Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Fall 2019

CHECK-MATES

CheckMates
READY GO

Jordan Burnes
Rahme Butaineh
Sadat Hossain
Michael Santellan
Dongchen Ye

REVISION HISTORY

Revision	Date	Author(s)	Description
0.1	11.05.2019	JB	document creation
0.2	11.11.2019	JB, RB, SH, MS, DY	complete draft
0.2	05.11.2020	JB, RB, SH, MS, DY	Rework

CONTENTS

List of Figures

List of Tables

1 Introduction

"READY GO!" is a system that will allow for a person to play chess against an AI. The person will be able to select from various difficulties using a switch that has 3 modes for easy, medium, and hard. The system will take input from the person using a camera and machine vision to discern the different pieces and their positions. The camera will be mounted on a pole and will only record position once a button is pressed by the user to signify that the person is done moving. The GUI will then make the appropriate move and/or emit words from a speaker in case of a check, checkmate, error or loss.

2 System Overview

Our system will consist of five layers that work together to successfully play a game against a human. Our input layer will take input from the user. Our Machine Learning layer will help us understand where pieces are being moved. Our Logic layer will help us organize data into something useful. Our Movement layer will handle the calculations needed for moving the pieces. Finally, our output layer will emit messages in a format that is understandable to people playing against the system.

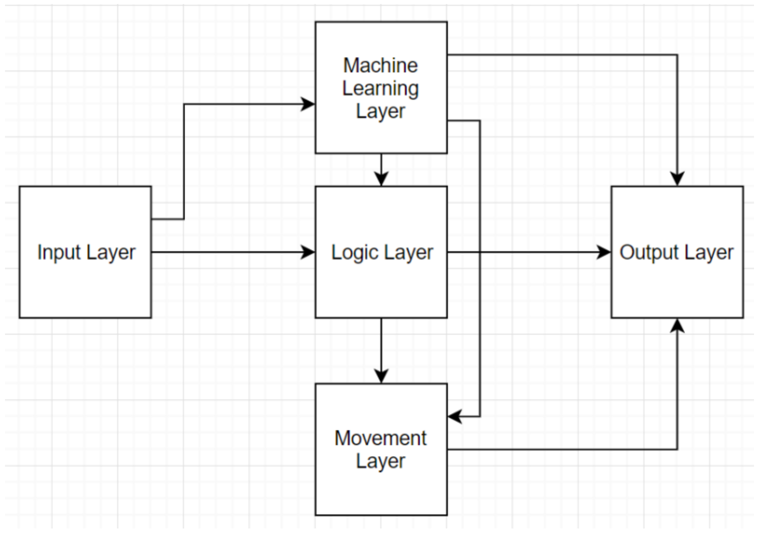

Figure 1: A simple architectural layer diagram

2.1 Layer Input Description

The input layer will be the main layer that handles the interaction between the person and the system. It will consists of input devices located on the GUI such as the switch which determines difficulty, the button which will signify when the person is done making their move, and the camera which will be used to determine the position of the pieces on the board. This layer will have input that it sends to two different layers; the Machine Learning Layer and the Logic Layer. It will also be effected by the output layer when things like errors are triggered, prompting the user to make an appropriate move.

2.2 Layer Machine Learning Description

This layer is what will be used when determining the position of the board after the user has moved. It will speak mainly to the Logic layer, sending data about where each piece is located. This layer will also talk to the Output layer if one of the pieces end up on the edge of the board, making it hard for the machine learning algorithm to determine where exactly the piece was placed. If this happens it will talk directly to the output layer to handle the error.

2.3 Layer Logic Description

The logic layer will be the layer that handles most of the deciding factors in the system such as where the next piece should be placed. It takes most of its inputs from the Machine Learning layer in order to get an idea of the board state. Once that information is received it can translate that information into a type that we can use to make decisions on the best place for the next move based on the difficulty setting. Once a decision has been made on where to move it will send data to the movement layer. If a piece had been placed in an illegal spot, however, then the data would be sent to the output layer to emit an error message from its speaker.

2.4 Layer Output Description

The output layer is going to be the layer that relays all of the information to the user in order to play the game. It will consist of a speaker that will be used for messages like check, check-mate, as well as other message for winning, losing, and error handling. This layer also includes the GUI. It will receive input from the Stockfish API and

3 Subsystem Definitions & Data Flow

This section gives more insight on the data flow on the system by going more in depth from figure 1. Figure 2 provides more details regarding how data is transferred withing the system. Figure 2 also displays the underlying components of each system and shows how they connect to other layers within the system. These following subsystems will be discussed in further sections of this document.

