SEPTeTS

Sprint Report

Team <INSERT TEAM NAME HERE>

Spyros Mantelos 4104

Christos Georgiou Mousses 4206

VERSIONS HISTORY

Date	Version	Description	Author
	0.1		Christos Georgiou Mousses Spyros Mantelos
	0.2		Christos Georgiou Mousses Spyros Mantelos
	0.3		Christos Georgiou Mousses Spyros Mantelos
	0.4		Christos Georgiou Mousses Spyros Mantelos
	0.5		Christos Georgiou Mousses Spyros Mantelos
	0.6		Christos Georgiou Mousses Spyros Mantelos
	0.7		Christos Georgiou Mousses Spyros Mantelos
	0.8		Christos Georgiou Mousses Spyros Mantelos
	0.9		Christos Georgiou Mousses Spyros Mantelos
	1.0		Christos Georgiou Mousses Spyros Mantelos

1 Introduction

This document provides information concerning the development and general architecture of the **Simple Editor Providing Text To Speech** (acronym **SEPTeTS**) project including the test cases and design patterns we used to structure the project.

1.1 Purpose

Accessibility is a subject of great gravity that industry standards must take into consideration. Developing applications with accessibility in mind, enables people with disabilities to use applications that otherwise they wouldn't, and provides them with an easy way to use applications effortless. With **SEPTeTS** we try to tackle these issues by offering a simple document editor with a text to speech implementation so that users with disabilities such as low vision can easily use. It provides a simplified and functional user interface with Quality of Life features such as an audio transform system that can record a sequence of audio transformations and replay them at any time allowing users with disabilities to have an intuitive user experience.

1.2 Document Structure

The rest of this document is structured as follows:

Section 2 describes out Scrum team and specifies the project's sprint backlog.

Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

Test	US1	US2	US3	US4	US5	US6	US7	US8	US9
CommandsFactoryTest									
createCommand	X	X	X	X	X	X	X	X	X
NewFileTest									
handle									
DocumentReaderFactoryTest									
createReader	X								

DocumentReaderTest									
excelReader	X								
wordReader	X								
textReader	X								
ReaderDecoratorAtbashTest									
decrypt	X								
ReaderDecoratorRot13Test									
ReaderDecoratorRot13:decrypt	X								
DocumentTest									
openAndSave	X		X						
getPath			X						
setContents		X							
clearContents									
TTSFacadeTest									
setVolume						X			
setPitch						X			
setRate						X			
DocumentWriterFactoryTest									
createDocumentWriter			X						
DocumentWriterTest									
excelWriter			X						
wordWriter			X						
texWriter			X						
WriterDecoratorAtbashTest									
encrypt			X						

WriterDecoratorRot13Test									
encrypt			X						

2.1 Scrum team

Product Owner	Apostolos Zarras
Scrum Master	Christos Georgiou Mousses, Spyros Mantelos
Development Team	Christos Georgiou Mousses, Spyros Mantelos

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**12/03/2021**	**12/04/2021**	**4**	**US1, US2, US3**
2	**13/04/2021**	**28/04/2021**	**2**	**US4, US5, US6**
3	**29/04/2021**	**13/05/2021**	**2**	**US7, US8, US9**

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

3.1 CreateNewFile

Use case ID	CreateNewFile
Actors	User

Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks at the **New File** button under the **File** menu or presses **Ctrl+N** 2. The contents of the current document are cleared 3. The contents of the text area are cleared 4. The path of the new document is set to nothing
Post conditions	A new document is created for the user to edit

3.2 OpenFile

Use case ID	OpenFile
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks at the **Open File** button under the **File** menu or presses **Ctrl+O** 2. A file chooser appears. 3. The user selects a file of the supported formats 4. An encoding dialog appears 5. The user selects the encoding 6. The user clicks the Confirm button 7. The contents of the document are decrypted 8. The contents of the document are shown in the text area
Alternative flow 1	1. The use case starts when the user clicks at the **Open File** button under the **File** menu or presses **Ctrl+O** 2. A file chooser appears. 3. The user selects a file of non-supported format 4. An encoding dialog appears 5. The user selects the encoding

	6. The user clicks the Confirm button or closes the dialog 7. Nothing is shown in the text area
Alternative flow 2	1. The use case starts when the user clicks at the **Open File** button under the **File** menu or presses **Ctrl+O** 2. A file chooser appears. 3. The user selects file of the supported formats 4. An encoding dialog appears 5. The user closes the dialog 6. The contents of the file are shown in the text area
Alternative flow 2	1. The use case starts when the user clicks at the **Open File** button under the **File** menu 2. A file chooser appears. 3. The user presses cancel or closes the file chooser 4. Nothing happens
Post conditions	If a document was opened it's ready for the user to edit

