|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case “Create New Game”**   1. **Use case code**   UC001   1. **Brief Description**   This use case describes the interaction between the Player and Chess Royale when the player wishes to create a new game session.   1. **Actors**  * Player  1. **Preconditions**  * The player is logged into Chess Royale and is at the main menu. * No active game session exists.  1. **Basic Flow of Events** 2. The player selects the “Create New Game” option from the main menu. 3. The software displays a prompt asking whether the player wants to start with a custom board position or a normal starting board. 4. The player enters the desired starting board option (see Table A). 5. The software validates the selection and initializes a new game session with the chosen board configuration. 6. The software calls the “Select AI Personality” use case to display available AI personalities. 7. The software confirms the selected AI personality and updates the game session settings accordingly (see Table B). 8. The software displays the selected starting position, and the AI opponent is activated. 9. **Alternative flows**   **Table N-Alternative flows of events for UC Place order**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **No** | **Location** | **Condition** | **Action** | **Resume location** | |  | Step 2 | The player cancels the board option prompt | The software aborts game creation and returns to the main menu | Use case ends | |  | Step 3 | The player enters an invalid board option | The software displays an error message and re-prompts for a valid selection | Returns to Step 3 | |  | Any step | A system error occurs | The software displays an error message and aborts game creation | Use case ends |  1. **Input data**   **Table A-Input data of …**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **No** | **Data fields** | **Description** | **Mandatory** | **Valid condition** | **Example** | |  | Create Game Command | Command indicating the initiation of game creation | Yes | Valid button | "Create New Game" | |  | Starting Board Option | Option selected by player for board setup | Yes | Must be "Custom" or "Normal" | "Normal" |  1. **Output data**   **Table B-Output data of …**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **No** | **Data fields** | **Description** | **Display format** | **Example** | |  | Game Session ID | Unique identifier for the newly created game session | Text/Number | "CHR-2023-001" | |  | Initialized Chess Board | Graphical display of the chess board with chosen position | Graphical UI | Board shown with normal or custom starting layout | |  | Confirmation Notification | Message confirming successful game creation and AI selection | Text message | "Game created with Aggressive AI" |  1. **Postconditions**  * A new game session is active with the chess board initialized to the selected starting configuration. * The selected AI personality is active as the opponent. * The game session is recorded for further moves and potential saving. |

**Use Case “Create New Game”**

1. **Use Case Code**  
   UC001
2. **Brief Description**  
   This use case describes the interaction between a Player and the Chess Royale app when the player starts a new chess game session.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • The player is at the main menu or dashboard.  
   • No active game session exists or the player has chosen to abandon the current session.
5. **Basic Flow of Events**
   1. The player selects the “Create New Game” option from the main menu.
   2. The software initializes a new game session, setting up the chess board with default settings.
   3. The software displays the empty chess board and initial game state to the player.
   4. The system confirms that a new game session has been successfully created.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 2 | Initialization failure (e.g., resource error) | The software displays an error message and aborts the game creation process | Use case ends |
| 2 | Step 3 | Player cancels the new game before board loads | The system aborts the initialization and returns to the main menu | Main menu |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | Create Game Command | Command indicating the creation of a game | Yes | Valid button click event | Player clicks “New Game” |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Game Session ID | Unique identifier for the new game session | Text/Number | "GameID: CHR12345" |
| 2 | Chess Board | Visual representation of the starting board | Graphical display | Chess board with default layout |
| 3 | Confirmation Notification | Message indicating successful game creation | Text message | "New game started successfully" |

1. **Postconditions**  
   • A new game session is active with an initialized chess board.  
   • The player is ready to proceed to subsequent actions (e.g., selecting AI personality).

**Use Case “Select AI Personality”**

1. **Use Case Code**  
   UC002
2. **Brief Description**  
   This use case describes the interaction where the player selects an AI personality that will determine the style of the AI opponent for the game.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session has been created (or is in the process of being created).  
   • The AI personality selection menu is available to the player.
5. **Basic Flow of Events**
   1. The player navigates to the “Select AI Personality” menu.
   2. The software displays a list of available AI personalities (e.g., Aggressive, Defensive, Positional, Random).
   3. The player selects one of the AI personalities.
   4. The software confirms the selection and updates the game settings accordingly.
   5. The updated game state, reflecting the chosen AI personality, is displayed.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 3 | Player cancels selection | The software retains the previous default AI personality | Use case ends |
| 2 | Step 4 | System error in updating settings | The software displays an error message and prompts for re-selection | Step 2 or 3 |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | AI Personality Selection | Chosen AI personality from the list | Yes | Must be one of the listed options | "Aggressive" |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Confirmation Notification | Message confirming successful selection | Text message | "Aggressive AI selected" |
| 2 | Updated Game Settings | Updated game configuration reflecting the choice | Data summary | "AI Mode: Aggressive" |

1. **Postconditions**  
   • The game session is updated with the chosen AI personality.  
   • The AI opponent will behave according to the selected personality in subsequent moves.

