**📝 Tri’s Chess GUI Project Documentation**

**Tri’s Chess – GUI Chess with Stockfish Engine**

**Student Name:** Tri Toan Vo  
**Course:** Year 12 Software Engineering  
**Project Type:** Major Software Engineering Project

**Document Details**

* **Due Date:** 1 July 2025
* **Title:** Tri’s Chess – Interactive Chess GUI using Pygame & Stockfish
* **Purpose:** Develop an interactive chess game with a graphical user interface, integrating Stockfish as the AI engine, following best practices in software engineering.
* **Version:** 1.0

**Logbook**

| **Version** | **Date** | **Description** |
| --- | --- | --- |
| 0.1 | 25/05/2025 | Set up Pygame window & drew board squares |
| 0.2 | 26/05/2025 | Imported & scaled piece images |
| 0.3 | 27/05/2025 | Implemented click-based move input |
| 0.4 | 28/05/2025 | Integrated Stockfish for AI moves |
| 0.5 | 29/05/2025 | Added game over detection & error handling |
| 0.6 | 30/05/2025 | Finalized documentation & testing |

**Table of Contents**

1. Executive Summary
2. Introduction
3. Project Requirements
4. Planning & Design
5. Development Process
6. Testing & Evaluation
7. Risk Assessment
8. Conclusion
9. References

**1. Executive Summary**

This document presents the development of “Tri’s Chess,” a Python-based chess game with a graphical user interface built using Pygame and integrated with the pretrained AI. The project involved creating an interactive board, handling player input via mouse clicks, and automating AI responses.

**2. Introduction**

**Overview:**  
The purpose of this project is to apply software engineering principles by developing a playable chess GUI with integrated AI. The final product allows users to play against AI in a simple but functional interface.

**3. Project Requirements**

* **Functional Requirements:**
  + Display an 8×8 chessboard.
  + Allow user to click to select and move pieces.
  + Send player moves to Stockfish and receive responses.
  + Update the board state with player and AI moves.
* **Non-functional Requirements:**
  + Cross-platform compatibility.
  + Maintainable and readable code.
  + Responsive input without significant lag.

**4. Planning & Design**

* Board and pieces drawn with Pygame rectangles and blitted images.
* Mouse clicks converted to chess square coordinates.
* Integrated python-chess library for board state tracking.

**5. Development Process**

* **Setup:** Created a Pygame window and drew the chessboard.
* **Graphics:** Loaded piece images from assets; scaled images to fit each square.
* **Input Handling:** Captured mouse clicks, converted to algebraic moves.
* **Engine Integration:** Connected AI binary to process and respond to player moves.
* **Game Loop:** Updated board after each move and checked for game over conditions.

**6. Testing & Evaluation**

* Manually tested all legal moves for correct behavior.
* Verified Stockfish responds with legal, valid moves.
* Tested error handling for invalid user input.
* Checked game-over detection for checkmate and stalemate.

**7. Risk Assessment**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Incorrect move detection | Printed debug info, verified square mapping. |
| GUI glitches during rapid moves | Added slight delay after AI move. |
| Asset loading failures | Checked image paths on startup, added error handling. |

**8. Conclusion**

The project successfully met the objectives by developing a working chess GUI integrated with pretrained AI, allowing the user to play interactively with correct rules and move validation. The final product demonstrates effective planning, code design, and testing.

**9. References**

* python-chess library: <https://python-chess.readthedocs.io>
* Pygame library: <https://www.pygame.org>
* Lucas Chess (assets): https://lucaschess.pythonanywhere.com