

System Verification and Validation Report for Chess Connect

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

March 8, 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 functional 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 | 3 |
| 3.4 | Engine Mode | 4 |
| 3.5 | Beginner Mode | 6 |
| 4 | Nonfunctional Requirements Evaluation | 8 |
| 4.1 | Look and Feel | 8 |
| 4.2 | Usability and Humanity | 8 |
| 4.3 | Performance | 8 |
| 4.4 | Health and Safety | 8 |
| 4.5 | Precision and Accuracy | 9 |
| 4.6 | Capacity | 9 |
| 4.7 | Security | 9 |
| 5 | Unit Testing | 9 |
| 6 | Changes Due to Testing | 10 |
| 7 | Automated Testing | 10 |
| 8 | Trace to Requirements | 10 |
| 9 | Trace to Modules | 12 |
| 10 | Code Coverage Metrics | 12 |
| A | Reflection Appendix | 12 |

List of Tables

| | | |
|---|--|---|
| 1 | Active State Functional Requirements Results | 1 |
| 2 | Inactive State Functional Requirements Results | 2 |
| 3 | Normal Mode Functional Requirements Results | 3 |
| 4 | Engine Mode Functional Requirements Results | 5 |
| 5 | Beginner Mode Functional Requirements Results | 7 |

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

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. | Rework |
| 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 while game inactive. | 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 | Start game button pressed while game inactive. | System variable 'gameInProgress' set to true. | System variable configured correctly. | | Pass |
| GI-2 | User mode button pressed while game inactive. | User mode unchanged. | System variable configured correctly. | Design changed, user mode is now switchable (only) while a game is inactive. | Rework |
| GI-3 | Draw/resign button pressed while game inactive. | System variable 'gameInProgress' remains false. | System variable configured correctly. | | Pass |
| GI-4 | Piece moved while game inactive. | System variable 'currFEN' is unchanged. | System variable configured correctly. | | Pass |
| GI-5 | Draw/resign button pressed, or move made that results in stalemate or checkmate according to the rules of chess while game active. | Game termination and winner are displayed on LCD screen. | Display updates correctly. | | Pass |

Table 2: Inactive State Functional Requirements Results

3.3 Normal Mode

| Test | Input | Expected | Actual | Notes | Result |
|------|---|---|---------------------------------------|-------|--------|
| NB-1 | Piece moved while in normal mode. | Game state is updated to reflect piece movement. | Game state updated correctly. | | Pass |
| NB-2 | Resign button pressed while in normal mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| NB-3 | Draw button pressed while in normal mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| ND-1 | Game state updated while in normal mode. | Updated game state is transmitted to the web application via Bluetooth. | Game state transmitted correctly. | | Pass |
| NA-1 | Web application receives updated game state while in normal mode. | Update to game state is reflected on web application display. | Display updates correctly. | | Pass |
| NA-2 | Game termination occurs while in normal mode. | Game termination and winner are displayed on web application display. | Display updates correctly. | | Pass |

Table 3: Normal Mode Functional Requirements Results

3.4 Engine Mode

| Test | Input | Expected | Actual | Notes | Result |
|------|---|---|---------------------------------------|--|--------------|
| EB-1 | Piece moved while in engine mode. | Game state is updated to reflect piece movement. | Game state updated correctly. | | Pass |
| EB-2 | Resign button pressed while in engine mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| EB-3 | Draw button pressed while in engine mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| EB-4 | Engine moves transmitted from the web application to microcontroller. | Engine moves are displayed on the LCD screen. | Display updated correctly. | | Pass |
| ED-1 | Game state updated while in engine mode. | Updated game state is transmitted to the web application via Bluetooth. | Game state transmitted correctly. | | Pass |
| ED-2 | Engine moves are calculated by the web application. | Calculated engine moves are transmitted from the web application to the microcontroller via Bluetooth | Moves transmitted correctly. | Only one engine move currently calculated, more planned in future revisions. | Partial Pass |

| Test | Input | Expected | Actual | Notes | Result |
|------|---|---|---------------------------------|---|--------|
| EA-1 | Web application receives updated game state while in engine mode. | Update to game state is reflected on web application display. | Display updates correctly. | | Pass |
| EA-2 | Engine moves are calculated by the web application. | Calculated engine moves are displayed on web application display. | Engine moves are not displayed. | Not implemented, planned in future revisions. | TBD |
| EA-3 | Game termination occurs while in engine mode. | Game termination and winner are displayed on web application display. | Display updates correctly. | | Pass |

