

Module Interface Specification for Chess Connect

Chess Connect

January 18, 2023

1 Revision History

Table of Revisions

Table 1: Revision History

Date	Developer(s)	Change
2023-01-16	Jonathan Cels, Rupinder Nagra	Web Application Modules
2023-01-18	Jonathan Cels, Rupinder Nagra	Finalized Web Application Modules

2 Symbols, Abbreviations and Acronyms

symbol	description
M	Module
MIS	Module Interface Specification
R	Requirement
FEN	Forsyth-Edwards Notation

Contents

1	Revision History	i
	Table of Revisions	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Web Application Input Module	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Constants	3
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	3
6.4.5	Local Functions	4
7	MIS of Display Module	5
7.1	Module	5
7.2	Uses	5
7.3	Syntax	5
7.3.1	Exported Constants	5
7.3.2	Exported Access Programs	5
7.4	Semantics	5
7.4.1	State Variables	5
7.4.2	Environment Variables	5
7.4.3	Assumptions	5
7.4.4	Access Routine Semantics	5
7.4.5	Local Functions	6
8	MIS of Web Application Output Module	7
8.1	Module	7
8.2	Uses	7
8.3	Syntax	7

8.3.1	Exported Constants	7
8.3.2	Exported Access Programs	7
8.4	Semantics	7
8.4.1	State Variables	7
8.4.2	Environment Variables	7
8.4.3	Assumptions	7
8.4.4	Access Routine Semantics	7
8.4.5	Local Functions	7
9	MIS of User Mode Module	8
9.1	Module	8
9.2	Uses	8
9.3	Syntax	8
9.3.1	Exported Constants	8
9.3.2	Exported Access Programs	8
9.4	Semantics	8
9.4.1	State Variables	8
9.4.2	Environment Variables	8
9.4.3	Assumptions	8
9.4.4	Access Routine Semantics	8
9.4.5	Local Functions	9
10	MIS of Board Module	10
10.1	Module	10
10.2	Uses	10
10.3	Syntax	10
10.3.1	Exported Constants	10
10.3.2	Exported Access Programs	10
10.4	Semantics	10
10.4.1	State Variables	10
10.4.2	Environment Variables	10
10.4.3	Assumptions	10
10.4.4	Access Routine Semantics	11
10.4.5	Local Functions	11
11	MIS of Web Application Game State Module	12
11.1	Module	12
11.2	Uses	12
11.3	Syntax	12
11.3.1	Exported Constants	12
11.3.2	Exported Access Programs	12
11.4	Semantics	12
11.4.1	State Variables	12

11.4.2	Environment Variables	12
11.4.3	Assumptions	12
11.4.4	Access Routine Semantics	12
11.4.5	Local Functions	13
12	MIS of Engine Module	14
12.1	Module	14
12.2	Uses	14
12.3	Syntax	14
12.3.1	Exported Constants	14
12.3.2	Exported Access Programs	14
12.4	Semantics	14
12.4.1	State Variables	14
12.4.2	Environment Variables	14
12.4.3	Assumptions	14
12.4.4	Access Routine Semantics	14
12.4.5	Local Functions	15
13	Appendix	17

3 Introduction

The following document details the Module Interface Specifications for Chess Connect. The Chess Connect project aims to bridge the gap between physical and online chess play by enabling two players to play a game on a physical board while simultaneously transmitting the moves to a web application via Bluetooth. This central platform will eliminate the need for players to switch between different mediums and will provide a more flexible and accessible way for new players to learn the game.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/ChessConnect/chess-connect>.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Chess Connect.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of Chess Connect uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Chess Connect uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding	
	Input Parameters
	Output Format
	Output Verification
Behaviour-Hiding	Temperature ODEs
	Energy Equations
	Control Module
	Specification Parameters Module
Software Decision	Sequence Data Structure
	ODE Solver
	Plotting

Table 2: Module Hierarchy

6 MIS of Web Application Input Module

6.1 Module

Web Application Input

6.2 Uses

[Board Module](#)

[User Mode Module](#)

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
parseInput	string	seq of string	invalidInput

6.4 Semantics

6.4.1 State Variables

inputString: string #String containing [FEN](#) string, user mode, game termination state, and delimiting characters

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

parseInput():

- output: sequence of strings. The first is the FEN string, the second is the user mode, the third is the game termination state.
- exception: invalidInput if any of validFen, validUserMode, or validGameTermination return false.