3.3 SaveFile

Use case ID	SaveFile
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Save** button under the **File** menu or presses **Ctrl+S** 2. If the document has a path associated with it 2.1. The contents of the document are saved in the disk on that path
Alternative flow 1	1. The use case starts when the user clicks the **Save** button under the **File** menu or presses **Ctrl+S** 2. If the document has not a path associated with it 2.1. **SaveFileAs** use case starts
Post conditions	The document is saved in the disk

3.4 SaveFileAs

Use case ID	SaveFileAs
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Save As…** button under the **File** menu 2. A save dialog appears 3. The user selects the name of the file and the extension of it 4. An encoding dialog appears 5. The user selects an encoding technique(s) 6. The file is encrypted 7. The file is saved in the disk
Alternative flow 1	1. The use case start when the user clicks the **Save** button under the **File** menu when the opened document has not a path associated with it 2. A save dialog appears 3. The user selects the name of the file and the extension of it 4. An encoding dialog appears 5. The user selects an encoding technique(s) 6. The file is encrypted 7. The file is saved in the disk
Post conditions	The document is saved in the disk

3.5 Exit

Use case ID	Exit
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Exit** button under the **File** menu or presses **Ctrl+Q**

	2. The application terminates
Post conditions	The application is terminated

3.6 TransformContent

Use case ID	TransformContent
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Transform Contents** button under the **Transform** menu or presses **Ctrl+T** 2. If there is text in the text area 2.1. If the recorder is active 2.1.1. Log transformation 2.2. The text in the text area is transformed into sound and is played
Alternative flow 1	1. The use case starts when the user clicks the **Transform Contents** button under the **Transform** menu or presses **Ctrl+T** 2. If there is no text in the text area 2.1. Nothing happens

3.7 TransformSelection

Use case ID	TransformSelection
Actors	User
Pre conditions	The user has selected some text from the text area
Main flow of events	1. The use case starts when the user clicks the **Transform Selection** button under the **Transform** menu or presses **Ctrl+E** 2. If some text is selected in the text area 2.1. If the recorder is active 2.1.1. Log transformation

	2.2. The selected text is transformed into sound and is played
Alternative flow 1	1. The use case starts when the user clicks the **Transform Selection** button under the **Transform** menu or presses **Ctrl+E** 2. If some text is not selected in the text area 2.1. Nothing happens

3.8 OpenPreferences

Use case ID	OpenPreferences
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Preferences** button under the **Transformation** menu or presses **Ctrl+P** 2. A preferences dialog is shown 3. The user can change the volume pitch and rate variables 4. If at any time the **Test** button is pressed 4.1. The preferences are saved 4.2. A sample sentence is played 5. The user clicks the **Apply** button 6. The preferences are saved 7. The dialog closes
Post conditions	The voice of the speaker has the volume and/or pitch and/or rate changed

3.9 Start Recording

Use case ID	StartRecording
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Start Sequence Recording** button under the **Recording** menu or presses **Ctrl+Space**

	2. The status of the recorder is changed to active
Post conditions	The status of the recorder is active

3.10 StopRecording

Use case ID	StopRecording
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Start Sequence Recording** button under the **Recording** menu or presses **Ctrl+.** 2. The status of the recorder is changed to non-active
Post conditions	The status of the recorder is non-active

3.11 ReplayRecording

Use case ID	ReplayRecording
Actors	User
Pre conditions	The user has recorded a series of audio transformations
Main flow of events	1. The use case starts when the user clicks the **Replay Sequence** button under the **Recording** menu or presses **Ctrl+R** 2. If the recording sequence has audio transformations 2.1. The recording sequence is played
Alternative flow 1	1. The use case starts when the user clicks the **Replay Sequence** button under the **Recording** menu or presses **Ctrl+R** 2. If the recording sequence is empty 2.1. Nothing happens

3.12 ClearRecording

Use case ID	ClearRecording
Actors	User
Pre conditions	The user has recorded a series of audio transformations
Main flow of events	1. The use case starts when the user clicks the **Clear Recording** button under the **Recording** menu or presses **Ctrl+L**. 2. If the recording sequence has audio transformations 2.1 The replay sequence is cleared
Alternative flow 1	1. The use case starts when the user clicks the **Clear Recording** button under the **Recording** menu or presses **Ctrl+L**. 2. If the recording sequence is empty 2.1 Nothing happens
Post conditions	The recording sequence is empty

