SE 3XA3: Test Plan
Sudoku Solver

Team 08, SudoCrew
Rashad A. Bhuiyan (bhuiyr2)
Kai Zhu (zhuk2)
Stanley Chan (chans67)

March 11, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2022-02-28	0.0	Created Test Plan; started on General Information
2022-03-09	0.1	Updated Plan and Appendix, started System Test Description
2022-03-11	0.2	Updated Sections 3-6

1 General Information

1.1 Purpose

The purpose of this document is to describe the testing, validation, and verification procedures that will be implemented for our Sudoku Solver program. The majority of the unit testing will be accomplished through PyTest as our program is primarily written using Python. Structural testing will be done using individually-created testing classes for proper analysis of branching. Non-functional requirements will primarily be realized through manual testing as there is a heavy emphasis on visualization of our project. System testing is done before proof-of-concept demonstration, revision 0 demonstration, and the final demonstration.

1.2 Scope

This test plan provides a basis for testing the functionality of the extended implementation and every new addition to the original Sudoku Solver algorithm by TechWithTim. Since the extended implementation involves creating a web application and methods of playing the game, the scope of testing covers web-application navigation and management, Sudoku generation and solution algorithms, and image recognition algorithms through machine learning. This document provides testing methodologies that covers the scope of the program as well as outlining all methods and tools used for testing.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Naming Conventions and Terminology**

Terminology	Meaning
PoC	Proof of Concept, one of the deliverables of the 3XA3 project
SRS	Software Requirements Specification, the document outlining all requirements for this program
GUI	graphical user interface, the front-end of the program that users can view and interact
Structural Test	Testing that focusses on the internal structure and branching of the software
Functional Test	Testing derived from the SRS
Dynamic Test	Testing that involves test cases running during execution
Static Test	None-code testing done through code inspection
Manual Test	Testing that is done manually by people
Automated Test	Testing that is done using testing software such as PyTest
Unit Test	Testing that focusses on individual functions and methods
System Test	Testing the entire system as a whole rather than individual components
UI	User interface, the interface that allows for user interaction with the system.
PyTest	A Python library designed to help create tests for Python code.

1.4 Overview of Document

This document contains all information regarding the testing plan for the Sudoku Solver project. An overall plan will be addressed through a schedule as well as creation of a testing team. Furthermore, every functional and non-functional requirement from the SRS will be addressed and tested using various form of blackbox and whitebox testing. PoC testing will cover a specific sample of methods and will highlight some key differences of implementation between the original project and the current implementation.

2 Plan

2.1 Software Description

The purpose of this software application is to provide a comprehensive suite of tools for generating, recognizing, and solving Sudoku puzzles. The application will provide a web-based front-end with an intuitive interface to cater to users of different technical abilities on most modern hardware. Computer vision is also utilized to improve ease of use, by directly interfacing puzzles from print-media to the application.

2.2 Test Team

The team responsible for testing consists of Rashad Bhuiyan, Stanley Chan, and Kai Zhu.

2.3 Automated Testing Approach

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url: https://gitlab.cas.mcmaster.ca/bhuiyr2/sudokusolver_l02_grp08/-/raw/main/ProjectSchedule/Gantt_Sudoku.pdf

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Homepage Navigation

Display Buttons

1. FS-DB-1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. FS-DB-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3. FS-DB-3

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.1.2 Image Upload

Initial Upload

1. FS-IU-1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. FS-IU-2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

Image Interpretation

1. FS-II-1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. FS-II-2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:

Input:

Output:

How test will be performed:

Solution Output

1. FS-SO-1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. FS-SO-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.1.3 Manual Input

Board Display

1. FS-BD-1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. FS-BD-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

Solution Verification

1. FS-SV-1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. FS-SV-2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3. FS-SV-3

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

4. FS-SV-4

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.1.4 Play Sudoku

Board Generation

1. FS-BG-1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:

Input:

Output:

How test will be performed:

2. FS-BG-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3. FS-BG-3

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

Sudoku Gameplay

1. FS-SG-1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. FS-SG-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3. FS-SG-3

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

Input:

Output:

How test will be performed:

User Sudoku Verification

1. FS-USV-1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. FS-USV-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.2 Tests for Nonfunctional Requirements

3.2.1 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:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.2 Unit testing of output files

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 teams.