6.4.5 Local Functions

Name	In	Out	Exceptions
validFen	string	boolean	
validUserMode	string	boolean	
validGameTermination	string	boolean	

7 MIS of Display Module

7.1 Module

Display

7.2 Uses

[Board Module](#)

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSquare	string		
drawBoard	seq of (seq of int)		
displayGameTermination	int		
setBackground	string		

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

drawSquare():

- output: Draw board square
- exception: none

drawBoard():

- transition: Uses drawSquare to display the game board
- exception: none

displayGameTermination():

- transition: Displays game termination state (checkmate, stalemate, etc.)
- exception: none

setBackground():

- transition: Sets the background colors of the display.
- exception: none

7.4.5 Local Functions

N/A

8 MIS of Web Application Output Module

8.1 Module

Web Application Output

8.2 Uses

[Engine Module](#)

[Game State Module](#)

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
sendData	string	string	

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

sendData(string):

- output: string #Encodes game state (none, check, checkmate, stalemate), and 3 engine-generated moves
- exception: none

8.4.5 Local Functions

N/A

9 MIS of User Mode Module

9.1 Module

User Mode

9.2 Uses

[Engine Module](#)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
getUserMode		string	
setUserMode	string		

9.4 Semantics

9.4.1 State Variables

userMode: string #Represents the current user mode (Normal, Beginner, Engine)

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

getMode():

- output: string

output := userMode

- exception: none

setMode(string):

- transition: Sets `userMode` to the input user mode

userMode := input

- exception: none

9.4.5 Local Functions

N/A

10 MIS of Board Module

10.1 Module

Board

10.2 Uses

[Engine Module](#)

[Game State Module](#)

10.3 Syntax

10.3.1 Exported Constants

```
#define letters ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']  
#define startFEN = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -  
0 1'  
#define boardDimension = 8
```

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize			
getXYPosition	int	tuple of int	invalidIndex
getPosition	int	tuple of int	
getFenString		string	
setFenString	string		

10.4 Semantics

10.4.1 State Variables

fenString: string #Stores FEN string of current game position

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

initialize is called before any other access routine.

10.4.4 Access Routine Semantics

initialize():

- transition: #Initializes fenString to the starting chess board position

$$fenString := startFEN$$

- exception: none

getXYPosition(int: squareInd):

- output: #X and Y number coordinate for an input square number. Eg. getXYPosition(14) returns (0, 6).

$$out := (squareInd // boardDimension, squareInd \% boardDimension)$$

- exception: none

getPosition(int: squareInd):

- output: #letter and number coordinate for an input square number. Eg. getPosition(14) returns 'g7'.

$$out := 'letters[squareInd \% boardDimension]' \\ + 'boardDimension - (squareInd // boardDimension)'$$

- exception: none

getFenString():

- output:

$$out := fenString$$

- exception: none

setFenString(string: fen):

- transition:

$$fenString := fen$$

- exception: none

10.4.5 Local Functions

N/A

11 MIS of Web Application Game State Module

11.1 Module

Web Application Game State

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
isCheck	string	boolean	
isCheckmate	string	boolean	
isStalemate	string	boolean	

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

isCheck():

- output: True if the position is ‘check’, false otherwise
- exception: none

isCheckmate():

- output: True if the position is ‘checkmate’, false otherwise

- exception: none

isStalemate():

- output: True if the position is ‘stalemate’, false otherwise
- exception: none

11.4.5 Local Functions

N/A

12 MIS of Engine Module

12.1 Module

Engine

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Constants

`#define depth` #How many layers of depth the chess engine should use to evaluate the position
`#define maxSearchTime` #The maximum time the chess engine should take to evaluate the position

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
evaluatePosition	string	string	

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

The depth and maxSearchTime values will be determined experimentally after the system is built. There is a trade-off between move quality and speed/depth of the search.

12.4.4 Access Routine Semantics

evaluatePosition(string):

- output: String containing 3 possible moves, calculated by a chess engine from the FEN input string
- exception: none

12.4.5 Local Functions

N/A

References

- FEN. Fen (forsyth-edwards notation) - chess terms. <https://www.chess.com/terms/fen-chess>. Accessed: 2023-01-18.
- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.

13 Appendix

[Extra information if required —SS]