Ontario Tech University (UOIT)
Faculty of Engineering and Applied Science (FEAS)
Department Of Electrical, Computer, And Software Engineering

Principles of Software and Requirements (SOFE 2720)

Major Project | Deliverable #3 | Major Document & Acceptance Testing

Dr. Sukhwant Kaur Sagar
TA: Md Maruf
Winter 2019

Group Members
Aryan Kukreja (100651838)
Sunil Tumkur (100620430)
Amin Khakpour (100669547)
Armando Cuesta Leyva (100652479)

1. Introduction

The purpose of this document is to provide a detailed explanation of all the tests that were implemented on a functional version of a Sudoku Game that our group implemented.

1.1 About the Game

The game that our group implemented was a solo-user, non-timed Sudoku Game. The interface for the first iteration is kept simple and minimal, and all extra functionalities (such as score tracking) is left for future and later iterations.

The game was implemented using HTML, CSS, and Javascript:

- **Javascript** served as the backbone, or "engine" of the game. All the algorithms for creating and modifying the Sudoku Puzzle, as well as controlling the information on the screen were done through a Javascript File.
- **HTML** was used for the final display of the web-page for the game, and any messages that the Javascript engine needed to convey to the user.
- **CSS** was used for styling the web-page. It had a minimal role amongst the other languages.

In addition, an image obtained from the Internet was used as a background for the game when it was shown in a browser. All the code and image files are provided on our GitHub Repository.

2. User Story Implemented

For this deliverable, we implemented the following user story (first created in Deliverable #1 of this project):

> "*As an end-user, I would like to have the option of playing a game of Sudoku individually (solo).*"

The requirements were extracted from this user story in Deliverable #2 of this project. They were:

> *"Selecting the solo-mode will load the correct interface that consists of a single Sudoku puzzle. The Puzzle will be loaded with a set of pre-entered values."*

Program Testing

The only part of our code that needed to be tested using a Test Package was our Javascript code; because unlike HTML and CSS, Javascript could not be tested on a purely visual basis.

The testing library used was `Mocha`. This library was used to write test-cases for our code. For unit tests that could not be implemented using `Mocha`, a screenshot of the code, along with some useful documentation for how the function works, is provided

GUIDE FOR ACCEPTANCE TESTS:
https://openclassrooms.com/en/courses/4544611-write-agile-documentation-user-stories-and-acceptance-tests/4810081-writing-acceptance-tests

Acceptance Tests

ID	Given	When	Then	Expected Results	Pass/Fail
01	Sudoku game source code	Code of the game is written and complete. Code is then pushed to origin.	Code is pushed to origin and uploaded online where the developer can access it for future references	Developer is able to access and view the source code online through GitHub, an online software development platform	Pass

269 lines (203 sloc) | 9.34 KB

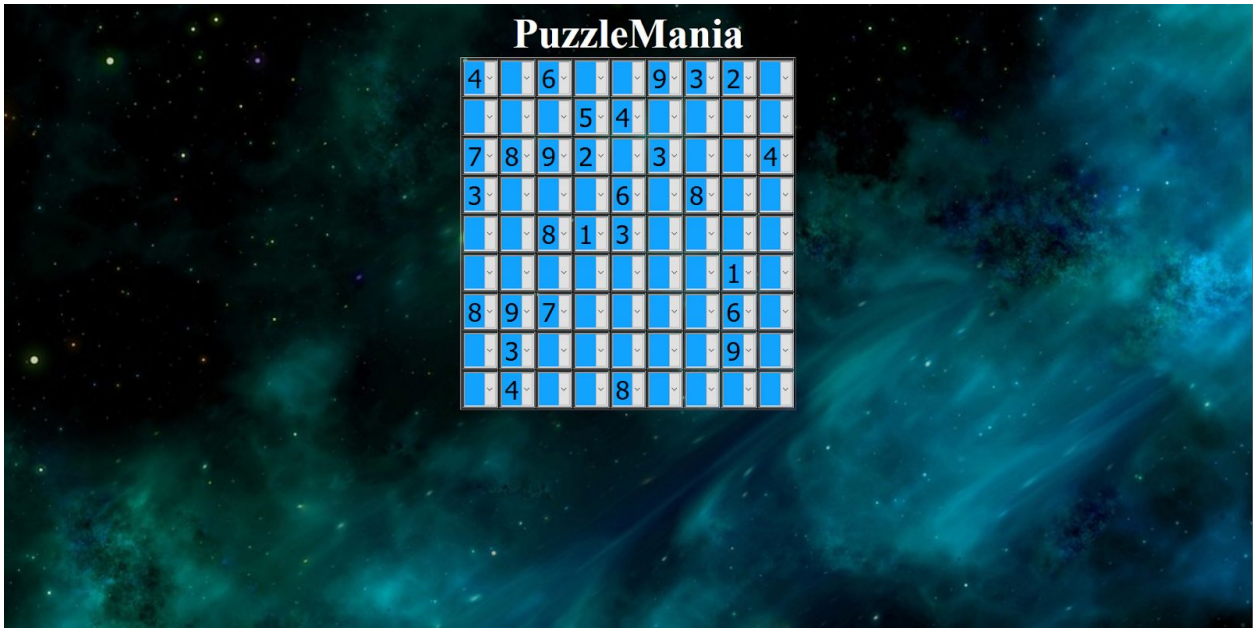
Raw Blame History

```
<!DOCTYPE html>

    <head>
        <link href="sudoku.css" rel="stylesheet" type="text/css">
        <script>
             var board;


                function random(max){
                    var value = Math.floor ( Math.random()*max);
                    return value;
                }

                    //convert my row and column to a quandrant #
                function getMyQ (r,c){

                    var qR, qC;
                    if (r<3)
                        qR = 0;
                    else if (r>5)
                        qR=2;
                    else
                        qR=1;

                    if (c<3)
                        qC = 0;
                    else if (c>5)
                        qC=2;
                    else
                        qC=1;

                    return "" + qR + "_" + qC;

                }


                    function changed( which){
                       // alert("" + which.selectedIndex);
                       //get the row and the column of "which"
                       var rc  = which.getAttribute("id");
                       //alert(rc);
                       var r = rc.slice(6,7);
                       var c = rc.slice(8,9);