**Use Case “Make a Move”**

1. **Use Case Code**  
   UC003
2. **Brief Description**  
   This use case describes the interaction where the player makes a chess move during an active game session.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session is active with an initialized chess board.  
   • It is the player’s turn to move.
5. **Basic Flow of Events**
   1. The player selects a chess piece on the board.
   2. The player selects the destination square for the move.
   3. The software validates the move against chess rules.
   4. The software updates the game board with the new position.
   5. The move is recorded in the game history.
   6. The turn passes to the AI opponent (or to the opponent if in multiplayer).
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 3 | The move is illegal according to chess rules | The software displays an error message and prompts the player to re-select the move | Back to Step 1 |
| 2 | Step 4 | System error in updating the board | The software displays an error and rolls back to the previous valid state | Use case ends |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | Piece Selection | Identifier or location of the selected piece | Yes | Must be a piece belonging to the player | "Knight at B1" |
| 2 | Destination Square | The square to which the piece is moved | Yes | Must be a legal move for the selected piece | "C3" |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Updated Chess Board | Visual representation of the new board state | Graphical board update | Board shows knight at C3 |
| 2 | Move History | Log entry of the move made | Text log or table | "Knight from B1 to C3" |

1. **Postconditions**  
   • The game board reflects the move.  
   • The move is recorded in the game history.  
   • The turn is passed to the next player (or AI).

**Use Case “Undo a Move”**

1. **Use Case Code**  
   UC004
2. **Brief Description**  
   This use case describes the interaction when a player undoes the last move in the current game session.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session is active with at least one move made.  
   • The undo feature is enabled and available.
5. **Basic Flow of Events**
   1. The player selects the “Undo a Move” option.
   2. The software prompts for confirmation to prevent accidental undos.
   3. The player confirms the undo action.
   4. The software reverts the game state to the state before the last move.
   5. The move is removed from the move history.
   6. The game board is updated accordingly.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 2 | The player cancels the undo confirmation | The software cancels the undo action and retains the current game state | Use case ends |
| 2 | Step 4 | Undo process fails (e.g., system error) | The software displays an error message and does not alter the game state | Use case ends |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | Undo Command | Command or button press to undo a move | Yes | Valid click/key press event | Player clicks “Undo” |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Updated Chess Board | The board state after undoing the move | Graphical update | Board reverts to previous state |
| 2 | Updated Move History | The move history excluding the undone move | Text log or table | Last move removed from the log |
| 3 | Confirmation Message | Notification confirming successful undo action | Text message | "Move undone successfully" |

1. **Postconditions**  
   • The last move is removed from the game history.  
   • The chess board reflects the previous state.  
   • The game session continues with the reverted state.

**Use Case “Save Game”**

1. **Use Case Code**  
   UC005
2. **Brief Description**  
   This use case describes the interaction where the player saves the current game state for later resumption.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session is active.  
   • The player is at a point where saving is allowed (e.g., not during critical animations).
5. **Basic Flow of Events**
   1. The player selects the “Save Game” option from the game menu.
   2. The software prompts the player to enter a file name or automatically generates one.
   3. The player confirms the save action.
   4. The software saves the current game state to persistent storage.
   5. A confirmation message is displayed, indicating successful saving.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 2 | The player cancels the save prompt | The software aborts the save operation and retains the current session | Use case ends |
| 2 | Step 4 | Saving fails (e.g., disk error) | The software displays an error message and suggests retrying the save | Returns to Step 2 |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | Save Command | Command/button press to initiate saving | Yes | Valid user action | Player clicks “Save Game” |
| 2 | File Name (if applicable) | Name for the saved game file | Optional | Valid string without illegal characters | "game\_session\_01" |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Saved Game File | File containing the current game state | File saved on disk | "game\_session\_01.chr" |
| 2 | Confirmation Notification | Message confirming successful save action | Text message | "Game saved successfully" |

1. **Postconditions**  
   • The current game state is saved to persistent storage.  
   • The saved game file can be later retrieved to resume the session.