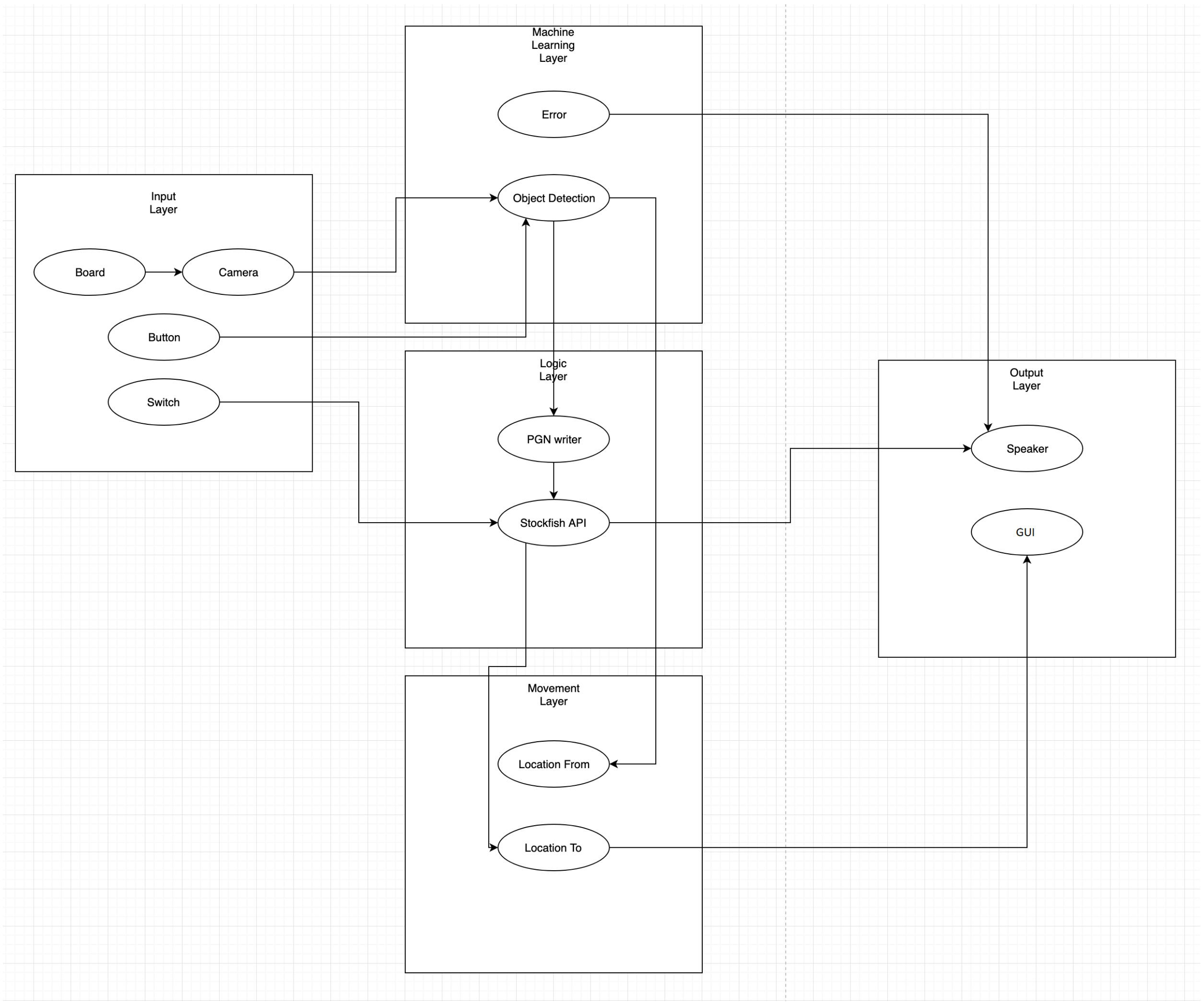

Figure 2: A simple data flow diagram

4 Input Layer Subsystems

This layer pertains to all of the inputs made into the system. There is several types of inputs that this layer deals with ranging from board movements, button presses, camera, and switch presses. All of these inputs will be recognized and sent to other layers within the system to be handled with appropriately to get the expected outputs. This layer basically handles all of the users interactions with the system.

4.1 Board

The board is a standard 8x8, white and black chess board with white and black pieces. We will be using a roll-out chess mat, as most of our clients (high-schoolers) will be acquainted with them.

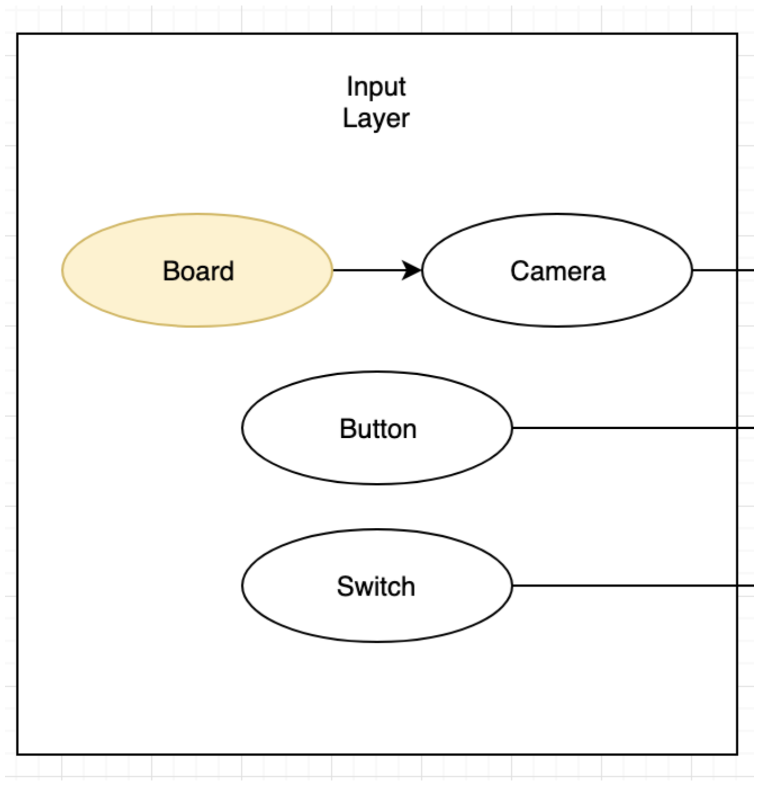

Figure 3: board subsystem

4.1.1 Assumptions

We will assume the following:

