Testing Document

CA400 Final Year Project

Project Title:	OCR Text to Speech Reading Aid
Student 1:	Róisín O’Rourke - 19360491
Student 2:	David Weir - 19433086
Supervisor:	Paul Clarke
Date Completed:	06/05/2023

Table of Content

1. Introduction

1.1 Scope

The features tested are crucial to functionality of the application and are necessary for the application to run properly and efficiently. These include:

- The image OCR functionality.
- The efficiency and accuracy of the translation.
- The language detection component.
- The conversion of the text into audio recordings.
- The summarisation functionality.

1.2 Quality Objective

The objective of the testing is to ensure that the core functionality of the OCR text-to-speech aid all operate correctly and to a high standard. To guarantee that the application will provide a positive user experience, we must ensure that it is free from bugs and works as it is intended.

2. Testing Strategies

2.1 Ad hoc testing

2.1.1 Ad hoc testing definition

Ad hoc testing is testing completed without formal documentation or planning. It is unstructured and involves randomly testing different functionalities [1].

2.1.2 Ad hoc testing use

In this application, ad hoc testing was used to locate any bugs or defects while writing the code for the application. It was used during the beginning stages of creating the application to ensure that the functionalities of the app were working as intended to, and at the end to check no new bugs had arisen.

2.2 Unit testing

2.2.1 Unit testing definition

Unit testing involves breaking down the code into the smallest possible units of code. These components are then tested in isolation from the rest of the system. It is used to verify the correctness of each unit of code [2].

2.2.2 Unit testing use

Unit testing was used to ensure that the individual functions of the system work properly and return the expected results. It was used in this application to verify that the text recognition, translation, audio file creation and more worked as intended.

2.3 Integration testing

2.3.1 Integration testing definition

Integration testing involves testing different components of an application together. It is used to identify errors that may occur when components are used together. It is a step above unit testing [3].

2.3.2 Integration testing use

In this project, integration testing was used to test how different components work together. It is used to test how language detection and translation work together and also how the audio creation and split text functionality work together.

2.4 Performance testing

2.4.1 Performance testing definition

Performance testing examines how well a system or component of a system performs. It can be used to identify any performance issues such as speed or reliability so that the system can be improved upon and operate at a more satisfactory level [4].

2.4.2 Performance testing use

The performance of the application is key to providing a positive user experience. The performance testing completed involved running the same piece of code ten times and then getting an average result for how long it took to run the code. It was used in this project to analyse the translation times and the audio file creation times.

2.5 Continuous Integration

2.5.1 Continuous Integration definition

Continuous integration involves automating the integration of any changes that have been made to the system. When changes have been made, automated tests run to ensure that the additional code will not affect the code already in the repository [5].

2.5.2 Continuous Integration use

For our application, CI/CD was connected to all tests that had been written. When code was pushed to GitLab, the tests were automatically run.

3. Test Cases

3.1 Translation

3.1.1 Test the Accuracy of French Translation Each Way

Test case name:	test_translation_eachway_fr
Pre-conditions:	There exists a text file with English text.
Steps taken:	● The text file is translated into French. ● The generated French text is translated back to English. ● Assert that the normalised edit distance is >= 0.8
Expected Result:	The original English text and the generated English text will have a normalised edit distance of greater than or equal to 0.8.

Actual Result:	The normalised edit distance is >= 0.8.
Test Results:	Pass.

3.1.2 Test the Accuracy of German Translation Each Way

Test case name:	test_translation_eachway_de
Pre-conditions:	There exists a text file with English text.
Steps taken:	● The text file is translated into German. ● The generated German text is translated back to English. ● Assert that the normalised edit distance is >= 0.8
Expected Result:	The original English text and the generated English text will have a normalised edit distance of >= 0.8.
Actual Result:	The normalised edit distance is >= 0.8.
Test Results:	Pass.

