Requirements Document

AUTHORING

**TEAM MAMBERS:** Rabia Ejaz

Danilo Torres

****Syed Usman

**DATE:** March, 07, 2017

**COURSE:** EECS 2311

## PROFESSOR: Bil Tzerpos

CONTENTS

1.0 INTRODUCTION 3

1.1 Purpose 3

1.2 Scope 3

1.3 Background 3

1.5 Constraints 4

2.0 USER REQUIREMENT 4

3.0 DESCRIPTION OF REQUIRED FEATURES 5

4.0 ACCEPTANCE TEST CASES 8

5.0 SUFFICIENCY OF ACCEPTANCE CASES 9

6.0 GLOSSARY 9

7.0 REFERENCES 9

## 

1.0 INTRODUCTION

This requirement document is prepared by TEAM 5 to describe the user needs, required features, acceptance test cases and their sufficiency for this phase of the project. This is the third and final phase of our project in which we allow the user (such as an instructor) to construct the scenarios for the students. With a user-friendly GUI, they are able to create, edit, and play their custom scenarios.

##### 1.1 Purpose

The purpose of this requirement document is to provide sufficient information for the user to understand the “Authoring” phase. It explains how the program is expected to perform and also includes definitions of technical terms. The first two phases of the project were the simulator and player phases. If you have any concerns about the simulator or player, please refer to their documentation.

##### 1.2 Scope

In this phase, our aim is to create a GUI that is easy to understand, allowing the user to create their scenarios with all the options provided in the player. The Authoring app will create the playable file in the background, while showing the user a more simplified log of their scenario. The due date of player app was April 5 and we successfully completed our project within the given time.

##### 1.3 Background

Previously, the team was asked to create a simulator for the braille tutoring device which would display the buttons and cells that the user can interact with when they play a scenario. We also had to design a file format that could be read by a player to execute specific scenarios. Now the team is asked to create a program called the authoring app that would be working simultaneously with the simulator and player app. Our responsibility is to implement a complete software package for a program used by the visually-impaired through a device. We delivered our first phase on Feb 7, our second phase on March 7 and we will deliver final phase on April 5. All implementation and documentation is publicly available.

##### 1.5 Constraints

In every system, there are a few assumptions and constraints. In our system, we do not have the hardware available and are using a simulator instead. We assume that the user will follow instructions described in the manual correctly and reads all of the information provided.

Some Constraints of our system include

* Delivery date (April 5).
* Using the simulator and player code provided.

2.0 USER requirements

## Following are the general requirements of the user for the authoring app:

## Able to add all of the functions covered in the file format into the scenario.

## Interface is understandable with proper labelling.

## Scenario files can be created, saved, and opened afterwards.

## Complicated file format should be hidden, showing a simplified scenario.

## Able to edit, rearrange, and delete components of a scenario.

## Able to record voice that can be played.

## Able to play scenarios.

## Interface is clean and organized.

## Interface is simple to use; no cumbersome operations.

## Handles user errors.

3.0 DESCRIPTION OF REQUIRED FEATUES:

|  |  |
| --- | --- |
| REQUIREMENT ID | DESCRIPTION |
| UR 1 | There should be an option for each function covered by the file format in the authoring app. For example, the user should be able to add a message, question, sound, etc. The user should be able to completely customize and organize their scenarios. |
| UR 2 | Buttons, menus, and instructions are properly labelled on the GUI. The user should easily understand what each button does and how they can customize the scenario through the interface. |
| UR 3 | The user can create multiple scenarios which they can open whenever they want. Changes can be saved and previous versions are accessible. |
| UR 4 | The lower level file format should not be visible for the user, but text corresponding to their work on the scenario is displayed. When something is added to the scenario, the playable file is written properly but is not visible, while the user can see simple text. The text they see describes the flow of their scenario, and each component is numbered. |
| UR 5 | User is able to edit, rearrange, and delete different components in their scenario. They should be able to easily select the component they want to change and enter new parameters, or delete them. |
| UR 6 | User should be able to record their own voice and play it in the scenario. They should decide when to start and stop recording, as well as what to name the sound file. |
| UR 7 | User can play their custom scenarios after creating and saving them. They can test the flow of the components, then go back to edit. |
| REQUIREMENT ID | DESCRIPTION |
| UR 8 | The interface has a nice overall appearance and is organized. It is consistent throughout, and all buttons can fit nicely in the frames. Text can be read easily and there is no overlapping of buttons/text. |
| UR 9 | The user should not be required to undergo a large process just to enter in simple operations or edit their scenarios. The most efficient way of inputting information should be implemented. |
| UR 10 | The program should be able to handle the user’s errors. Errors such as the user entering the wrong parameter should be considered. Weird scenario flow does not count as a program error and the user should be able to edit the flow. |

4.0 Acceptance test cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Step #** | **Execution Procedure or Input** | **Expected result or output** | **Pass/Fail** |
| ONE | Running player app | Simulator opens with cells and buttons. The number of cells and buttons are same as specified in the file that has the scenario. | Pass |
| TWO | Ask the message if encounters “message command” | Greeting message played with background music. | Pass |
| THREE | Ask the question if encounters “question command” | User asked a question and given choices or either repeating the question or choosing the correct option | Pass |
| FIVE | User pressing button | Depending on the button pressed the program will proceed. Eg:If button1 is pressed then question repeat, button 2 correct answer. | Pass |
| SIX | Clear Cells | Pins in cell reset after every question scenario comes to an end | Pass |
| SEVEN | Pass the correct button number and check that is it giving error? | The system should not give error if we ask user to press the button that already exists. So, it should fail the test | Fail |
| EIGHT | Set String a in the file | It should display ‘a’ on simulator and ask a question accordingly | Pass |
| NINE | Put a “goto” command in file | Should go to the location specified by the goto command. | Fail  (the location does not exist) |
| TEN | Check question Response by pressing the button and comparing that with the argument gotten by question | Should get the same argument as the button pressed | Pass |

5.0 SUFFICIENCY OF ACCEPTANCE CASES

6.0 GLOSSARY

|  |  |
| --- | --- |
| WORDS | DEFINITIONS |
| API | Application Programming Interface |
| UR | User Requirement |
| GUI | Graphical User Interface |

7.0 REFERENCES

We received help from our professor and TA to develop the authoring app. In addition, we searched through various APIs provided by Java Oracle and used numerous free libraries from FreeTTS, Eclipse, and Java.