Text2SpeechEditor

Product Backlog Specification

VERSIONS HISTORY

Date	Version	Description	Author
2/23/2020	<1.0>	1st version of the requirements definition document	A. Zarras

1 Introduction

This document defines the basic requirements for Text2SpeechEditor, an application for writing documents and transforming them to audio.

1.1 Purpose

The objective of this project is to develop a simple editor that allows preparing text documents and transforming them to audio. Such an application can be of much help for people who have speaking problem. Moreover, it could be a useful tool for preparing presentations that are automatically played without a physical speaker. The application consists of a graphical user interface, a back-end that enables the text editing process and the part that transforms text to speech, which will be based on external text to speech libraries.

1.2 Document Structure

The rest of this document is structured as follows. In Section 2 we focus on the development process that shall be followed and other scoring and organizational issues. The basic domain properties related to the project are discussed in Section 3. Sections 4 and 5 provide the Product Backlog, i.e., the "raw" functional and non-functional requirements that should be further analyzed to drive the design, implementation and testing of the application.

2 Development process and organization issues

To realize the project we shall rely on a Scrum approach. There will be 2 Sprints. In the **1st sprint**, we have to implement **user stories 1 to 6** and their **acceptance tests**. The **2nd sprint**, should develop the **rest of the user stories** and their **acceptance tests**. The deadlines for the Sprints are:

1. Deadline for **1st Sprint is 6/4/2020**
2. Deadline for **2nd Sprint (final) is 11/5/2020**

2.1 Deliverables

Definition of "done" story: A user story is done if it is **implemented correctly** and validated with one or more appropriate **acceptance tests**.

For **each Sprint**, Scrum team shall **deliver (via turnin) the project implementation and a Sprint report,** according to the given **Sprint report template, describing the "done" user stories that have been developed so far**. The team shall also deliver a printed version of the report.

2.2 Scoring

1. Working implementations of the **user stories** is **35%** of the total score.
2. **Acceptance tests** for the implemented user stories is **25%.**
3. Usage of recommended **patterns** to satisfy the extensibility and maintainability requirements is **30%** the total score.
4. Quality of **reporting** is **10%** of the total score.

3 Domain Properties

The Text2SpeechEditor enables the creation and management of text documents. A text document is characterized by the following basic properties:

- The author.
- The title
- The date that it was created.
- The date that it was last saved.
- The contents of the document. The contents of the document are structured as **a list of lines**. Each **line** consists of a **list of words**.

4 Functional Requirements / User Stories

- [US-1] As a user, I want to create a new empty document, by giving its title and author. The application should automatically record the creation date.
- [US-2] As a user, I want to edit the contents of the document, via the application's user interface.
- [US-3] As a user, I want to save the contents of the document to disk by providing a particular filename. The application should automatically record the save date.
- [US-4] As a user, I want to open the contents of an existing document from disk by providing a particular file path, or by browsing the file system folders.

- [US-5] As a user, I want to transform the contents of the document to speech.
- [US-6] As a user, I want to select a line and transform it to speech.
- [US-7] As a user I want to transform the contents of the document to speech in reverse, i.e. play the last word of the last line first and so on.
- [US-8] As a user I want to select a line and transform it to speech in reverse, i.e. and play the last word first and so on.
- [US-9] As a user I want to encode the contents of the document and then transform them to speech.
- [US-10] As a user I want to select a line, encode it and transform it to speech.
- [US-11] As a user I want to tune the text encoding technique. In particular the application should support at least the following encoding strategies:
    - Atbash: The Atbash cipher is formed by taking the alphabet and mapping it to its reverse, so that the first letter becomes the last letter, the second letter becomes the second to last letter, and so on.
    - Rot-13: Rot-13 is a letter substitution cipher that replaces a letter with the 13th letter after it, in the alphabet. Rot-13 is a special case of the Caesar cipher, which was developed in ancient Rome.
- [US-12] As a user I want to be able to tune the audio parameters, i.e., the volume, the speech rate and the pitch.
- [US-13] As a user I want to be able to store a sequence of actions/commands (e.g. open file, edit contents, play contents, play line, save file) that I have performed in main memory and re-execute them multiple times.

5 Non Functional Requirements

[NF1] **Extensibility**: In software engineering, extensibility is a design principle where the implementation takes future growth into consideration. In the case of this project, extensibility concerns the text to speech API that is going to be used. In particular, the application should work with a specific API but it should be easy to change the API for another, without having to make extensive changes to the implementation. Extensibility further concerns the encoding strategies that are supported. Specifically, it should be possible to easily add new encoding strategies to the application, without having to make extensive changes to the implementation. To achieve these extensibility aspects the application should be designed according to the well known open-closed principle and exploit GoF design patterns [1] like Factory, Adapter, Strategy, Template Method, etc.

[NF2] **Usability**: In software engineering usability concerns the ease of use and learnability. In the context of this project the application should provide a simple and user-intuitive interface. The application should also provide help, in the form of user guidelines, concerning its main functionalities, and the contents of the different pattern templates.

[NF3] **Performance**: Performance concerns the time to transform text to speech. At least 9 times out of 10, the transformation of a regular word with < 10 characters should not take more than 2 secs to be transformed to speech.

6 IDE, Java and External API Requirements/Constraints

The application GUI should be implemented using the **Java Swing API**. The application should be compatible at least with **Java 11**. Text to speech translation should be based at least to the Free TTS API (https://freetts.sourceforge.io/)

- Eclipse
- JUnit
- Java 11
- Swing
- FreeTTS

7 References

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.