- the board is standard 8x8 squares, black and white
- the pieces are standard shapes, black and white, and the same style as the pieces the model is trained on
- the board will be orientated correctly
- the pieces are in the correct positions at the start of the game
- there are no damaging markings on the board that would cause a detection error

4.1.2 Responsibilities

The board is the interface in which the game is played on, and through which our model runs. It is the job of the player to play valid moves on the board and position pieces squarely in appropriate squares. The board will hold the state of the game.

4.1.3 Subsystem Interfaces

Table 2: Board interfaces

ID	Description	Inputs	Outputs
#01	Board State	player move	board state

4.2 Camera

The camera will be a small, portable HD camera, that attaches to a pole for a birds-eye view of the board.

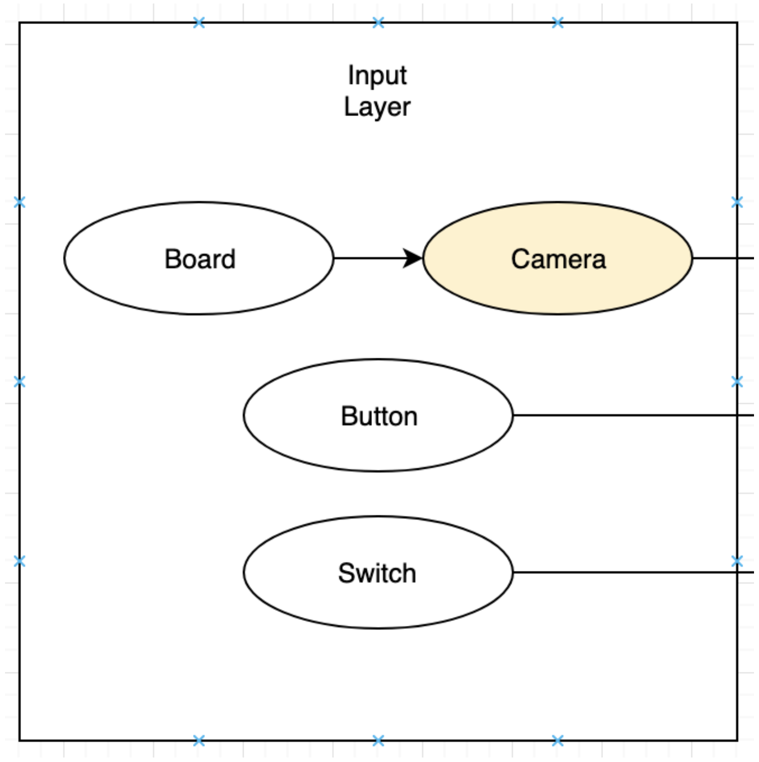

Figure 4: camera subsystem

4.2.1 Assumptions

We will assume the following:

- the camera will take HD quality images
- the camera will be attached to a pole for a birds-eye view of the board
- the pole will be attached to the board in order to standardize the view

4.2.2 Responsibilities

The camera is responsible for capturing images of the board state, and saving them to a .pgn image file to be read by the model.

4.2.3 Camera interface

Table 3: Camera interfaces

ID	Description	Inputs	Outputs
#02	Image	board state	.pgn image

4.3 Button

The button will be on the GUI and will tell the system that it is it's turn to play.

