

Project Title: System Verification and  
Validation Plan for Chess Connect

Author Name

November 1, 2022

# 1 Revision History

Table 1: Revision History

Date	Developer(s)	Change
2022-10-31	Jonathan Cels	NFR Testing
Date 2	1.1	Notes

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>iv</b>
<b>3</b>	<b>General Information</b>	<b>1</b>
3.1	Summary . . . . .	1
3.2	Objectives . . . . .	1
3.3	Relevant Documentation . . . . .	1
<b>4</b>	<b>Plan</b>	<b>1</b>
4.1	Verification and Validation Team . . . . .	1
4.2	SRS Verification Plan . . . . .	2
4.3	Design Verification Plan . . . . .	2
4.4	Implementation Verification Plan . . . . .	2
4.5	Automated Testing and Verification Tools . . . . .	2
4.6	Software Validation Plan . . . . .	2
<b>5</b>	<b>System Test Description</b>	<b>3</b>
5.1	Tests for Functional Requirements . . . . .	3
5.1.1	Area of Testing1 . . . . .	3
5.1.2	Area of Testing2 . . . . .	4
5.2	Tests for Nonfunctional Requirements . . . . .	4
5.2.1	Look and Feel . . . . .	4
5.2.2	Usability and Humanity . . . . .	5
5.2.3	Performance . . . . .	6
5.3	Traceability Between Test Cases and Requirements . . . . .	7
<b>6</b>	<b>Unit Test Description</b>	<b>7</b>
6.1	Unit Testing Scope . . . . .	7
6.2	Tests for Functional Requirements . . . . .	7
6.2.1	Module 1 . . . . .	7
6.2.2	Module 2 . . . . .	8
6.3	Tests for Nonfunctional Requirements . . . . .	8
6.3.1	Module ? . . . . .	9
6.3.2	Module ? . . . . .	9
6.4	Traceability Between Test Cases and Modules . . . . .	9

<b>7</b>	<b>Appendix</b>	<b>10</b>
7.1	Symbolic Parameters . . . . .	10
7.2	Usability Survey Questions? . . . . .	10

## List of Tables

1	Revision History . . . . .	i
	[Remove this section if it isn't needed —SS]	

## List of Figures

[Remove this section if it isn't needed —SS]

## 2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

## **3 General Information**

### **3.1 Summary**

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

### **3.2 Objectives**

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: “build confidence in the software correctness,” “demonstrate adequate usability.” etc. You won’t list all of the qualities, just those that are most important. —SS]

### **3.3 Relevant Documentation**

[Reference relevant documentation. This will definitely include your SRS and your other project documents (MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. —SS]

Author (2019)

## **4 Plan**

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

### **4.1 Verification and Validation Team**

[You, your classmates and the course instructor. Maybe your supervisor. You should do more than list names. You should say what each person’s role is for the project. A table is a good way to summarize this information. —SS]

## 4.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

[Remember you have an SRS checklist —SS]

## 4.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Remember you have MG and MIS checklists —SS]

## 4.4 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walk-throughs, code inspection, static analyzers, etc. —SS]

## 4.5 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

## 4.6 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

## 5 System Test Description

### 5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. —SS]

#### 5.1.1 Area of Testing<sup>1</sup>

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

#### Title for Test

##### 1. test-id1

Control: Manual versus Automatic

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

##### 2. test-id2

Control: Manual versus Automatic

Initial State:

Input:

Output: [The expected result for the given inputs —SS]



Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

### 5.1.2 Area of Testing2

...

## 5.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. —SS]

[Tests related to usability could include conducting a usability test and survey. —SS]

### 5.2.1 Look and Feel

#### Style

#### 1. NFT1

Type: Structural, Static, Manual

Initial State: Users have not seen the product before, and the product is in its initial game state.

Input/Condition: Users are asked to identify the product.

Output/Result: The majority of users are able to successfully identify the product as a chess set without any outside assistance within 10 seconds.

How test will be performed: A test group of people who do not play chess regularly are asked to identify the product immediately after it is revealed to them. The time it takes for them to identify the product will be recorded. The subjects must identify the product as a chess set within 10 seconds or less, averaged over the number of people in the test group.

### 5.2.2 Usability and Humanity

#### Learnability

##### 1. NFT2

Type: Structural, Static, Manual

Initial State: Product is in normal mode, a game has started, and the users have not interacted with the product before.

Input/Condition: Users are asked to use the product and move one specified piece from one square to another specified square on the board.

Output/Result: The majority of users understand which piece they moved and to where, and are able to identify that the web application has reflected their move in the virtual model within 30 seconds of studying the visual representation of the board state.

How test will be performed: A test group of people who do not play chess regularly are asked to move pieces on the chessboard. They are then asked to identify the move they just made as reflected in the web application model. The subjects must identify that the piece they moved on the board has also been moved on the web application's virtual model within 30 seconds or less, averaged over the number of people in the test group.

##### 2. NFT3

Type: Functional, Dynamic, Manual, Static etc.

Initial State: The product shall be representing the state of the game that is in progress. The pieces shall be in a legal position according to the rules of chess. The pieces shall not all be in their starting positions, and there is at least one of each type of piece (pawn, knight, bishop, rook, queen, king) on the board.

Input: Users are asked to identify the names of different pieces and squares based on their visual appearance in both the physical product and the web application.

Output: The majority of users are able to identify the names of pieces and squares based on their likeliness to historically used symbols and shapes.

How test will be performed: A test group of people who have played chess in the past or play chess regularly are asked to identify each of the pieces and squares from an in-progress game of chess using the system. The justification for this is to avoid piece identification based on their starting positions. The majority of the group should be able to visually identify every piece and square within 2 minutes of seeing the position for the first time.

### **5.2.3 Performance**

#### **Speed and Latency**

##### **1. NFT4**

Type: Structural, Static, Manual

Initial State: The product is in normal mode, and a game has started.

Input/Condition: Users are asked to pick up a piece and suspend it midair without placing it down.

Output/Result: The board shall visually indicate where the held piece is able to move according to the rules of chess within a specific time frame.

How test will be performed: A test group of people who have played chess in the past or play chess regularly are asked to pick up a specific piece and suspend hold it. The system will give visual indicators of where the held piece is able to move according to the rules of chess. The time between when they pick up the piece and when the visual response occurs is measured and recorded. This process is repeated 5 times per user. The individual and average times are recorded. The response times are then averaged over the entire test group. The average response time of the entire test group must be less than 3 seconds.

##### **2. NFT5**

Type: Structural, Static, Manual

Initial State: The results of the previous test, NFT4, have been measured and recorded.

Input/Condition: The results of NFT4.

Output/Result: The maximum recorded time of any individual response is within a specific time frame.

How test will be performed: The times measured in the previous test, NFT4, will be inspected. The maximum recorded individual time must be less than 10 seconds.

### **5.3 Traceability Between Test Cases and Requirements**

[Provide a table that shows which test cases are supporting which requirements. —SS]

## **6 Unit Test Description**

[Reference your MIS and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS has been completed. —SS]

### **6.1 Unit Testing Scope**

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

### **6.2 Tests for Functional Requirements**

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

#### **6.2.1 Module 1**

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

3. ...

### 6.2.2 Module 2

...

## 6.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

### 6.3.1 Module ?

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

### 6.3.2 Module ?

...

## 6.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

## References

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

## 7 Appendix

This is where you can place additional information.

### 7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC\_CONSTANTS. Their values are defined in this section for easy maintenance.

### 7.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]