3.13 OpenManual

Use case ID	OpenManual
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **Manual** button under the **Help** menu 2. The operating system's default browser opens the GitHub repository page of this application

3.14 OpenAbout

Use case ID	OpenAbout
Actors	User
Pre conditions	The user has started the application
Main flow of events	1. The use case starts when the user clicks the **About** button under the **Help** menu

	2. A dialog appears showing info about the application

4 Design

4.1 Architecture

<Specify the overall architecture for this release in terms of a **UML package diagram**.>

4.2 Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

Command Package

Class Name: About	
Responsibilities: ▪ Command that displays information about the program in a new window.	**Collaborations:**

Class Name: Clear	
Responsibilities: ▪ Command that clears anything that the ReplayManager has recorded.	**Collaborations:** ▪ DocumentCommand

Class Name: CloseProgram	
Responsibilities: ▪ Command that exits the program.	**Collaborations:** ▪ Speaker

Class Name: CommandsFactory	
Responsibilities: ▪ A factory that is used to create the correct DocumentCommand and give it the Document object passed.	**Collaborations:** ▪ DocumentCommand ▪ NewFile

	▪ OpenDocument ▪ SaveDocument ▪ SaveDocumentAs ▪ Preferences ▪ CloseProgram ▪ DocumentToSpeech ▪ SelectionToSpeech ▪ About ▪ Manual ▪ StartRec ▪ StopRec ▪ Replay ▪ Clear

Class Name: DocumentCommand	
Responsibilities: ▪ Abstract class that has a Document and an action it performs usually involving that Document.	**Collaborations:** ▪ Document

Class Name: DocumentToSpeech	
Responsibilities: ▪ Command that takes all the text its document contains and passes it to the TTSFacade to be transformed into speech.	**Collaborations:** ▪ ReplayManager

Class Name: Manual	
Responsibilities: ▪ Opens in the systems default browser this report.	**Collaborations:**

Class Name: NewFile

Responsibilities:	Collaborations:
▪ Command that clears the contents of its Document, meaning the text stored and displayed and the path.	▪ DocumentCommand

Class Name: OpenDocument	
Responsibilities:	**Collaborations:**
▪ Command that allows the user to select and open a File from his computer. After selecting a File he also choses if the file has to be decrypted in any way.	▪ DocumentCommand ▪ SimpleFileChooser

Class Name: Preferences	
Responsibilities:	**Collaborations:**
▪ Command that opens a new window where the user makes adjustments on the TTS volume, pitch and word rate.	▪ PreferencesController

Class Name: Replay	
Responsibilities:	**Collaborations:**
▪ Command that starts the replay of every command the ReplayManager recorded.	▪ ReplayManager

Class Name: ReplayManager	
Responsibilities:	**Collaborations:**
▪ Is responsible for recording a sequence of commands when the user wishes ▪ Can replay the sequence of recorded commands ▪ Can clear the recorded sequence of commands	▪ DocumentCommand

Class Name: SaveDocument	
Responsibilities: ▪ Command that saves a file to its existing location. ▪ If a file doesn't have an existing location works the same way with SaveDocumentAs	**Collaborations:** ▪ DocumentCommand ▪ SimpleFileChooser

Class Name: SaveDocumentAs	
Responsibilities: ▪ Command that opens the FileChooser so the user can select where to save his file ▪ User is also asked to select an encoding if he want to encode the file contents	**Collaborations:** ▪ DocumentCommand ▪ SimpleFileChooser

Class Name: SelectionToSpeech	
Responsibilities: ▪ Takes the text the user selected from its document and passes it to the TTSFacade to be transformed into speech.	**Collaborations:** ▪ DocumentCommand ▪ ReplayManager ▪ Speaker

Class Name: StartRec	
Responsibilities: ▪ Command that starts the recording of ReplayManager	**Collaborations:** ▪ ReplayManager

Class Name: StopRec	
Responsibilities: ▪ Command that stops the recording of ReplayManager.	**Collaborations:** ▪ ReplayManager

Input package

Class Name: DocumentReader	
Responsibilities: ▪ An interface for classes that are reading files.	**Collaborations:**

Class Name: DocumentReaderFactory	
Responsibilities: ▪ Creates the correct type of DocumentReader according to the given file type. ▪ Creates the correct Decorators to use above the DocumentReader according to the given decryption type.	**Collaborations:** ▪ DocumentReader ▪ WordReader ▪ ExcelReader ▪ TextReader ▪ ReaderDecoratorAtbash ▪ ReaderDecoratorRot13