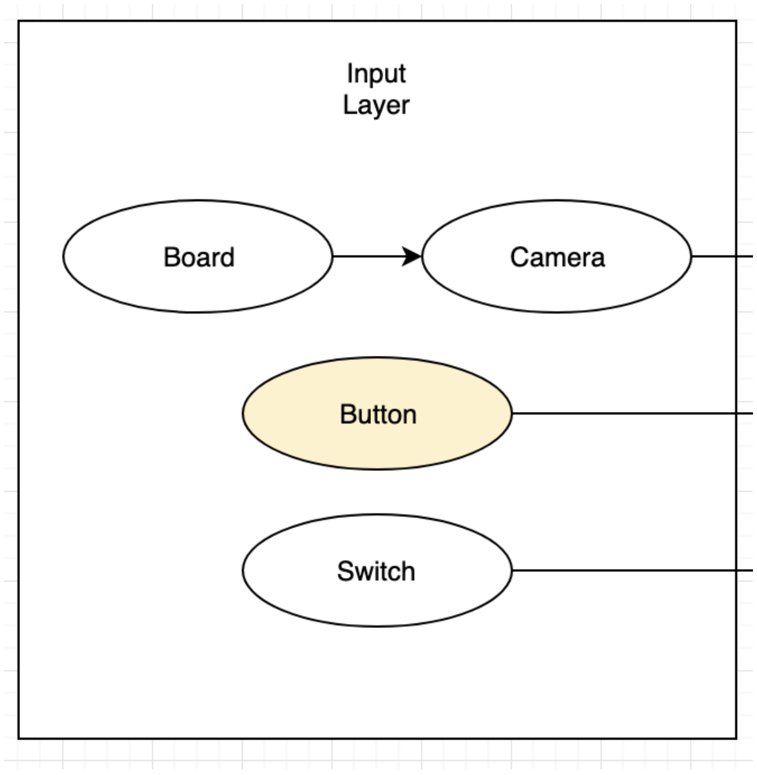

Figure 5: button subsystem

4.3.1 Assumptions

We will assume the following:

- the user knows what a button is and how to press one

4.3.2 Responsibilities

As simple as it is, the button plays a critical role. When pressed, it tells the machine to start the object detection, analyze the position, and triggers a movement on the GUI.

4.3.3 Button interface

Table 4: Button interfaces

ID	Description	Inputs	Outputs
#03	Button	user press	start object detection

4.4 Difficulty Switch

The difficulty switch will be a standard, metal switch, with three notches to which the user can move it. It is responsible for setting the difficulty level at which the AI will play chess against the opponent, with the three difficulties being easy, medium, hard.

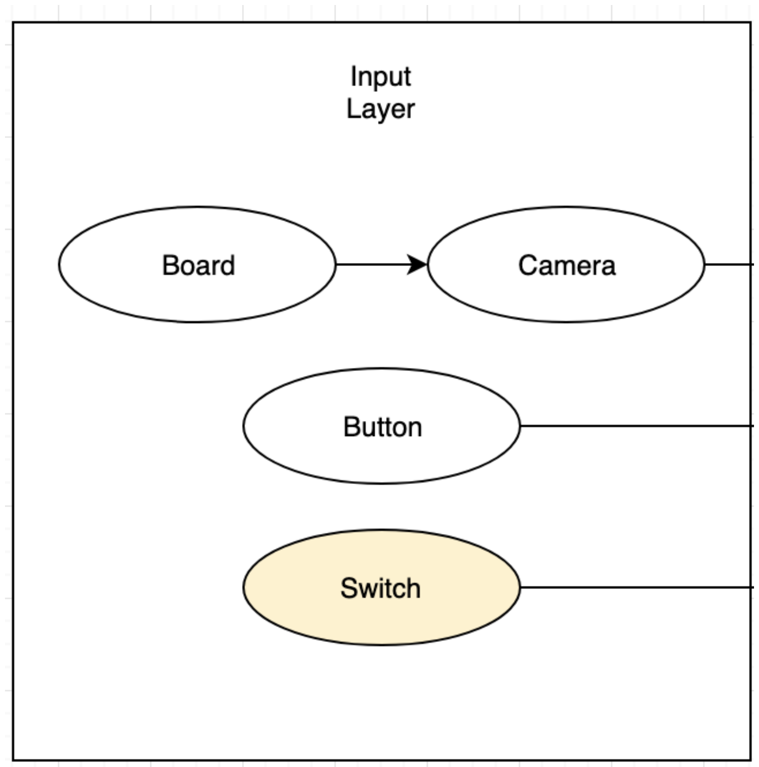

Figure 6: switch subsystem

4.4.1 Assumptions

We will assume the following:

- the user knows what a switch is and how to operate it

4.4.2 Responsibility

The switch is responsible for setting the difficulty level for the AI. When the switch changes states, the Stockfish chess engine will determine the next best move at the appropriate depth. The easy the difficulty, the less depth the engine will search for the best move.

4.4.3 SWITCH INTERFACE

Table 5: Switch interfaces

ID	Description	Inputs	Outputs
#04	Switch	user toggle	change difficulty level

5 Machine Learning Layer Subsystems

This layer is what will be used when determining the position of the board after the user has moved. It will speak mainly to the Logic layer, sending data about where each piece is located. This layer will also talk to the Output layer if one of the pieces end up on the edge of the board, making it hard for the machine learning algorithm to determine where exactly the piece was placed. If this happens it will talk directly to the output layer to handle the error.