**Use Case “Load Game”**

1. **Use Case Code**  
   UC006
2. **Brief Description**  
   This use case describes the interaction where the player loads a previously saved game session.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A saved game file exists.  
   • The player is at the main menu or within an active session that allows loading.
5. **Basic Flow of Events**
   1. The player selects the “Load Game” option from the menu.
   2. The software displays a list of available saved game files.
   3. The player selects a file from the list.
   4. The software loads the game state from the selected file.
   5. The game board and session are updated to reflect the loaded state.
   6. A confirmation message is displayed indicating a successful load.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 3 | The player cancels file selection | The software aborts the load process and retains the current session | Use case ends |
| 2 | Step 4 | File load fails (e.g., corrupted file) | The software displays an error message and prompts for another selection | Returns to Step 2 |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | File Selection | Choice of a saved game file to load | Yes | Must be a valid, existing file | Player selects "game\_session\_01.chr" |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Loaded Game State | Updated game state loaded from the file | Graphical board update | Board displays state from "game\_session\_01" |
| 2 | Confirmation Notification | Message confirming the game has been loaded | Text message | "Game loaded successfully" |

1. **Postconditions**  
   • The game session resumes from the loaded state.  
   • The game board and history reflect the loaded game data.

**Use Case “Customize AI”**

1. **Use Case Code**  
   UC007
2. **Brief Description**  
   This use case describes the interaction where the player customizes parameters and settings of the AI opponent to adjust its behavior.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session is active or the AI customization feature is accessible from the main menu.  
   • The customization interface is available.
5. **Basic Flow of Events**
   1. The player selects the “Customize AI” option from the game settings menu.
   2. The software displays various AI settings (e.g., difficulty level, evaluation weights, randomness).
   3. The player adjusts one or more settings.
   4. The player confirms the customization choices.
   5. The software updates the AI parameters accordingly and displays a confirmation message.
   6. The updated settings are applied to the current or subsequent game sessions.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 4 | The player cancels the customization process | The software retains the previous AI settings and aborts changes | Use case ends |
| 2 | Step 5 | System error while updating AI settings | The software displays an error message and prompts the player to try again | Returns to Step 3 |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | AI Setting Changes | New values for AI parameters | Yes | Must be within acceptable ranges | "Difficulty: Medium", "Aggressiveness: High" |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Updated AI Parameters | Confirmation of updated AI settings | Text or summary display | "AI settings updated successfully" |
| 2 | Confirmation Notification | Message confirming that customization was successful | Text message | "Customization saved" |

1. **Postconditions**  
   • The AI opponent now operates with the newly customized settings.  
   • Future moves and behavior in the current session reflect these changes.

**Use Case “Exit Game”**

1. **Use Case Code**  
   UC008
2. **Brief Description**  
   This use case describes the interaction where the player exits the current game session.
3. **Actors**  
   3.1 Player
4. **Preconditions**  
   • A game session is active.  
   • The player is at a point where exiting is allowed (e.g., from the main game screen or pause menu).
5. **Basic Flow of Events**
   1. The player selects the “Exit Game” option from the game menu.
   2. The software prompts the player to confirm the exit action (especially if unsaved progress exists).
   3. The player confirms the exit action.
   4. The software terminates the current game session.
   5. The system returns the player to the main menu or closes the application.
6. **Alternative Flows**

| **No.** | **Location** | **Condition** | **Action** | **Resume Location** |
| --- | --- | --- | --- | --- |
| 1 | Step 2 | The player cancels the exit confirmation | The software aborts the exit action and returns to the game session | Use case ends |
| 2 | Step 4 | System error while terminating the session | The software displays an error message and attempts to safely close the session | Retry exit process |

1. **Input Data**

| **No.** | **Data Field** | **Description** | **Mandatory** | **Valid Condition** | **Example** |
| --- | --- | --- | --- | --- | --- |
| 1 | Exit Command | Command or button press to exit the game | Yes | Valid user action | Player clicks “Exit Game” |

1. **Output Data**

| **No.** | **Data Field** | **Description** | **Display Format** | **Example** |
| --- | --- | --- | --- | --- |
| 1 | Termination Notification | Message indicating that the game session has ended | Text message | "Game session ended. Returning to main menu." |

1. **Postconditions**  
   • The current game session is terminated.  
   • Unsaved progress is lost if not previously saved.  
   • The player is returned to the main menu or the application is closed.