```

Figure 1: Source code online for ID 01

ID	Given	When	Then	Expected Results	Pass/Fail
03	Developer has the game code	Developer adds in a dynamic background with vibrant cell colors	The game looks visually stunning	Game looks catchy visually and marketing team can then market the game	Pass

Figure 2: Visually Appealing Game for ID 03



ID	Given	When	Then	Expected Results	Pass/Fail
10	Sudoku game	User clicks the refresh button	A new sudoku game table will be ready to play	User will be able to have access to a new sudoku game	Pass

Figure 3: Version 1 Example

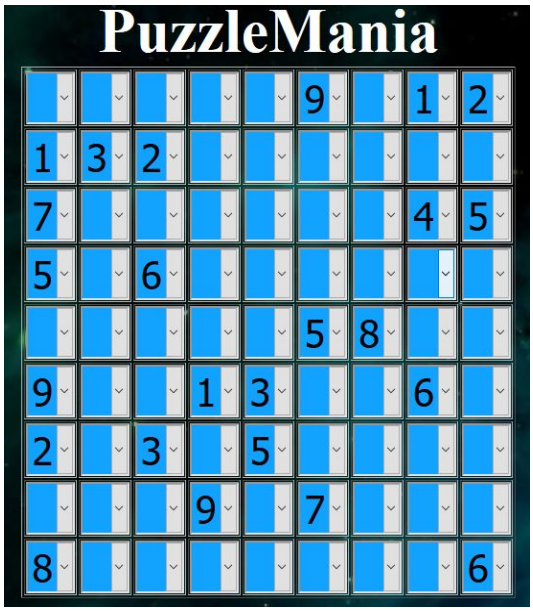
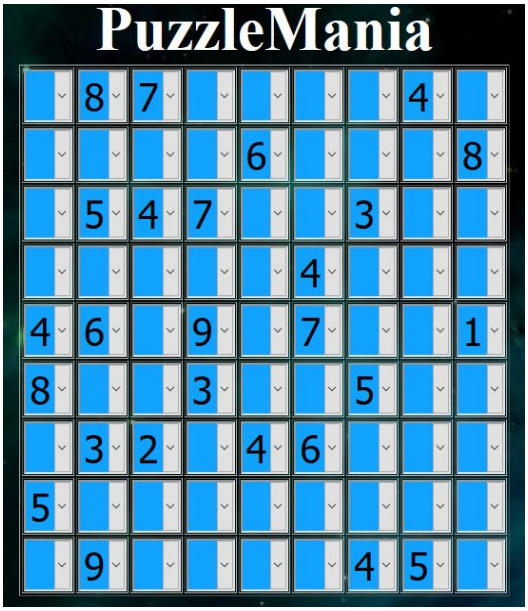


Figure 4: Version 2 Example



ID	Given	When	Then	Expected Results	Pass/Fail
11	Sudoku game	User fills out every column and row of the game correctly	User is treated with a “You win the game!” message on screen	User fills out the last correct cell of either the column and row and wins the game where a winner message is displayed	Pass

Figure 5: “You win the game!” message