Project Title: System Verification and Validation Plan for Chess Connect

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

November 2, 2022

1 Revision History

Date	Version	Notes
October 31, 2022	Arshdeep Aujla	Added section 3
Date 2	1.1	Notes

Contents

List of Tables

[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

The project name is Chess Connect. It is comprised of software and hardware components. The hardware will consist of a reactive chess set connected to a microcontroller. The microcontroller will relay information on the chess board in the form of LEDs of the possible moves the user can make. The software component of this project will consist of a web application that will reflect all of the chess piece's location on the physical board.

3.2 Objectives

The following objectives are the qualities that are the most important for the project.

- The hardware should reflect relevent information on the LEDs on the chess board
- The software component should reflect the physical chess board in near-real time
- The movement of the chess pieces should be recorded by the hardware

3.3 Relevant Documentation

The following documents are relevent to this project.

- SRS
- Hazard Analysis
- Requirements Document
- Design Document
- VnV Report

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 shoud 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 walkthroughs, 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 Testing1

[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 Area of Testing1

Title for Test

1. test-id1

   Type:

   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:

5.2.2 Area of Testing2

...

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

6.2.1 Game Active State

[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. GA-0

   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. GA-1

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Active State.

   Input: The user will press the New/Resign/Draw button.

   Output: The system will be changed to the Game Inactive State.

   Test Case Derivation: The game shall be in the Game Inactive State due to the New/Resign/Draw button being pressed.

   How test will be performed: The function that changes the game state will be run using the appropriate inputs. Then after it has ran we will check to see if the game state has been modified.

3. GA-3

   Type: Structural, Static, Manual

Initial State: The game is in the Game Active State.

Input: The user will switch to one of the user modes (Normal Mode, Engine Mode, Beginner Mode).

Output: The system will be changed to the selected user mode.

Test Case Derivation: The game shall be in the selected user mode due to the user mode appropriate switch being pressed.

How test will be performed: The function that changes the user mode will be run using the appropriate inputs. Then after it has ran we will check to see if the user mode has been modified.

4. GA-4

   Type: Structural, Static, Manual

   Initial State: The game is in a user mode.

   Input: The game is changed to a different user mode.

   Output: The system will follow the Chess Board section based on the user mode.

   Test Case Derivation: The game shall follow the Chess Board section based on the selected user mode.

   How test will be performed: The function that changes the user mode will be run using the appropriate inputs. Then after it has ran we will check to see if the appropriate Chess Board section is being followed.

5. GA-5

   Type: Structural, Static, Manual

   Initial State: The game is in a user mode.

   Input: The game is changed to a different user mode.

   Output: The system will follow the Data Transfer section based on the user mode.

   Test Case Derivation: The game shall follow the Data Transfer section based on the selected user mode.

How test will be performed: The function that changes the user mode will be run using the appropriate inputs. Then after it has ran we will check to see if the appropriate Data Transfer section is being followed.

6. GA-6

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Inactive State.

   Input: The game is changed to the Game Active State.

   Output: The game state will be reset to the starting position.

   Test Case Derivation: The game shall be reset to the starting position due to it entering the Game Active State.

   How test will be performed: The function that resets the starting position will be run using the appropriate inputs. Then after it has ran we will check to see if the game state has been reset to the starting position.

7. GA-7

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Active State.

   Input: The game results in a stalemate or checkmate.

   Output: The game state will be changed to the Game Inactive State.

   Test Case Derivation: The game shall be changed to the Game Inactive state due a stalemate or checkmate.

   How test will be performed: The function that modifies the game state will be run using the appropriate inputs. Then after it has ran we will check to see if the game state has been changed to the Game Inactive State.

8. GA-8

   Type: Structural, Static, Manual

   Initial State: The game is in a user mode.

Input: The game is changed to a different user mode.

Output: The system will follow the Web Application section based on the user mode.

Test Case Derivation: The game shall follow the Web Application section based on the selected user mode.

How test will be performed: The function that changes the user mode will be run using the appropriate inputs. Then after it has ran we will check to see if the appropriate Web Application section is being followed.

6.2.2 Game Inactive State

1. GI-1

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Inactive State.

   Input: The user will press the New Game button.

   Output: The system will be changed to the Game Active State.

   Test Case Derivation: The game shall be in the Game Active State due to the New Game button being pressed.

   How test will be performed: The function that changes the game state will be run using the appropriate inputs. Then after it has ran we will check to see if the game state has been modified.

2. GI-2

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Inactive State.

   Input: The user will try to switch to one of the user modes (Normal Mode, Engine Mode, Beginner Mode).

   Output: The system will be unchanged to the selected user mode.

   Test Case Derivation: The game state shall be unchanged due to the user mode appropriate switch being pressed in the Game Inactive State.

How test will be performed: The function that changes the user mode will be run using the appropriate inputs. Then after it has ran we will check to see if the game state has been modified.

3. GI-3

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Inactive State.

   Input: The user will press the Resign/Draw button.

   Output: The system will be unchanged.

   Test Case Derivation: The game shall be in the Game Inactive State due to the Resign/Draw button having no effect.

   How test will be performed: The function that changes the game state will be run using the appropriate inputs. Then after it has ran we will check to see if the game state is unchanged.

4. GI-4

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Inactive State.

   Input: The user will move a piece.

   Output: The board state is not sent to the web application.

   Test Case Derivation: The game shall be in the Game Inactive State due to the Resign/Draw button having no effect.

   How test will be performed: The function that changes the game state will be run using the appropriate inputs. Then after it has ran we will check to see if the game state is unchanged.

5. GI-5

   Type: Structural, Static, Manual

   Initial State: The game is in the Game Active State.

   Input: The game is terminated.

Output: The display the final game and message with the game termination type (stalemate, checkmate, resignation, draw).

Test Case Derivation: The game shall output the final game and message due to the game being terminated.

How test will be performed: The function that handles actions after game termination will be run using the appropriate inputs. Then after it has ran we will check to see if the final game and message with termination type are displayed.

6.2.3 Normal Mode

1. NM-1

Type: Structural, Static, Manual

Initial State: The game is in the Game Active State.

Input: The game is terminated.

Output: The display the final game and message with the game termination type (stalemate, checkmate, resignation, draw).

Test Case Derivation: The game shall output the final game and message due to the game being terminated.

How test will be performed: The function that handles actions after game termination will be run using the appropriate inputs. Then after it has ran we will check to see if the final game and message with termination type are displayed.

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]