AdvancedTextToSpeechApp

Product Backlog Specification

VERSIONS HISTORY

Date	Version	Description	Author
3/8/2021	<1.0>	1st version of the requirements definition document	A. Zarras

1 Introduction

The objective of this project is to develop an application that allows to edit and transform documents to audio. The documents may be in various formats. Specifically the application supports at least Microsoft Word (.docx) and Excel documents (.xlsx). For privacy and security reasons the contents of the documents may be encoded with different encoding mechanisms. The target users of the application may be people who have reading problems or visually impaired people. 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.

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. Sections 3 and 4 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. Each team shall organize a number of sprints during which the team shall implement **user stories** from the project backlog and their **acceptance tests**. The deadline for the project is: **26/5/2021**

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**.

At the end of the project the Scrum team shall **deliver (via turnin)**

- **The project implementation.**
- **A Sprint report,** according to the given **Sprint report template (SprintReport-v0.doc), describing the sprints that they performed and the "done" user stories that have been developed during each sprint**. The report shall also comprise the specification of **detailed use cases, derived from the given user stories,** the **detailed design** of the application and **CRC cards** that document the responsibilities and the collaborations between the different part of the application.
- **A DEMO video**, about 15' minutes, using a screen capture tool like ActivePresenter for example. In the demo you should illustrate that the user stories of the application are working. The demo further demonstrate the acceptance tests that have been developed for the application.

    - Put the demo video on GitHub, Google Drive or youtube. In the project folder that you turnin include a txt file named DEMO-LINK.txt that contains the link that points to the folder that contains the video.

- Turn in the **project** and the other deliverables using **turnin deliverables@myy803 <your-project>.zip**, where your-project is a zip file of your Eclipse project.

2.2 Scoring

1. Working implementations of the **user stories** is **40%** of the total score.
2. **Acceptance tests** for the implemented user stories is **20%.**
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 Functional Requirements / User Stories

ID	AS A <User Type>	I WANT <An Action>	SO THAT <A Benefit/Value>
US1	User	To open a file that is stored on disk and view its contents. The application should allow me to open different kinds of files. The application should support at least Microsoft Word (.docx) and Excel (.xlsx) documents. The application should also allow me to open files with encoded contents. The application should support different encodings, including Atbash [1]and Rot-13[2].	I can edit the contents of the file and transform them to audio.
US2	User	To be able to edit the contents of the file.	I can produce a new version of the file that I opened.
US3	User	To save the contents of the file that I opened on disk. The application should allow me to specify the format of the file, the encoding (if	I can store a new version of the file that I opened.

[1] 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.

[2] 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.

		any) and the filename.	
US4	User	Transform the contents of the file that I opened to audio.	I can listen what is in the file instead of having to read it.
US5	User	Select a part of the contents of the file (e.g from line X to line Y) that I opened and transform them to audio.	I can listen only a part of the contents of the file instead of all.
US6	User	To tune the audio parameters, i.e., the volume, the speech rate and the pitch.	Customize the audio to my needs.
US7	User	To activate a recording operation that keeps track of a sequence of text to audio transformation actions/commands.	I can re-execute them multiple times.
US8	User	To replay the recorded sequence of actions.	I can listen again the contents of the file that I opened.
US9	User	To de-activate the recording operation.	I can clean up the sequence of actions that have been recorded.

4 Non Functional Requirements

[NF1] **Maintainability:** In software engineering, maintainability is the degree of effectiveness and efficiency with which a product or system can be modified by the maintainers.

- In the case of this project, one primary concern is the formats of the input/output files supported by the application. In particular, the application should support at least **Microsoft Word files (.docx)** and **Microsoft Excel files (.xlsx)** but it should also be easy to support further file formats in the future, without having to make extensive changes to the implementation.

- Another concern is the different **strategies** that can be used for **encoding**. Specifically, it should be possible to easily add new strategies to the application, without having to make extensive changes to the implementation.

- Finally, a third issue is **the text to speech API** that will be used from the transformation of the text to audio. In particular, the design of the application should allow to easily substitute the API that is used for another in a future version without having to make extensive changes to the application.

To achieve these maintainability concerns the application should be designed according to the well known open-closed principle and exploit GoF design patterns [1] like Factory, Adapter, Facade, Strategy, Decorator, 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 of the application.

[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.

5 IDE, Java and External API Requirements/Constraints/Recommendations

The application should be implemented in **Java**. You can use the **Eclipse IDE** for this purpose. The application should be compatible at least with **Java 8**. To write automated tests for the application use **JUnit.** A well-known toolkit that can be used to the implementation of the GUI is **Java Swing.** A project that provides useful libraries that can be used for parsing Microsoft Word and Excel files is **Apache POI** (https://poi.apache.org/download.html). Text to speech translation can be based on the **Free TTS API** (https://freetts.sourceforge.io/).