PROJECT NAME		PRIORITY
Chess Game		High
DESCRIPTION	VERSION	TEST DATE
SDL based chess game	1.1	5/Apr/2025
TEST OBJECTIVE		
Verify user requirements (sprint 1)		

Test Case ID	Test Steps	Input Data	Expected Results	Actual Results	Test Environment	Execution Status	Bug Severity	Bug Priority	Notes
1	1. Launch the game. 2. Navigate to the main menu. 3. Select the “Play” option. 4. Start the game and allow each player to make moves in turn.	Players make moves alternately using mouse clicks to select and move pieces.	Game starts with two players (White and Black). - Player 1 (White) can make the first move. - After Player 1 moves, turn switches to Player 2 (Black). - Players can make legal moves in turn until the game ends.	- Game launched successfully. - “Two Player” mode selected without error. - White pieces were controlled by Player 1. - Black pieces were controlled by Player 2. - Turns switched correctly after each move. - Game continued smoothly with no crashes or logic errors.	Windows 11	PASS	N/A	N/A	No issues Found.
2	1. Launch the game. 2. Start a new game. 3. Attempt to move a piece to an illegal square	- Attempt to move a bishop straight forward. - Try moving a pawn backward. - Attempt to move during the	- Illegal moves are blocked and not executed. - no action occurs when attempting an illegal move.	- Game blocked bishop's invalid straight move. - Pawn could not be moved backward. - Input during	Windows 11	PASS	N/A	N/A	No issues Found.

	(e.g., move a bishop like a rook, or move through other pieces). 4. Attempt to move a piece when it’s not that player’s turn. 5. Try a legal move for comparison.	other player’s turn. - Attempt a valid knight move (L-shape).	- Only valid moves are allowed for each piece type. - Turn-based play is enforced (only one player can move at a time).	opponent's turn was ignored. - Legal knight move was successful. - All illegal move attempts were prevented without crashing or graphical glitches.					
3	1. Launch the game. 2. Observe the main game screen after starting a new game. 3. Check for proper rendering of the 8x8 board and all chess pieces. 4. Compare piece layout to standard initial positions. 5. Make a move and verify the piece updates visually.	Start the game normally without extra user input beyond launching it.	- A standard 8x8 chess board is displayed. - All 32 pieces appear in correct starting positions. - Each piece is visually distinct (knight, rook, bishop, etc.). - When a piece moves, its new position is updated graphically.	- Game window opened with an 8x8 chess board clearly visible. - All pieces rendered with correct sprites/images. - Pieces appeared in correct initial positions. - Visuals updated correctly after making a move (e.g., pawn from e2 to e4). - No visual glitches or missing sprites encountered.	Windows 11	PASS	N/A	N/A	Color scheme of home screen and game screen do not match(pending for sprint 2)else issues Found.
4	1. Launch the game. 2. Start a new game. 3. Click on a piece (e.g., a white pawn). 4. Click on a valid destination square. 5. Repeat with other pieces and check interaction.	- Mouse click on a white pawn. - Mouse click on a valid move square (e.g., e2 → e4). - Mouse click on an empty square or opponent's piece during Player 1's turn.	- Chessboard is rendered graphically. - Pieces can be selected by clicking. - After clicking a piece, valid moves are allowed via further clicks. - Invalid clicks are ignored or trigger a small error notification. - Visual feedback may be shown.		Windows 11	PASS	N/A	N/A	No issues Found.

				- Graphical board was visible on launch. - Clicking a piece allowed it to move. - placing piece on a valid square moved the piece correctly. - Invalid clicks (e.g., empty square or wrong turn) did nothing, as expected. - Mouse input worked reliably across all tested pieces.					
5	1. Launch the game. 2. Start a new game and play several moves involving all types of pieces (pawn, rook, knight, bishop, queen, king). 3. Track responsiveness after each move (especially from mid- to late-game). 4. Test game behavior when many pieces are on the board and during complex move scenarios (e.g., checks, captures, promotions).	- Sequence of legal moves across various scenarios (opening, midgame, endgame). - Complex board states involving multiple potential legal moves.	- Game maintains fast and responsive move validation. - No noticeable lag when selecting or moving pieces. - Moves scales well as the board state becomes more complex. - Internal structures like 2D arrays or vectors are used efficiently (verified by code review).	- All moves executed with no noticeable delay. - System responded instantly during multiple phases of the game. - Complex board positions (e.g., multiple checks, pinned pieces) were processed quickly. - Code inspection confirmed usage of optimized structures like 2D arrays for board state and vectors for move lists.	Windows 11	PASS	N/A	N/A	No issues Found.

6	1. Launch the game. 2. Observe the layout of the main menu and game screen. 3. Click on a piece and observe selection 4. make a move and observe visual cues. 5. Make an illegal move and observe feedback.	- Click on pieces and squares. - Attempt legal and illegal moves.	- UI elements are clearly labeled and positioned. - Piece selection provides clear visual feedback (piece starts to move with mouse). - Invalid moves provide feedback (no move).	- Main menu and in-game UI were clean and intuitive. - Clicking on a piece made it move. - Invalid move attempts had no disruptive effects, just ignored as expected. - All feedback was consistent and user-friendly.	Windows 11	PASS	N/A	N/A	Further Ui elements to be added in sprint 2

7	1. Launch the game. 2. Start a new game and make a series of moves (both legal and illegal). 3. After a few moves, review the debug terminal 4. Check the order, accuracy, and correctness of the recorded moves. 5. Test scenarios where moves are made quickly or out of order to ensure the log is accurate.	- Make legal and illegal moves alternately. - Use different pieces (pawn, rook, knight, etc.). - Check the game terminal after multiple moves.	- Moves are logged in the correct sequence with no errors. - terminal include sufficient data to trace each move (piece moved, from square, to square).	- Moves were correctly logged in sequence, including player turns, starting and ending positions. - Log data was accurate, and each move was traceable. - No errors in move logging were observed.	Windows 11	PASS	N/A	N/A	No issues Found.

This suite of test cases covers the core functionalities of the chess game, ensuring both the game mechanics and user experience meet expected standards.

1. Two-Player Mode Functionality: Verifies that the game allows two players to play against each other, with correct turn-taking and interaction between Player 1 and Player 2.
2. Legal Move Enforcement: Ensures that the system enforces the rules of chess, blocking illegal moves and only allowing valid ones based on piece types and turn order.
3. Visual Display of Game Board and Pieces: Confirms that the game board and chess pieces are displayed correctly, with accurate initial positions and real-time updates during gameplay.
4. Clickable Graphical Chessboard: Tests the interactive elements of the UI, confirming that players can click on pieces and valid move squares to make moves, with proper visual feedback.
5. Optimized Data Structures for Move Calculations: Ensures the internal data structures, like 2D arrays or vectors, are used efficiently to speed up move validation and game state management, ensuring smooth gameplay even with complex board positions.
6. Intuitive UI with Clear Visual Feedback: Verifies that the user interface is easy to use, with clear visual feedback for piece selection, valid/invalid moves, and game status (e.g., check, checkmate).
7. User Move Logging Integrity: Ensures that all user moves are logged correctly, maintaining a secure and unaltered move history to prevent manipulation. This ensures fair play and traceability.

The AI opponent will be implemented in Sprint 2. This functionality will allow players to play against the computer, and we will develop test cases around AI behavior, decision-making, difficulty levels, and response time in future sprints.