5.1 Object Detection

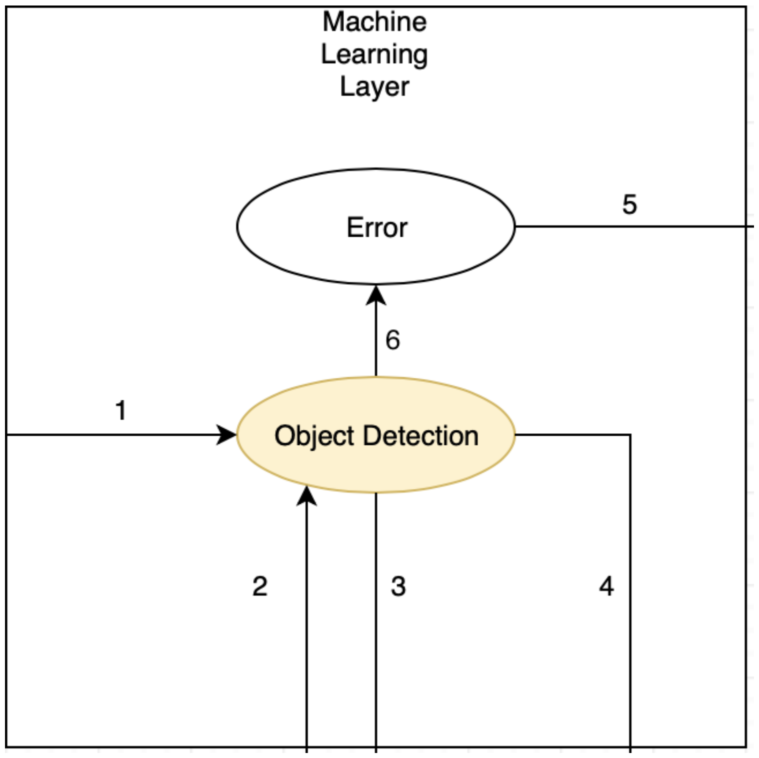

Figure 7: object detection subsystems

5.1.1 Assumptions

Assume the button in the "Input" Layer works properly and the camera in the "Input" Layer works properly with low degree of noises.

5.1.2 Responsibilities

This subsystem will be responsible for detecting what movement has been made by the opponent. It will be triggered by pressing the button in the "Input" Layer. Then, it takes input from the camera in the "Input" and runs algorithms to detect the current board states. If a legal move is detected, it will send the feedback to the "Logic" Layer and "Movement" Layer. If a illegal move is detected, it will send the error message to the Error subsystem.

5.1.3 Object Detection Interfaces

Table 6: Object Detection interfaces

ID	Description	Inputs	Outputs
#1&2	Object Detection	Button pressed Images	-
#3&4	Detect legal move	-	Piece's movement Piece's location
#6	Detect illegal move	-	Board State Information

5.2 Error Detection

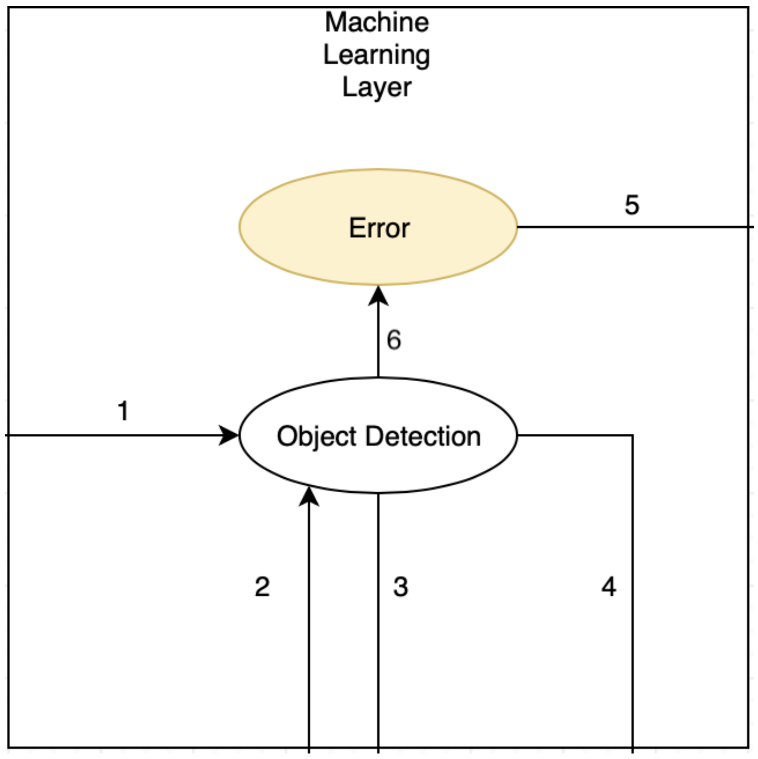

Figure 8: error detection subsystems

5.2.1 Assumptions

Assume the Object detection subsystem works properly and sends correct board states information.

5.2.2 Responsibilities

This subsystem will be responsible for identify the illegal move made by the opponent. It will be triggered by the Object Detection Subsystem and take the board states as input. Then it evaluates the differences between the current state and last state to identifies the error and send the corresponding error messages to the "Output" layer.

5.2.3 Error Detection Interfaces

Table 7: Error Detection interfaces

ID	Description	Inputs	Outputs
#5&6	Error Identification	Board States	Error Message

6 Logic Layer Subsystems

The logic layer will be the layer that handles most of the deciding factors in the system such as where the next piece should be placed. It takes most of its inputs from the Machine Learning layer in order to get an idea of the board state. Once that information is received it can translate that information into a type that we can use to make decisions on the best place for the next move based on the difficulty setting. Once a decision has been made on where to move it will send data to the movement layer. If a piece had been placed in an illegal spot, however, then the data would be sent to the output layer to emit an error message from its speaker.

