

System Verification and Validation Report for Chess Connect

Team #4,
Alexander Van Kralingen
Arshdeep Aujla
Jonathan Cels
Joshua Chapman
Rupinder Nagra

March 7, 2023

1 Revision History

Date	Version	Notes
2023-03-04	Arshdeep Aujla	Added Template for Nonfunctional Requirements
2023-03-05	Arshdeep Aujla	Added Table for functional requirements, traceability matrix
2023-03-07	Jonathan Cels	Added some nonfunctional requirement test reports

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

Refer to SRS Section 1 for an extensive list of used symbols, abbreviations, and acronyms.

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Functional Requirements Evaluation	1
3.1	Game Active State	1
3.2	Game Inactive State	2
3.3	Normal Mode	2
3.4	Engine Mode	2
3.5	Beginner Mode	3
4	Nonfunctional Requirements Evaluation	3
4.1	Look and Feel	3
4.2	Usability and Humanity	3
4.3	Performance	4
4.4	Health and Safety	4
4.5	Precision and Accuracy	4
4.6	Capacity	4
4.7	Security	4
5	Unit Testing	5
6	Changes Due to Testing	5
7	Automated Testing	5
8	Trace to Requirements	5
9	Trace to Modules	6
10	Code Coverage Metrics	6
A	Reflection Appendix	6

List of Tables

1	Active State Functional Requirements Results	1
2	Inactive State Functional Requirements Results	2
3	Normal Mode Functional Requirements Results	2
4	Engine Mode Functional Requirements Results	2
5	Beginner Mode Functional Requirements Results	3

6	Look and Feel Non-Functional Requirements Results	3
7	Usability and Humanity Non-Functional Requirements Results	3
8	Performance Non-Functional Requirements Results	4
9	Health and Safety Non-Functional Requirements Results	4
10	Precision and Accuracy Non-Functional Requirements Results	4
11	Capacity Non-Functional Requirements Results	4
12	Security Non-Functional Requirements Results	4
13	Requirements Traceability Matrix	6

List of Figures

3 Functional Requirements Evaluation

Refer to the VnV Plan for descriptions of the tests derived to evaluate the functional requirements.

3.1 Game Active State

Test	Input	Expected	Actual	Notes	Result
GA-1	Draw/resign button pressed while game active.	System variable 'gameInProgress' set to false.	System variable configured correctly.		Pass
GA-2	Start game button pressed while game active.	System variable 'gameInProgress' remains true.	System variable configured correctly.		Pass
GA-3	User mode button pressed while game active.	System variable 'currMode' changed to represent the selected user mode.	User mode unchanged.	Design changed, user mode not switchable while a game is active. Test fails by design.	Fail
GA-4	Start game button pressed while game inactive.	System variable 'gameInProgress' set to true, 'currFEN' variable is set to the starting FEN.	System variables configured correctly.		Pass
GA-5	Move made that results in stalemate or checkmate according to the rules of chess.	System variable 'gameInProgress' set to false.	System variable configured correctly.		Pass

Table 1: Active State Functional Requirements Results

3.2 Game Inactive State

Test	Input	Expected	Actual	Notes	Result
GI-1					
GI-2					
GI-3					
GI-4					
GI-5					

Table 2: Inactive State Functional Requirements Results

3.3 Normal Mode

Test	Input	Expected	Actual	Notes	Result
GA-1					
GA-2					
GA-3					
GA-4					
GA-5					

Table 3: Normal Mode Functional Requirements Results

3.4 Engine Mode

Test	Input	Expected	Actual	Notes	Result
GA-1					
GA-2					
GA-3					
GA-4					
GA-5					

Table 4: Engine Mode Functional Requirements Results

3.5 Beginner Mode

Test	Input	Expected	Actual	Notes	Result
GA-1					
GA-2					
GA-3					
GA-4					
GA-5					

Table 5: Beginner Mode Functional Requirements Results

4 Nonfunctional Requirements Evaluation

Refer to the VnV Plan for descriptions of the tests derived to evaluate the non-functional requirements.

