Requirements Document: Authoring App Version

Team 3 Members:
Eric Dao, 213551379
Dong Jae Lee, 214461560
Siddharth Bhardwaj, 21358439

EECS 2311 – Software Development Project
Instructor: Bill Tzerpos
Due: April 5, 2017

Table of Contents

1. Introduction:

This part of the requirements document is to provide a description of the project and an explanation as to why the program needs to satisfy the customers' needs and the acceptance test cases that show the correctness of the program.

The rest of this document will explain the features and how it satisfies the customer's needs, in addition to the acceptance test cases to validate that the software was developed correctly.

1.1 Purpose:

The project for this course is to develop a software that will function accordingly with a hardware device to teach young, vision-impaired adolescents how to read Braille. The hardware device will use the 8-pin Braille cell system, and will have Braille cells and physical buttons. The project will also include a program to write scenario files for the hardware device, so that the adolescents can have a better interaction.

1.2 Final Stage:

For this final stage of the project, our group has developed a program that allows the user to create a new scenario text file based on how the user wants the scenario to play out. The program can open and edit any existing scenario text file, along with being able to record their own audio file which can be included in the scenario text file. The authoring app program works alongside the previous two stages of the project, as the player program can run and simulate the scenario files while the simulator program will display the cells and buttons.

1.3 Customer Needs and Acceptance Test Case:

The authoring app program will contain the necessary implementations needed to provide complete functionality for the customer. For this to happen, the authoring app program must implement all the features that the player program can read, along with satisfying all acceptance test cases to show correct functionality. Otherwise, if the acceptance test fails, then the program ends up being useless to the customer as the hardware device will not function properly.

2. Features:

The authoring app program contains all the features necessary to maximize the customer's use of the software:

1. Player program must be able to create a new blank scenario file.
2. The player program must be able to open and edit an existing scenario file.
3. Ability to performing voice recording and saving that audio file.
4. Ability to save changes to a scenario file.
5. Ability for the user to add commands into the scenario file, via buttons, without having to expose the underlying format of the scenario file. (Examples: Adding a sound file to be played, adding normal text, changing the cell display, etc. into the scenario file.)
6. Displays to the user any error messages when entering incorrect inputs for the commands to be added to the scenario file, as all commands must follow the scenario file format.
7. Ability to edit the number of cells and buttons in the scenario file, so that it is more adaptable to the device.
8. Ability to delete any line from the scenario file.
9. Ability to allow the user to create questions and answers, without having to understand the underlying format or concept behind it.
10. Allows the user to traverse to different parts of the scenario file, in order to edit or change those parts.
11. Ability to play current scenario file.

By being able to implement all these features, it provides the customer with enough functionality to use the hardware Braille device and maximize learning and interaction.

3. Acceptance Test Cases:

For this section about the acceptance test cases, it will describe the features of the authoring program and that these features will show that the test cases are sufficient for correctness.

Table 1: This table describes the relationship between the necessary features, how it is represented in code and the output after running that feature.

Feature description	Method being called	Output
Creates a blank scenario file with default cell and button numbers.	newFile ();	Changes the text area to display just the cell and button numbers. Nothing else in the scenario file as it is a new one.
Move to a different section of the scenario file.	drawMap ();	By clicking one of the buttons on the “map”, the text area displays the section of the scenario file corresponding to that button.
Save an edited scenario file.	save (“NewScenario.txt”);	A new scenario file is saved in the directory “SampleScenarios” or an existing scenario file was overwritten with changes.
Records sound spoken by the user.	captureAudio();	A new .wav sound file has been created which contains the user’s recording.
Opens an existing scenario file.	open (“Scenarion_One.txt”);	If the selected scenario file is of a valid format, the text area will display the scenario file and that it can be edited.
Skip to another part of the scenario file.	skip (“here”);	Skips to the next occurrence of the identifier “here” in the scenario file.
Adds a line or command to the scenario file.	add (“/~sound:”, “Adding sound file”);	A new line is inserted into the scenario file which contains a specific command or normal text.
Removes a certain line in the scenario file.	remove( );	Removes the line that is being highlighted in the text area.

These acceptance test cases show that all the above functionality is suitable for use by the customer, as the customer does not require any other features that satisfies their requirements. For more details, read the user manual and testing documentation.