6.1 PGN Writer

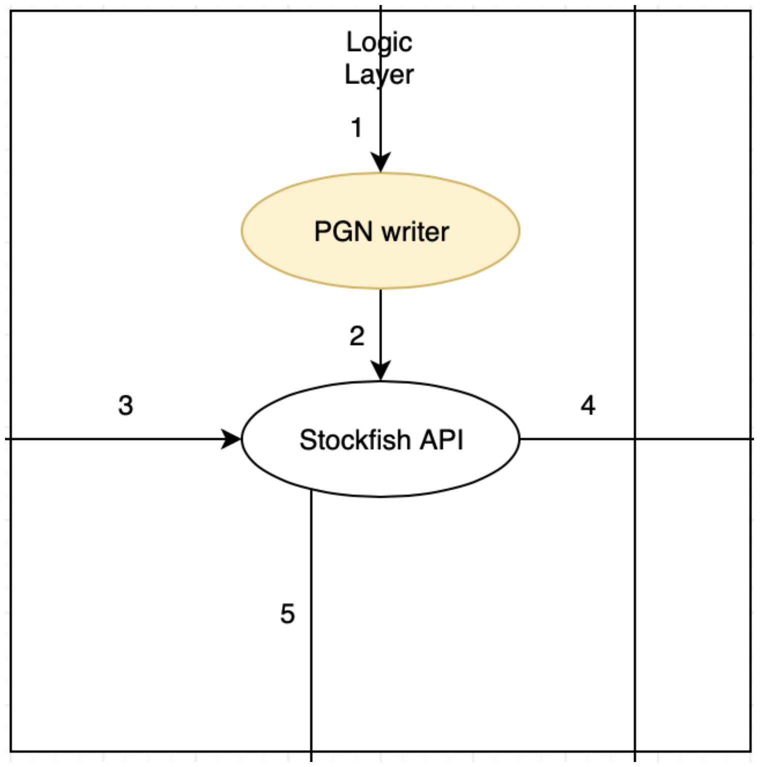

Figure 9: PGN writer subsystems

6.1.1 Assumptions

Assume the all the chess variants can be recorded using PGN.

6.1.2 Responsibilities

This subsystem will be responsible for structuring the game data into PGN format for the purpose of easy parsing and generation by computer programs. The output PGN code consist of a set of "tag pairs" (game information), followed by the "movetext" (chess moves).

6.1.3 PGN Writer Interfaces

Table 8: PGN Writer interfaces

ID	Description	Inputs	Outputs
#1&2	PGN Writer	Pieces movement	PGN code

6.2 Stockfish API

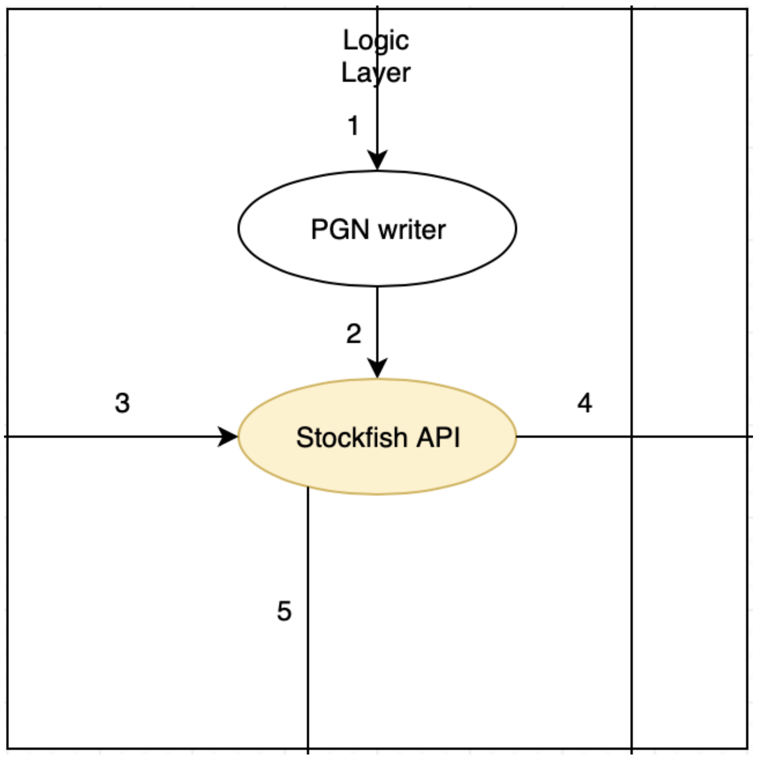

Figure 10: stockfish API subsystems

6.2.1 Assumptions

Assume the PGN code is in the correct format.

6.2.2 Responsibilities

This subsystem will be responsible for determining the best place for the next move based on the difficulty setting. It will use the Stockfish API which is a free and open-source UCI chess engine. It could trigger the speaker in the "Output" layer when an illegal movement is detected. The decisions of the next movement made by the chess engine will be passed down to "Movement" layer. A movement could also trigger the speaker if a checkmate happens or the game is finished.

6.2.3 Stockfish API Interfaces

Table 9: Stockfish API interfaces

ID	Description	Inputs	Outputs
#2&3	Stockfish API Input	PGN code Difficult setting	-
#4	Game status (when finished)	-	Game status message
#4	Illegal movement detected	-	Error message
#4&5	Make a decision	-	Pieces movement

7 Movement Layer Subsystems

The movement layer decides the movement of the pieces to be displayed on the GUI. Based on image recognition that is done in the machine learning layer, it will take measurements of the distance the GUI has to move certain pieces.