4.1 Look and Feel

Test	Inputs	Expected Result	Actual Result
NFT-1			

Table 6: Look and Feel Non-Functional Requirements Results

4.2 Usability and Humanity

Test	Inputs	Expected Result	Actual Result
NFT-2			
NFT-3			

Table 7: Usability and Humanity Non-Functional Requirements Results

4.3 Performance

Test	Inputs	Expected Result	Actual Result
NFT-4			
NFT-5			
NFT-6			
NFT-7			

Table 8: Performance Non-Functional Requirements Results

4.4 Health and Safety

Test	Inputs	Expected Result	Actual Result
NFT-8	Test		

Table 9: Health and Safety Non-Functional Requirements Results

4.5 Precision and Accuracy

Test	Inputs	Expected Result	Actual Result
NFT-9			

Table 10: Precision and Accuracy Non-Functional Requirements Results

4.6 Capacity

Test	Inputs	Expected Result	Actual Result
NFT-10			

Table 11: Capacity Non-Functional Requirements Results

4.7 Security

Test	Inputs	Expected Result	Actual Result
NFT-11			
NFT-12			

Table 12: Security Non-Functional Requirements Results

5 Unit Testing

6 Changes Due to Testing

7 Automated Testing

8 Trace to Requirements

Test	Requirement
GA-1	GA1
GA-2	GA2
GA-3	GA3
GA-4	GA6
GA-5	GA7
GI-1	GI1
GI-2	GI2
GI-3	GI3
GI-4	GI4
GI-5	GI5, GI6
NB-1	NB1
NB-2	NB2
NB-3	NB3
ND-1	ND1
NA-1	NA1, NA2
NA-2	NA3
EB-1	EB1
EB-2	EB2
EB-3	EB3
EB-4	EB4
ED-1	ED1
ED-2	ED2
EA-1	EA1, EA2
EA-2	EA3, EA4, EA5
EA-3	EA6

BB-1	BB1
BB-2	BB2
BB-3	BB3
BB-4	BB4
BB-5	BB5
BD-1	BD1
BA-1	BA1
BA-2	BA2
NFT1	LF3
NFT2	UH5
NFT3	UH6
NFT4	PR1
NFT5	PR2
NFT6	PR3
NFT7	PR4
NFT8	PR6
NFT9	PR7
NFT10	PR10
NFT11	SR4
NFT12	SR3

Table 13: Requirements Traceability Matrix

9 Trace to Modules

10 Code Coverage Metrics

A Reflection Appendix

References

Author Author. System requirements specification. <https://github.com/...>, 2019.

CSA. *Canadian Electrical Code*. CSA Group, 2021.

FEN. Fen (forsyth-edwards notation) - chess terms. <https://www.chess.com/terms/fen-chess>. Accessed: 2023-01-18.

- Interal Chess Federation FIDE. Fide laws of chess. <https://www.fide.com/FIDE/handbook/LawsOfChess.pdf>, 2018.
- Canadian Centre for Occupational Health and Safety. Hazard and risk: Osh answers. https://www.ccohs.ca/oshanswers/hsprograms/hazard_risk.html, 2022. Accessed: 2022-10-05.
- 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>.
- List of Chess Variants. List of chess variants, Sep 2022. URL https://en.wikipedia.org/wiki/List_of_chess_variants.
- David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.
- David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.
- David L. Parnas and P.C. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 12(2):251–257, February 1986.
- D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.
- James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.
- W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL <http://www.ifi.unizh.ch/req/events/RE06/>.
- W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.
- W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

- W. Spencer Smith, John McCutchan, and Jacques Carette. Commonality analysis of families of physical models for use in scientific computing. In *Proceedings of the First International Workshop on Software Engineering for Computational Science and Engineering (SECSE 2008)*, Leipzig, Germany, May 2008. In conjunction with the 30th International Conference on Software Engineering (ICSE). URL <http://www.cse.msstate.edu/~SECSE08/schedule.htm>. 8 pp.
- W. Spencer Smith, John McCutchan, and Jacques Carette. Commonality analysis for a family of material models. Technical Report CAS-17-01-SS, McMaster University, Department of Computing and Software, 2017.
- Bill Wall. History of chess sets and symbols. <https://www.chesscentral.com/pages/chess-sets-pieces-boards/a-history-of-chess-pieces-and-chess-sets.html>, 2003. Accessed: 2022-10-04.
- WCAG. Web content accessibility guidelines 2 overview. <https://www.w3.org/WAI/standards-guidelines/wcag/#intro>, 2018. Accessed: 2022-10-05.