**Chess Game Application – Report Documentation**

**1. Introduction**

This report documents the design and implementation of a Python-based chess game developed using the **Tkinter** GUI toolkit and the **python-chess** library. The program provides an interactive chessboard, enforces the official rules of chess, and incorporates advanced features such as move validation, highlighting legal moves, check/checkmate detection, timers, and game statistics tracking.

The purpose of this report is to explain the functionality of the implemented code, with emphasis on **check and checkmate handling**, user interface responsiveness, and system interaction.

**2. System Overview**

The chess application provides players with a visually interactive chessboard where pieces can be dragged and dropped according to the rules of chess.

Key features include:

* **Move Validation:** Ensures all moves follow legal chess rules.
* **Check Detection:** Notifies players when their king is under attack.
* **Checkmate Handling:** Declares the winner and ends the game.
* **Timers:** Each player has five minutes, decrementing only during their turn.
* **Game Statistics:** Records wins, losses, and draws.
* **Interactive Interface:** Drag-and-drop movements, hover highlights, and captured piece display.

**3. Imports and Dependencies**

The following libraries are used:

* **tkinter** – Provides the graphical user interface components.
* **chess (python-chess)** – Manages chess logic, rules, and board state.
* **time** – Used for player timers and countdown functionality.

**4. Application Structure**

The program is encapsulated within the **ChessApp** class, which inherits from **tk.Tk**. This class orchestrates GUI rendering, chess logic, timers, and event handling.

**4.1 Initialization (\_\_init\_\_ method)**

* Configures the main window and initializes board parameters (square size, colors, fonts).
* Creates UI elements:
  + **Chessboard canvas**
  + **Player timers**
  + **Captured pieces trays**
  + **Control buttons (New Game, Undo)**
  + **Game status display**
* Initializes game state variables: active turn, win/loss counters, draw count.
* Binds mouse events for piece interaction.
* Starts the countdown timers.

**5. Core Functionalities**

**5.1 Timer System**

* **start\_timer()** – Activates the countdown for the current player.
* **stop\_timer()** – Pauses the timer.
* **update\_timer()** – Updates the display in *minutes:seconds* format and detects timeout losses.

**5.2 Board Rendering**

* **draw\_board()** – Draws the 8×8 alternating colored grid.
* **update\_pieces()** – Places all chess pieces on their correct squares using Unicode symbols.
* **Highlighting Logic:**
  + Legal moves highlighted in yellow.
  + The king highlighted in **red** if under check.

**5.3 User Interaction**

* **on\_click()** – Selects a piece for movement.
* **on\_drag()** – Allows drag-and-drop motion of pieces.
* **on\_drop()** – Validates the move upon release, updating the board or resetting if illegal.
* **on\_hover()** – Displays possible legal moves.

**5.4 Game Logic**

* **Move Validation:** All moves are checked against the python-chess board state.
* **ask\_promotion()** – Invokes pawn promotion dialog when reaching the final rank.
* **add\_to\_tray()** – Stores captured pieces for display.
* **check\_game\_state()** – Determines if the game is in progress, check, or checkmate.
* **game\_over()** – Displays a popup declaring the result and updates win/draw statistics.
* **undo\_move()** – Reverts to the previous move.
* **reset\_game()** – Resets the board and starts a new game while preserving statistics.

**6. Check and Checkmate Implementation**

The newly introduced feature provides **visual and interactive responses** when a king is in check or checkmate:

1. **Check Detection:**
   * If the king is attacked, the program:
     + Highlights the king’s square in **red**.
     + Displays a popup: *“Check! Protect your king.”*
2. **Checkmate Detection:**
   * If no legal moves remain and the king is in check:
     + The program halts gameplay.
     + Declares victory for the opponent in a popup.
     + Highlights the defeated king in **red**.
     + Updates game statistics accordingly.

**7. Utility Functions**

* **get\_square()** – Converts pixel coordinates into chessboard positions (e.g., a2, e4).
* **reset\_piece\_position()** – Returns a piece to its original location after invalid movement.
* **update\_status()** – Refreshes labels and displays information such as current player, move number, and check/checkmate state.

**8. Error Handling**

* Illegal moves are **rejected** with user notifications.
* Invalid promotions default to a queen but prompt the user for confirmation.
* The system ensures consistency by rolling back invalid actions without crashing.

**9. Conclusion**

The Chess Game Application successfully integrates graphical rendering, official chess logic, and user-friendly interactions. With the addition of **check and checkmate popups** and **visual king highlighting**, the application provides players with real-time feedback on critical game states.

The system achieves its design goals by offering:

* A robust and rule-compliant chess experience.
* Clear and informative game state communication.
* Intuitive controls suitable for casual and competitive play.

📑