7.1 Location From

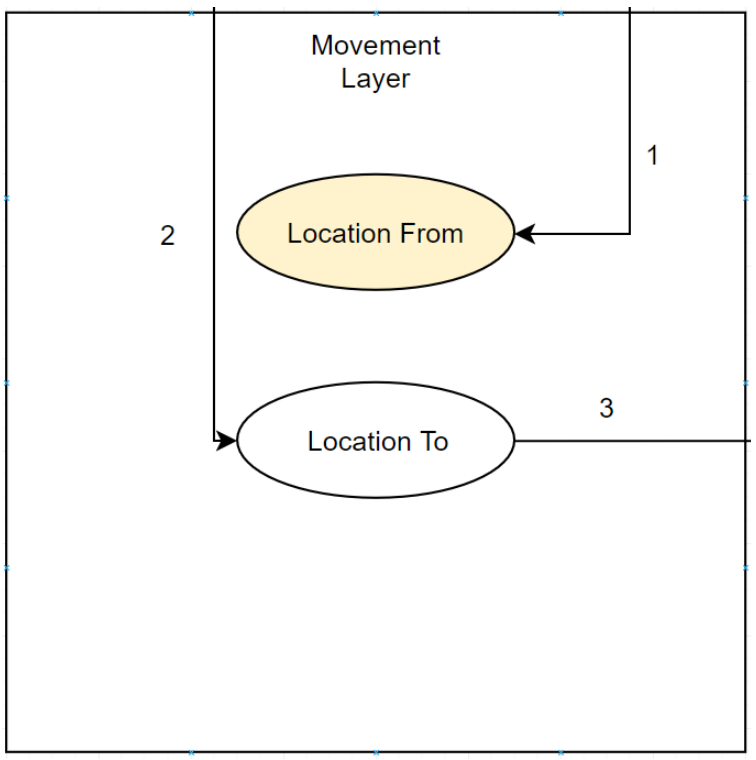

Figure 11: Location From subsystems

7.1.1 Assumptions

Assume that the last picture received from image recognition is correct and that the state of the chess board has not been changed.

7.1.2 Responsibilities

This subsystem will be responsible for calculating the distance that the pieces have to move.

7.1.3 Location From Interfaces

Table 10: Location From interfaces

ID	Description	Inputs	Outputs
1	Input from object detection	image of chess board state	Calculation for GUI

7.2 Location To

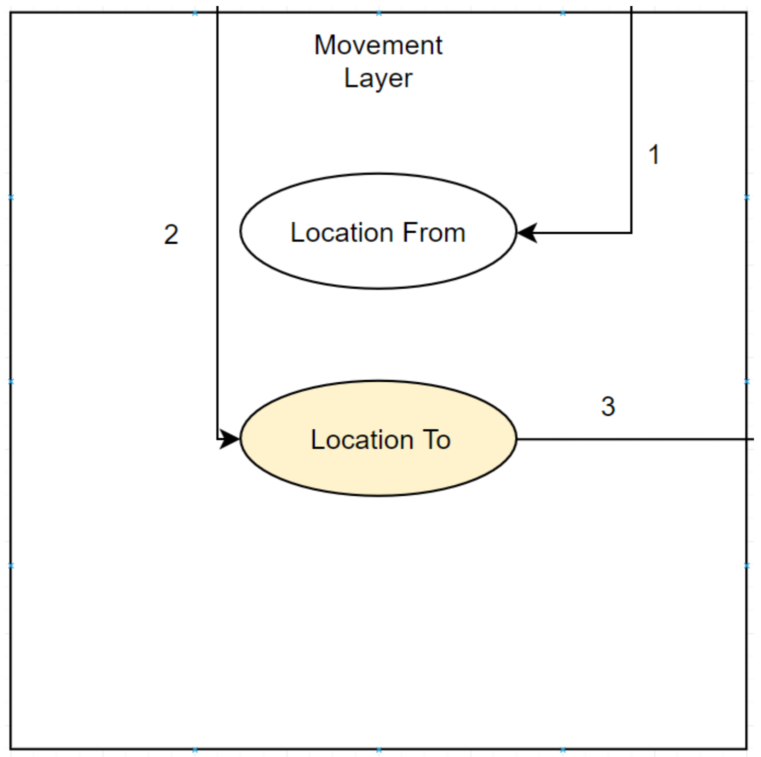

Figure 12: Location To subsystems

7.2.1 Assumptions

Stockfish API will send correct values for movement.

7.2.2 Responsibilities

Depending on the value of where to move the piece the Calculation will be done and sent to the GUI.

7.2.3 Location To Interfaces

Table 11: Location To interfaces

ID	Description	Inputs	Outputs
#2	Input from stockfish API	Values for movement	-
#3	Output to GUI	-	Movement specifics sent to GUI

8 Output Layer Subsystems

The output layer subsystem handles the actual outputs of the GUI and audio from the speakers. The GUI will output after given the destination from the movement layer subsystem. The speaker will output audio after encountering an error from the machine learning layer subsystem or from the Stockfish API in the logic layer subsystem.