Table 4: Engine Mode Functional Requirements Results

3.5 Beginner Mode

| Test | Input | Expected | Actual | Notes | Result |
|------|---|---|---------------------------------------|---|--------|
| BB-1 | Piece moved while in beginner mode. | Game state is updated to reflect piece movement. | Game state updated correctly. | | Pass |
| BB-2 | Piece picked up and held while in beginner mode. | LEDs on board indicate legal moves. | Correct LEDs light up. | | Pass |
| BB-3 | Piece moved such that an illegal move is made while in beginner mode. | LEDs on board indicate illegal move. | Correct LEDs light up. | Not implemented, planned in future revisions. | TBD |
| BB-4 | Resign button is pressed while in beginner mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| BB-5 | Draw button is pressed while in beginner mode. | System variable 'gameInProgress' set to false. | System variable configured correctly. | | Pass |
| BD-1 | Game state is updated while in beginner mode. | Updated game state is transmitted to the web application via Bluetooth. | Game state transmitted correctly. | | Pass |

| Test | Input | Expected | Actual | Notes | Result |
|------|---|--|----------------------------|---|--------|
| BA-1 | User selcetions chess instructions in web application. | Web application displays detailed rules for how to play chess. | N/A | Not implemented, planned in future revisions. | TBD |
| BA-2 | Web application receives updated game state while in beginner mode. | Update to game state is reflected on web application display. | Display updates correctly. | | Pass |

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 | Result | Notes |
|-------|--------|-------|
| NFT-1 | | |

Table 6: Look and Feel Non-Functional Requirements Results

4.2 Usability and Humanity

| Test | Result | Notes |
|-------|--------|-------|
| NFT-2 | | |
| NFT-3 | | |

Table 7: Usability and Humanity Non-Functional Requirements Results

4.3 Performance

| Test | Result | Notes |
|-------|--------|-------|
| NFT-4 | | |
| NFT-5 | | |
| NFT-6 | | |
| NFT-7 | | |

Table 8: Performance Non-Functional Requirements Results

4.4 Health and Safety

| Test | Result | Notes |
|-------|--------|-------|
| NFT-8 | | |

Table 9: Health and Safety Non-Functional Requirements Results

4.5 Precision and Accuracy

| Test | Result | Notes |
|-------|--------|-------|
| NFT-9 | | |

Table 10: Precision and Accuracy Non-Functional Requirements Results

4.6 Capacity

| Test | Result | Notes |
|--------|--------|-------|
| NFT-10 | | |

Table 11: Capacity Non-Functional Requirements Results

4.7 Security

| Test | Result | Notes |
|--------|--------|-------|
| NFT-11 | | |
| NFT-12 | | |

Table 12: Security Non-Functional Requirements Results

5 Unit Testing

Unit testing is a crucial aspect of software development that involves testing individual units or components of a software application in isolation from the rest of the system. It provides us with a way to ensure that each unit of code is functioning as intended and that it integrates seamlessly with other parts of the software application. By identifying defects and bugs early in the development cycle, unit testing helps reduce the overall time spent on software development while also improving the quality and reliability of the final product. Additionally, unit tests serve as a form of documentation, helping us understand how different components of the system are supposed to interact and ensuring that future modifications do not break existing functionality. We implemented unit testing to ensure our project was robust, maintainable, and of high quality. As mentioned in the VnV Plan, the React Testing Library was used for the Javascript unit tests. Additionally, with the inclusion of many hardware related components in our project, most of our testing is done manually, leaving very few tests that require being automated. However, some of the automated tests that were performed are shown in the table below.

```
boolean inStalemate(string fenString)
```

| Test | Input | Expected | Actual | Result |
|------|---|----------|--------|--------|
| GA-1 | rnbqkbnr/- pppppppp- /8/8/8/8/- PPPPPPPP/- RNBQKBNR w KQkq - 0 1 | false | false | Pass |

```
boolean inCheckmate(string fenString)
```

| Test | Input | Expected | Actual | Result |
|------|--|----------|--------|--------|
| GA-1 | rnb1kbnr/- pppp1ppp- /8/4p3- /5PPq/8/- PPPPP2P/- RNBQKBNR w KQkq - 1 3 | true | true | Pass |

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.