Requirement ID	Requirement Description	Status
US1	As a player, I want to start a new game with an existing gameboard.	Implemented
US2	As a player, I want to name the players with input or randomly generated strings to better differentiate the players.	Implemented
US3	As a player, I want to see the status of any specific player and all players.	Implemented
US4	As a player, I want to see the status of the game, including the squares and the players' positions on the gameboard.	Implemented

US5	As a player, I want to query the next player.	Implemented
US6	As a player, I want to save the current game to a file.	Implemented
US7	As a player, I want to load a game from a file and continue the game.	Implemented
NEWUS8	As a gameboard designer, I want to design a new gameboard based on an existing gameboard by modifying the property squares (including their name, price, and rent).	Implemented
US9	As a gameboard designer, I want to load an existing gameboard and customize it by modifying its squares.	Implemented

US10	As a gameboard designer, I want to save the gameboard I designed.	Implemented
REQ-001	The game shall process the user input to load the corresponding gameboard.	Implemented
REQ-002	The game shall prompt the user to enter the number of players and the name of the players, which are used to instantiate the players.	Implemented
REQ-003	The game shall process the user input to load the corresponding saved game file and continue the game.	Implemented
REQ-004	The game shall include a function allowing players to simulate the action of throwing two four-sided dice and use the output to modify player position data.	Implemented

REQ-005	The game shall include a function that is able to output the data of each player and each square for every player at any round.	Implemented
REQ-006	The game shall automatically initialize the money as HKD 1500 and property as an empty list for all players.	Implemented
REQ-007	The game shall initialize the position value as 1 (the first square) for all players.	Implemented
REQ-008	The game shall map the players' position to correct indices of squares, following the clockwise movement and returning to square 1 after reaching square 20.	Implemented
REQ-009	The game shall include the function allowing the player to purchase unowned properties and automatically decreasing the money according to the property's price.	Implemented

REQ-010	The game shall include the function to automatically decrease the property's rent from the money of a player A who landed on a property owned by another player B, while increasing the rent to the money of player B.	Implemented
REQ-011	The game shall include the function to automatically deduct the player money by 10% (rounded down to the nearest 10) when landing on the Income Tax square.	Implemented
REQ-012	The game shall include the function to randomly select a number in multiples of 10 from a range of -300 to 200 (both inclusive) when the player landing on a Chance square, representing both losses and gains. The maximum gain is HKD200, while the maximum loss is HKD300.	Implemented
REQ-013	The game shall include a function to automatically add HKD1500 to player money when the player passed or landed on the Go square.	Implemented
REQ-014	The game shall have no effect on the player's state (money, properties, and position) when landing on the Free Parking square.	Implemented

REQ-015	The game shall include the function to automatically change the player’s position to the Jail square and change the player status to “In Jail” when landing on the Go to Jail square.	Implemented
REQ-016	The game shall have no effect on the player’s state (money, properties, and position) when landing on the Jail square and without “In Jail” status.	Implemented
REQ-017	The game shall include the function to let the player choose to decrease the player’s money by HKD150 or roll doubles to get rid of the “In Jail” status. The player shall have three attempts to roll doubles; if they fail to do so within those attempts, their money will be automatically decreased by HKD150, and they will advance steps equivalent to their third dice throw from "In Jail" status.	Implemented
REQ-018	The game shall include a function to monitor the player's money. Once the data becomes negative, the player will be automatically retired from the game and the occupied properties will be released.	Implemented
REQ-019	The game shall include a function to determine the end of the game, either when only one player remains or when the number of rounds reaches 100.	Implemented

REQ-020	The game shall include a function to determine the winner (s), either by identifying the last remaining player, or by comparing the total amount of money of players at the end of the game. The winner is the one with the most amount of money in the latter case. If multiple players own the same amount of money, they are all listed as winners.	Implemented
REQ-021	The game shall include the function to save all players' data and the gameboard of the current game into a file.	Implemented
REQ-022	The game shall include the function to load the previously saved game data and continue the game.	Implemented
REQ-023	When there are finite available options to be chosen by players, the game shall print out all available user options on the command line for users to check and choose.	Implemented
REQ-024	The game shall include exception handlers for each possible exception and print out the exception message, including what should be done next to handle it.	Implemented

REQ-025	The game shall include the function for the gameboard designers to create a customized gameboard by editing the given property's data, including name, price and rent.	Implemented
REQ-026	The game shall include the function for the gameboard designer to review the updated list of properties (if any) of the gameboard.	Implemented
REQ-027	The game shall develop the function for the gameboard designers to save the gameboards they modified.	Implemented
REQ-028	Every function and class should have accompanied documentation, with a minimum of 80% code coverage for necessary inline comments and explanations.	Implemented
REQ-029	The command-line interface of the game should be intuitive, with clear instructions and navigation on how to play and interact with the game such that a player or designer could use no more than 5 minutes to be able to understand and execute all operations with no more than 5 mistakes.	Implemented

REQ-030	The game should automatically handle and recover from unexpected errors 95% of the time, providing a clear error message and guidance for recovery.	Implemented