered')`
	If the user types in text but does not press the enter key (e.g. presses space or some other key), a socket emit event is not triggered (invalid case)	`test('does not emit socket event if key pressed is not enter')`
`app/src/components/Chat/ChatMessage.js`		**Tests**: `ChatMessage.test.js`

View cases (no invalid or boundary cases)	If message was sent by the current user, the user name does not appear above the message	`test('does not render username when user message')`
	If message was not sent by the current user, the user name does appear above the message	`test('renders username when not user message')`
`app/src/components//CreatePuzzleModalContent/CreatePuzzleModalContent.js`		**Tests**: `CreatePuzzleModalContent.test.js`
	User can select difficulty from modal (set difficulties available so no invalid cases)	`test('can select difficulty on modal')`
	User can input email into modal (accepts any string, error handling done on backend)	`test('can input email on modal')`
	User is able to create a game without filling in email textbox (optional parameter)	`test('submit calls createGame')`

Part 3: Coverage

3.1 Backend Coverage

Code coverage for the backend was completed using the coverage tool. Backend branch coverage upon completion of iteration two was ~100%. The script to run the coverage reports and the reports themselves can be found in these directories, respectively:
https://github.com/mmfrenkel/KERMit/blob/main/bin/run_backend_coverage.sh
https://github.com/mmfrenkel/KERMit/tree/main/reports/backend/coverage

Example early coverage report:
(Note that early on, coverage reports did not check for branch coverage yet)
https://github.com/mmfrenkel/KERMit/blob/main/server/reports/coverage/all-2020.11.28-13.35.10.txt

Example final coverage report:
(In our final coverage tests, we check for branch coverage)
https://github.com/mmfrenkel/KERMit/blob/main/server/reports/coverage/all-2020.12.02-17.12.04.txt

3.2 Frontend Coverage

Code coverage for the frontend was completed using the coverage tool. The script to run the coverage reports in the package.json file:

https://github.com/mmfrenkel/KERMit/blob/main/app/package.json

The reports themselves can be found in the following directory:
https://github.com/mmfrenkel/KERMit/tree/main/app/reports

Example **early** coverage report:
https://github.com/mmfrenkel/KERMit/blob/main/app/reports/incomplete_frontend_coverage_report.txt

Note that we excluded some components from the coverage report. GoogleBtn.js uses an external public component so we did not test the external component's functionality. Additionally, index.js and reportWebVitals.js are files that were initialized with the project and do not have any functionality that we added. Example **final** coverage report:
https://github.com/mmfrenkel/KERMit/blob/main/app/reports/recent_frontend_coverage_report.txt

Part 4: Continuous Integration

We chose to use TravisCI as a CI tool. Our `.travis.yml` file can be found here:
https://github.com/mmfrenkel/KERMit/blob/main/.travis.yml

Our repository is public and CI reports may be viewed here:
https://travis-ci.com/github/mmfrenkel/KERMit/builds

Example CI reports are linked here. Note that test output is visible in the CI logs:
https://travis-ci.com/github/mmfrenkel/KERMit/builds/206920830 (failing)
https://travis-ci.com/github/mmfrenkel/KERMit/builds/206921615 (passing)

Backend unit test results from TravisCI automatically populate the travis_reports/ directory in the travis_results branch: https://github.com/mmfrenkel/KERMit/tree/travis_results/travis_reports