8.1 Speaker

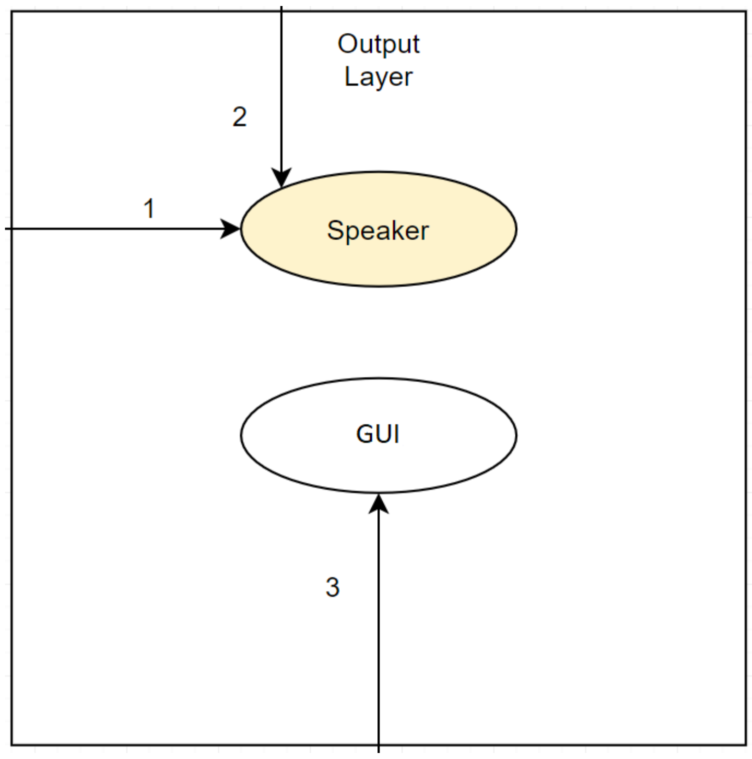

Figure 13: Speaker subsystems

8.1.1 Assumptions

The speaker is fully functional and doesn't have a faulty component. The Logic layer and Machine Learning layers are functioning properly without errors. Also they are able to pass data from the Stockfish API and error components correctly.

8.1.2 Responsibilities

The responsibilities of the speaker component of the output layer subsystem is to output audio to the users. The speaker can output error messages if illegal or incorrect moves are made. The speaker can also output messages if the game has ended by either a checkmate or time based victory.

8.1.3 Speaker Interfaces

Table 12: Speaker interfaces

ID	Description	Inputs	Outputs
#1&2	Error or Victory Messages	Audio Inputs	Audio Outputs

8.2 GUI

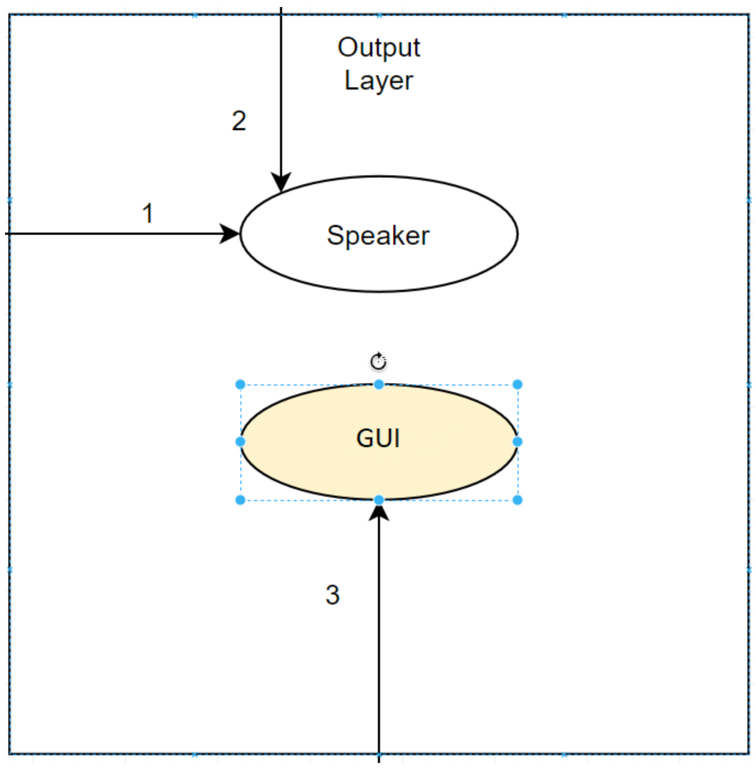

Figure 14: GUI subsystems

8.2.1 Assumptions

The GUI is full functional and doesn't have a faulty component. The Movement layer needs to also be functioning properly to be able to pass location data.

8.2.2 Responsibilities

The responsibilities of the GUI component of the output layer subsystem is to simulate the movement of chess pieces to the correct location with the display.

8.2.3 GUI INTERFACES

Table 13: GUI interfaces

ID	Description	Inputs	Outputs
#3	GUI Movements	Distance Calculations	Chess pieces moved

References