Class Name: ReaderDecorator	
Responsibilities: ▪ Abstract class for decorators that decrypt the text a DocumentReader has extracted from a file.	**Collaborations:** ▪ DocumentReader

Class Name: ReaderDecoratorAtbash	
Responsibilities: ▪ Decorator that implements the atbash decryption on the text extracted from a file.	**Collaborations:** ▪ ReaderDecorator

Class Name: ReaderDecoratorRot13	
Responsibilities: ▪ Decorator that implements the rot13 decryption on the text extracted from a file.	**Collaborations:** ▪ ReaderDecorator

Class Name: WordReader	
Responsibilities: ▪ DocumentReader that is used to read files with the extension doc or docx.	**Collaborations:** ▪ DocumentReader

Class Name: TextReader	
Responsibilities: ▪ DocumentReader that is used to read files with an unknown extension. ▪ Will work on simple text formats and ignore more complicated files.	**Collaborations:** ▪ DocumentReader

Class Name: ExcelReader	
Responsibilities: ▪ DocumentReader that is used to read files with the extensions xls or xlsx.	**Collaborations:** ▪ DocumentReader

Model Package

Class Name: Document	
Responsibilities: ▪ Stores the text contents of currently opened file. ▪ Is used for any operations involving text.	**Collaborations:** ▪ TTSFacade ▪ DocumentReaderFactory ▪ DocumentReader ▪ DocumentWriterFactory ▪ DocumentWriter

Class Name: Speaker	
Responsibilities: ▪ Is responsible for preventing any Threads from running together and messing up TTS.	**Collaborations:**

Class Name: TTSFacade	
Responsibilities: ▪ A façade that is used by the rest of the classes to use the TTS library.	**Collaborations:**

Output Package

Class Name: DocumentWriter	
Responsibilities: ▪ An interface for classes that are writing files.	**Collaborations:**

Class Name: DocumentWriterFactory	
Responsibilities: ▪ Creates the correct type of DocumentWriter according to the given file type. ▪ Creates the correct Decorator to use above the DocumentWriter according to the given encryption type.	**Collaborations:** ▪ DocumentWriter ▪ WordWriter ▪ ExcelWriter ▪ TextWriter ▪ WriterDecoratorAtbash ▪ WriterDecoratorRot13

Class Name: WordWriter	
Responsibilities: ▪ DocumentWriter that is used to write files with the extension doc or docx.	**Collaborations:** ▪ DocumentWriter

Class Name: ExcelWriter	
Responsibilities: ▪ DocumentWriter that is used to write files with the extension xls or xlsx.	**Collaborations:** ▪ DocumentWriter

Class Name: TextWriter	
Responsibilities: ▪ DocumentWriter that is used to write files with an unknown extension. ▪ Will work on simple text formats and ignore more complicated files.	**Collaborations:** ▪ DocumentWriter

Class Name: WriterDecorator	
Responsibilities: ▪ Abstract class for decorators that encrypt the text a DocumentWriter will write into a file.	**Collaborations:** ▪ DocumentWriter

Class Name: WriterDecoratorAtbash	
Responsibilities: ▪ Decorator that implements the atbash encryption on the text that will be written into a file.	**Collaborations:** ▪ WriterDecorator

Class Name: WriterDecoratorRot13	
Responsibilities: ▪ Decorator that implements the rot13 encryption on the text that will be written into a file.	**Collaborations:** ▪ WriterDecorator

View Package

Class Name: EncodingController	
Responsibilities: ▪ Controller for the window opened where the user selects encodings. ▪ Passes selected encryptions to SimpleFileChooser.	**Collaborations:** ▪ SimpleFileChooser

Class Name: Main	
Responsibilities: ▪ Contains the main method. ▪ Loads the fxml and starts the program.	**Collaborations:** ▪ MainController ▪ Speaker

Class Name: MainController	
Responsibilities: ▪ Controller for the main program window. ▪ Adds usability to all the items in the menu.	**Collaborations:** ▪ CommandsFactory ▪ Document

Class Name: PreferencesController	
Responsibilities: ▪ Controller for the window opened where the user selects values for volume, pitch and word rate of the TTS. ▪ Sets the values of the TTS after the user selects them.	**Collaborations:** ▪ TTSFacade

Class Name: SimpleFileChooser	
Responsibilities: ▪ Contains the methods for opening the FileChooser either for opening or saving a file. ▪ Also creates a window where the user selects encryption methods.	**Collaborations:** ▪ Document ▪ EncodingController