Group Members: Kevin Li, JiongHeng Li, Kevin Wu, Tri Le, Sahithi Pisupati							
	ID	**Title**	**Priority**	**Description**	**Expected Results**	**Actual Results (PASS/FAIL)**	**Comments**
Phase 1: Starting Game	1.1	Create New Game	High	User starts the application session. User then clicks on Create Game button.	A unique 6-character game code is generated and displayed. User is also directed to the Play page.	PASS	
	1.2	Join Existing Game	High	User click on the Join button and enter a code to be able to join an existing game session someone else has created	User is able to connect to an existing lobby and is directed to the Play page.	PASS	
	1.3	Show About Screen	Low	User clicks on the About button in the main menu	A popup comes up and shows the about screen containing information about the game checkers	PASS	
	1.4	Show Credits Screen	Low	User clicks on the Credits button in the main menu	A popup comes up and shows the credits screen containing information about how this project was created	PASS	
Phase 2: Game Mechanics	2.1	Move piece	High	User clicks on a piece and clicks on a highlighted square to move to that piece	The piece should move to that square when it is clicked, given that it is vacant. Pawns can move in any forward diagonal direction, towards the opponent's back rank. Kings can move in any diagonal direction.	PASS	
	2.2	Capturing piece	High	User captures an opponent piece by clicking on their own piece and clicking on the square behind an opponent's piece if the opponent is next to the user's piece and there is an unoccupied square behind the opponent piece	This move moves the user's piece behind the opponent piece and the opponent's piece is removed from the game.	PASS	
	2.3	Promotion	High	User move to the end of the board to upgrade their piece to a king	When a user gets their piece to the opposite end of the board, the piece is promoted to a king.	PASS	
	2.4	Take Turns	High	User moves when it is their turn as indicated, otherwise they wait for their opponent to finish their turn. This keeps double-jumps and chain-jumps in mind.	Text will be displayed to indicate whether it is the user's turn or the opponent's turn.	PASS	
	2.5	Display possible moves	Medium	When clicking on a piece, possible spots that the piece can move onto will be highlighted	When clicking on one of the user's pieces, spots on the board will be highlighted for all possible valid moves the player can make. After taking a move or clicking on another piece or area of the website, the highlight will disappear.	PASS	
	2.6	Invalid move	High	User clicks on a piece. User then clicks on a square representing an invalid move.	User cannot click on an invalid square. Game board will not change, and a move will not take place	PASS	
	2.7	Win	High	If the opponent's pieces are all captured from the board, a display will show that the opponent loses and the user wins.	A display message shows up. The loser and winner is presented with a losing or winning message, respectively	PASS	
	3.1	Move Timer	Medium	User must make a move within sixty (60) seconds, or they will automatically forfeit.	A timer is displayed on the page. When 60 seconds pass with no move made, the opponent loses and the user wins. The game result is indicated, as per *3.5 Indicate Game Result.*	FAIL	

Phase 3: Game Session	3.2	Display Board	High	User sees the game board, which is up-to-date with the current position	When the game state updates, each user has their board in their web browser updated with the latest game state.	PASS	
	3.3	Themed Pieces	Low	The game board has a theme: Attack on Titan. Pieces and the gameboard follow this theme	The gameboard and checker pieces are Attack on Titan themed	PASS	
	3.4	Board Generation	High	Upon the start of a game session, the game board should be populated with the checkers starting position	The black squares on the three (3) rows on the user and opponent sides are populated with their respective checker pieces.	PASS	
	3.5	Indicate Game Result	High	The game ends when either player has run out of pieces to move or forfeits.	When a player has run out of pieces to move or forfeits, the winner is indicated on both players' screens. If a draw is offered and accepted, then there is no winner and a draw is indicated.	PASS	
	3.6	Forfeit	Medium	User is able to forfeit the game at any time.	Upon indicating that the user would like to forfeit (with a button), game flow defers to the 3.5 Indicate Game Result test	FAIL	Not implemented
	3.7	Offer Draw	Low	User is able to offer a draw	Upon indicating the user would like to draw (with a button), the opponent can either accept this draw, in which the game ends. Otherwise, the opponent ignores/declines the offer and the game continues.	FAIL	Not implemented
	3.8	Rematch	Low	Upon the end of a game, both players can offer a rematch.	When the game ends, either player can send a rematch request (with a button), and the other user is able to accept or reject this request.	FAIL	Not implemented
Phase 4: Server connection	4.1	User disconnect pre-game	Low	User creates/joins a game. Before game can start, user disconnects from game lobby.	Since game has not started, the connected user will remain in game lobby, and continue waiting for a second user.	PASS	
	4.2	User disconnect mid-game	Medium	User creates/joins a game. After game already starts, user disconnects from game.	Since game has already started, the connected user is declared the winner.	PASS	
Unit Tests (White-Box):							
Data_Storage	A.1	getGames()	High	Preconditions: firebase is connected, Post conditions: received games are accurate	Test every statements and branches to make sure that games are correctly received	PASS	
Data_Model	B.1	getGameById(string)	High	Precoditions: argument not null, must be a string, Post conditions: received game object structure is accurate	Test every statement and branch to make sure the code runs given all the possible conditionals	PASS	
	B.2	getOpenGames()	High	Precondition: firebase is connected. Post condition: the client side receives correct number of open games	Test every statement and branch to make sure the open games are correctly received	FAIL	Not implemented
	B.3	getPlayerById(string)	High	Precoditions: argument not null, must be a string, Post conditions: received player object structure is accurate	Test every statement and branch to make sure the player id is correctly received	PASS	
	B.4	connectPlayerToGame (playerId:string,gameId: string)	High	Preconditions: playerId and gameId must be a string and not null. Post condition: Player is connected to the game	Test every statement and branch to make the player is successfully connected to the game	PASS	
CheckerBoard	C.1	getWhite()	High	Preconditions: game is loaded. Post-conditions: returns the correct white player	Test every statement and branch to make the correct white player is received	PASS	
	C.2	getBlack()	High	Preconditions: game is loaded. Post-conditions: returns the correct black player	Test every statement and branch to make the correct black player is received	PASS	

	C.3	getSquareAtPosition (BoardPosition	High	Preconditions: BoardPosition is valid. Post-conditions: returns the correct square	Test every statement and branch to make sure the correct square is received	PASS	
Square	D.1	getColor()	High	Preconditions: square is valid. Post-conditions: returns the correct color	Test every statement and branch to make sure the correct color is received	PASS	
	D.2	getPiece()	High	Preconditions: square is valid. Post-conditions: returns the correct piece	Test every statement and branch to make sure the correct piece is received	PASS	
	D.3	getPosition()	High	Preconditions: square is valid. Post-conditions: returns the correct position	Test every statement and branch to make sure the correct position is received	PASS	