3.1.3 Test the Accuracy of French Translation

Test case name:	test_translation_en_fr
Pre-conditions:	There exists a text file of English text and a text file with the same text in French.
Steps taken:	● The English file is translated into French using the translate function. ● Assert the generated French and the reference French text have a normalised edit distance of >= 0.8.
Expected Result:	The generated French text and the reference French text should have a normalised edit distance of >= 0.8.
Actual Result:	The generated French text and the reference French text has a

	normalised edit distance of >= 0.8.
Test Results:	Pass.

3.1.4 Test the Accuracy of German Translation

Test case name:	test_translation_en_de
Pre-conditions:	There exists a text file of English text and a text file with the same text in German.
Steps taken:	● The English file is translated into German using the translate function. ● Assert the generated German and the reference German text have a normalised edit distance of >= 0.8.
Expected Result:	The generated French text and the reference French text should have a normalised edit distance of >= 0.8.
Actual Result:	The generated French text and the reference French text has a normalised edit distance of >= 0.8.
Test Results:	Pass.

3.1.5 Test the Performance of the Translation

Test case name:	test_translation_time
Pre-conditions:	There exists a text file with English text.
Steps taken:	● Create an array to store the times taken. ● Translate the English file. ● Add the time taken to complete the function to the array. ● Repeat 10 times. ● Calculate the average time taken to complete the function. ● Assert the average time is less than 15 seconds.

Expected Result:	The average time taken to create the translation is <= 15 seconds.
Actual Result:	The average time taken is <= 15 seconds.
Test Results:	Pass.

3.2 Detection

3.2.1 Test the Accuracy of English Detection from a file

Test case name:	test_en_detection
Pre-conditions:	There exists a text file with English text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in English.
Expected Result:	The function detects that the text in the file is English.
Actual Result:	The detected language is English.
Test Results:	Pass.

3.2.2 Test the Accuracy of English Detection on a Short file

Test case name:	test_en_detection_short
Pre-conditions:	There exists a text file with a short English text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in English.
Expected Result:	The function detects that the text in the file is English.
Actual Result:	The detected language is English.

Test Results:	Pass.

3.2.3 Test the Accuracy of French Detection from a file

Test case name:	test_fr_detection
Pre-conditions:	There exists a text file with French text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in French.
Expected Result:	The function detects that the text in the file is French.
Actual Result:	The detected language is French.
Test Results:	Pass.

3.2.4 Test the Accuracy of French Detection on a Short file

Test case name:	test_fr_detection_short
Pre-conditions:	There exists a text file with a short French text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in French.
Expected Result:	The function detects that the text in the file is French.
Actual Result:	The detected language is French.
Test Results:	Pass.

3.2.5 Test the Accuracy of German Detection from a file

Test case name:	test_de_detection

Pre-conditions:	There exists a text file with German text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in German.
Expected Result:	The function detects that the text in the file is German.
Actual Result:	The detected language is German.
Test Results:	Pass.

3.2.6 Test the Accuracy of German Detection on a Short file

Test case name:	test_de_detection_short
Pre-conditions:	There exists a text file with a short German text.
Steps taken:	● The file is passed to the detection function. ● Assert that the detected language is in German.
Expected Result:	The function detects that the text in the file is German.
Actual Result:	The detected language is German.
Test Results:	Pass.

3.3.2 Test that Detection will not work on Empty file

Test case name:	test_detection_empty
Pre-conditions:	There exists an empty text file.
Steps taken:	● The text file is passed to the detection function. ● Assert the function returns False.
Expected Result:	The detection function will not detected a language and returns False

Actual Result:	The function returns False as it has not detected the language.
Test Results:	Pass.

