SE 3XA3: Test Plan
DNA Says

Team #10, Team Name: DNA
Kareem Abdel Mesih (abdelk2)
John-Paul Dakran (dakranj)
Shady Nessim (nessimss)

October 29, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2016/10/25	1.0	Addition of General Information section
2016/10/29	1.1	Addition of Comparison to Existing Implementation section

1 General Information

1.1 Purpose

In the engineering process, verification and validation of the requirements outlined in the Software Requirements Specification (SRS) document is essential. This process is executed through a series of tests executed on the requirements to prove that the functionality of the game is correct. This document serves the purpose of outlining how the requirements will be validated and verified.

The implementation of the game DNA Says consists of numerous functional capabilities. These functional capabilities range from detecting user input to outputting a correct sound at a precise given time. The complete set of requirements will be broken down into specific and simple tests to prove the functionality of each specific requirement.

1.2 Scope

The main objective of this document is to outline an agreed upon set of tests that will be performed on the software system to validate its functionality. The scope of the testing for this game includes testing the animations, sounds, buttons, integration of the system, and all functional and non-functional requirements outline in the Software Requirements Specification (SRS) document.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
SRS	Software Requirement Specification
PoC	Proof of Concept
GUI	Graphical User Interface

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

1.4 Overview of Document

This document outlines a collection of information about the software system - DNA Says - that is in the process of creation. The test plan document begins by describing the software system and its functionality. It then proceeds to introducing the reader with the test team and the plan for testing - I.e. Testing tools and the testing schedule.

Next, the tests for functional and non-functional requirements will be described. Each test will have a type, initial state, input, output, and description of how the test will be performed. The same format will be used for the next section which outlines the tests for the proof of concept.

Proceeding, the reader will be introduced to a concise comparison between the original implementation and the implementation that is currently in the process of creation. Next the unit testing plan will be revealed to the reader which describes the unit testing of the internal functions and output files. The test plan document will be concluded by the appendix which will hold a list of symbolic parameters and survey questions for user testing.

2 Plan

2.1 Software Description

2.2 Test Team

2.3 Automated Testing Approach

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Input

- **Test #1: REQ**
  - Test ID: FT-UI-1
  - Type: Functional, Dynamic, Manual, Static etc.
  - Initial State:
  - Input:
  - Output:
  - Execution (How test will be performed):
- **Test #2: REQ**
  - Test ID: FT-UI-2
  - Type: Functional, Dynamic, Manual, Static etc.
  - Initial State:
  - Input:
  - Output:
  - Execution (How test will be performed):

3.1.2 Navigation

...

3.1.3 Display

3.1.4 Functionality

3.1.5 Constraints

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

- **Test #1: REQ**
  - Test ID:
  - Type: Functional, Dynamic, Manual, Static etc.
  - Initial State:
  - Input:
  - Output:
  - Execution (How test will be performed):
- **Test #2: REQ**
  - Test ID:
  - Type: Functional, Dynamic, Manual, Static etc.
  - Initial State:
  - Input:
  - Output:
  - Execution:

3.2.2 Area of Testing2

...

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

The implementation of DNA Says and Simon Says have many similarities and differences that will be outlined in this section. The main principle of the game stays constant however there are many improvements and modifications that have been added to truly make DNA Says unique.

The first significant difference between DNA Says and Simon Says is the fact that DNA Says has three modes. The original version of Simon Says only has one mode or one level which consists of 4 buttons that display the pattern to the user. In contrast DNA Says has three levels or three modes - each with a different implementation but overall the same theory from the

Simon Says implementation. The first mode - Kareem Says, consists of piano keys that will display the pattern to the user. The second mode - Shady Says, consists of 4 different coloured buttons that will display the pattern to the user. The final mode - JP Says, consists of 9 different coloured buttons that will display the pattern to the user.

The second critical difference between the two implementations is the 4 second timeout. The original implementation - Simon Says, has no timeout. This means when the user clicks a button and does not click the next button, the program will wait for the user to enter the next button. In contrast, DNAY Says has a 4 second timeout that is implemented. When a user clicks a button/key and does not click the next button within 4 seconds, the player loses.

The third significant difference between DNA Says and Simon Says is the protocol the program displays to the user when an incorrect button/key is inputted. The original version - Simon Says - displays the correct button that should have been clicked along with a sound that indicates an incorrect key has been inputted. The implementation of DNA Says makes the entire screen flash white 3 times and outputs a harmonic combination of notes that indicate an incorrect input.

Along with all the differences that truly make the game DNA Says truly unique, there are numerous similarities as the essence of the game is the same. Both implementations of the game have a main menu that allow the user to enter a mode and play the game. During a game, there is a score icon which displays the current score and there is also an exit button which will allow the user to return to the main menu.

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.