3.3 Audio file creation

3.3.1 Test if an Audio Recording has been Created

Test case name:	test_audio_created
Pre-conditions:	A text file that is not empty is present.
Steps taken:	● A text file is passed to the download_mp3 function, which creates the recording. ● The recording is passed through the mutagen MP3 module to retrieve the audio metadata. ● Assert there is a path to the new mp3 file. ● Assert the length of the mp3 file is > 0 seconds.
Expected Result:	The mp3 recording has been created and the length of the recording is greater than zero seconds
Actual Result:	There exists a new mp3 file and the length of the recording is greater than zero seconds.
Test Results:	Pass.

3.3.2 Test that an Empty Audio Recording has not been Created

Test case name:	test_empty_audio_created
Pre-conditions:	There exists an empty text file.
Steps taken:	● The text file is passed to the download_mp3 function. ● Assert that an AssertionError has been raised. ● Assert there is no path to a new mp3 file.
Expected Result:	An mp3 recording has not been created, and an

	AssertionError is raised.
Actual Result:	An AssertionError is raised when the empty text is passed, and there does not exist a path to a new mp3 file.
Test Results:	Pass.

3.3.3 Test the Performance of Audio Recording Creation

Test case name:	test_audio_time
Pre-conditions:	A text file that is not empty
Steps taken:	● Create an array to store the times taken. ● Pass a text file to the download_mp3 function. ● Add the time taken to complete the function to the array. ● Repeat 10 times. ● Calculate the average time taken to complete the function. ● Assert the average time is <= 10 seconds.
Expected Result:	The average time taken to create an mp3 recording is less than 10 seconds.
Actual Result:	The average time taken is less than 10 seconds.
Test Results:	Pass.

3.4 Summarisation

3.4.1 Test the English text has been Summarised

Test case name:	test_summarisation_en
Pre-conditions:	There exists a text file with English text.
Steps taken:	● The text file is passed to the summarisation function. ● Assert the length of the summarised text is shorter

	than the original text.
Expected Result:	The summarised text will be shorter than the original text.
Actual Result:	The length of the summarised text is shorter than the original text.
Test Results:	Pass.

3.4.2 Test the German text has been Summarised

Test case name:	test_summarisation_de
Pre-conditions:	There exists a text file with German text.
Steps taken:	● The text file is passed to the summarisation function. ● Assert the length of the summarised text is shorter than the original text.
Expected Result:	The summarised text will be shorter than the original text.
Actual Result:	The length of the summarised text is shorter than the original text.
Test Results:	Pass.

3.4.3 Test the French text has been Summarised

Test case name:	test_summarisation_fr
Pre-conditions:	There exists a text file with French text.
Steps taken:	● The text file is passed to the summarisation function. ● Assert the length of the summarised text is shorter than the original text.
Expected Result:	The summarised text will be shorter than the original text.

Actual Result:	The length of the summarised text is shorter than the original text.
Test Results:	Pass.

3.5 Print OCR

3.5.1 Test the Accuracy of OCR on excerpt from Bram Stoker's Dracula

Test case name:	test_drac
Pre-conditions:	● There is a non-empty Dracula.txt file in the ocr_results folder ● There is a corresponding correct ground truth text file
Steps taken:	● The OCR and ground truth files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9
Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The OCR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.5.2 Test the Accuracy of OCR on excerpt from Grimms' Fairy Tales

Test case name:	test_grimms
Pre-conditions:	● There is a non-empty Grimms' Fairy Tales.txt file in the ocr_results folder ● There is a corresponding correct ground truth text file
Steps taken:	● The OCR and ground truth files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9

Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The OCR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.5.3 Test the Accuracy of OCR on excerpt from The Great Gatsby

Test case name:	test_gatsby
Pre-conditions:	● There is a non-empty Great Gatsby text file in the ocr_results folder ● There is a corresponding correct ground truth text file
Steps taken:	● The OCR and ground truth files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9
Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The OCR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.5.4 Test the Accuracy of OCR on excerpt from Romeo and Juliet

Test case name:	test_randj
Pre-conditions:	● There is a non-empty Romeo and Juliet text file in the ocr_results folder ● There is a corresponding correct ground truth text file
Steps taken:	● The OCR and ground truth files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9

Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The OCR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.5.5 Test the Accuracy of OCR on excerpt from the Simple Sabotage Field Manual

Test case name:	test_oss
Pre-conditions:	● There is a non-empty OSS handbook text file in the ocr_results folder ● There is a corresponding correct ground truth text file
Steps taken:	● The OCR and ground truth files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9
Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The OCR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.6 HTR

3.6.1 Test the Accuracy of HTR on a folder of IAM handwritten sentence images

Test case name:	test_htr

Pre-conditions:	● There is a folder of images from the IAM dataset to test ● There is a corresponding folder of correct ground truth texts
Steps taken:	● HTR is run on all images in the folder and result text files are created ● All HTR and ground truth text files are opened. ● Edit distance function is run to compare the files. ● Assert that the normalised edit distance is >= 0.9
Expected Result:	The files are compared and the edit distance is above or equal to 0.9
Actual Result:	The HTR and ground truth files have a normalised edit distance >= 0.9
Test Results:	Pass.

3.7 Detection and Translation Integration

3.7.1 Test Successful Detection and Translate

Test case name:	test_detect_translate
Pre-conditions:	There exists a text file in English and a text file with the same text in another language.
Steps taken:	● Detect the language of the second text file. ● Translate the text into English and write it into a new text file. ● Detect the language of the translated file. ● Assert the language of the second file. ● Assert the language of the translated file. ● Assert the normalised edit distance between the generated English file and the reference English file is >= 0.85.
Expected Result:	The languages will be detected correctly and the file will be translated correctly.

Actual Result:	The second language is French, the translated language is English, and the normalised error distance is >= 0.85.
Test Results:	Pass.

3.7.2 Test Unsupported Language

Test case name:	test_unsupported_lang
Pre-conditions:	There exists a text file written in a language that is not supported by the application.
Steps taken:	● Pass the text file to the detection function. ● Translate the text file into English. ● Assert that the text is in Italian. ● Assert that a ValueError is raised when an unsupported language is passed to the translation function.
Expected Result:	The text file will be in Italian and the translate function will raise a ValueError when Italian is passed to it as the source language.
Actual Result:	The detected language of the text file is Italian and the translate function raises a ValueError.
Test Results:	Pass.

3.8 Audio and Split Text Integration

3.8.1 Test the mp3 Creation and Split Text Functionality

Test case name:	test_split_audios
Pre-conditions:	There exists a text file with text in it.
Steps taken:	● Create a temporary folder. ● Split the text file into smaller text files and add to the folder. ● Convert each text file into an mp3 recording.

	● Assert that the text files are 100 words long ● Assert that the mp3 recordings are between 35 and 50 seconds long.
Expected Result:	Each split text file will be 100 words in length and the mp3 recordings will all have a length of 35 to 50 seconds.
Actual Result:	Each split text file is 100 words in length and the mp3 recordings all have a length of 35 to 50 seconds.
Test Results:	Pass.

4. References

[1] *Adhoc testing* (no date) *Tutorials Point*. Available at: https://www.tutorialspoint.com/software_testing_dictionary/adhoc_testing.htm (Accessed: May 6, 2023).

[2] Hamilton, T. (2023) *Unit testing tutorial – what is, Types & Test example*, *Guru99*. Available at: https://www.guru99.com/unit-testing-guide.html (Accessed: May 6, 2023).

[3] *Integration testing - javatpoint* (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/integration-testing (Accessed: May 6, 2023).

[4] *Learn with tricentis: What is performance testing?* (2022) *Tricentis*. Available at: https://www.tricentis.com/learn/performance-testing (Accessed: May 6, 2023).

[5] Atlassian *What is continuous integration*, *Atlassian*. Available at: https://www.atlassian.com/continuous-delivery/continuous-integration (Accessed: May 6, 2023).