Software Testing Plan

Team name:
LangLens

Faculty Mentor
Italo Santos

Sponsor
Dr. Okim Kang

Team Members
Stefan Mihailovic, Daniel Navarette Martin
Brian Ruiz, Sami Tanquary
Kyle Young

Overview
The Software Testing Plan describes how LangLens intends to ensure that the expectations presented in the requirements and design specification documents are met.

Version 1.0

April 19, 2023

Contents

Contents **ii**

1 Introduction **1**

2 Unit Testing **3**
2.1 Text_Mode 4
2.2 Yolo_View 8

3 Integration Testing **10**
3.1 Camera Integration 10
3.2 Language Integration 11
3.3 Object Detection Integration 12
3.4 Text Recognition Integration 12
3.5 Learning Page Integration 13

4 Usability Testing **14**

5 Conclusion **16**

1

Introduction

LangLens is a language learning web application that utilizes augmented reality to help people learn languages by scanning objects and text around them. The big component of the application is the scanning, which is done on the user's end where they take a picture of the object or text they want to scan and then pick the object/text they want translated. From there the user is redirected to Collins Dictionary with the data that was collected from the scan; the data being the language the user selected, and the text or object that was scanned. The user also has the option to change the language of the application of their appropriate language.

The purpose of this document is to outline the software tests that we will conduct for our product. We want to test our application to see what it is capable of doing, as well as seeing where our application may break for us to know if it has any limitations and perhaps see if we can push past these limitations. Additionally, we want to make sure that it functions properly with the tests we give and that it gives the correct outcome from the tests. On the other hand, there may be certain small bugs that we are not aware of, which is why testing the application may flush out these bugs.

There are three types of tests we will conduct on our application: unit tests, integration tests, and usability tests. Firstly, the unit tests will be focused on certain parts of the application or components of code to ensure that the back end framework is processing data correctly by testing valid and invalid requests that are coming from the user. Secondly, the integration tests will be to see if the actions that we want the application to perform are running smoothly such as: testing the scanning of objects and text, returning the accurate results of scanning, and making sure the learning page provides the accurate outcome. Lastly, the usability tests will be more focused on the front end

and make sure that a user is capable of using the application with little to no trouble by having people use the product.

For the most part these tests are more geared towards the scanning of objects/text, and producing the correct translations, since these are the two main components of the application. We want to make sure that it scans correctly, returns the correct translation of object or text, and makes sure that there are no problems with using the scanner. On the other hand, the translations need to be accurate, otherwise the user will be receiving incorrect translations. The next sections of this document will go more in depth on the three types of tests for our product individually.

2

Unit Testing

Unit testing is a software testing technique that focuses on testing individual components or units of code independent from an entire system. In most cases, these units would be considered functions, methods, or subroutines and are typically simple in design in order to get the most accurate and efficient outcomes. The goal of unit testing is to ensure that each unit of code is working as intended and meets the specified requirements which is crucial for improving the quality of software, detecting bugs and reducing the time and cost of fixing future defects.

The following section will discuss the two critical components of the web application, Text Recognition Mode and Object Detection Mode, and their related unit tests. The LangLens web application uses Django, a Python-based web framework, which utilizes Python's standard unittest module. Within the unittest module, is the subclass TestCase which allows the team to create and run each test inside an isolated transaction. Additionally, the subclass Client will be used to act as a simulated web browser allowing the team to test the functionality of the views, templates, HTTP requests / responses, and URL mapping for both of the web app components in this section.

The team is specifically focused on testing the main functionality of the web application's server-side components or Apps: Text Recognition Mode (OCR) and Object Detection Mode (Yolov5). Due to the fact that the Client testing class can be used to simulate frontend actions, as explained above, the primary subset for unit testing is the Views functions for each detection mode component that handles the processing of the user's requests and responses as these are the most critical parts of the backend functionality.

For the OCR App, the following units will be tested: text_mode. This is the core view for Text Recognition Mode that is recognized by the Django backend to process client-side requests and responses. Within the unit is a 2 sets of unit tests specifically designed to target individual processes in both the request and responses, as well as the OCR processes and translations.

For the Yolov5 App, the following units will be tested: yolo_view. This is the main and only one view that handles the Object Detection Mode. We will test the responses of the view, as well as to ensure that the data is being handled correctly, even though there may be missing data or incorrect one.

2.1 | Text_Mode

Text_Mode is the main method in the OCR App. This method is a Django View function and handles a POST request containing image data and the approved language code information. This view function is responsible for performing all the OCR image processing on the inputted image data using Pytesseract (a Python-wrapper for Google's Tesseract OCR) and then translates the detected text from English into whichever approved target language is selected by the user. The correct response from this function is a JsonResponse containing a query dictionary with the url to the stored image, the target language information, and the coordinates of the bounding boxes in order to create clickable links for the user to be redirected to CollinsDictionary.com for more information on a detected word.

The following test cases for the Text_Mode unit were designed based on the following equivalence partitions and boundary values to test the request and responses of the view:

- **Equivalence Partitions:**
  - **Valid Image Data vs. Invalid Image Data:**
    - Valid Image Data is an image/png data url which is base64 encoded.
    - Invalid Image Data includes the other variety of image types (i.e. JPEG, PDF, GIF)
  - **POST Request vs. GET Request**
  - **Present AJAX Header vs. No AJAX Header:**

    - The AJAX Header should contain the type, url, header types, data img, language, dataType, and success flag.
  - **Valid Language Data vs. Invalid Language Data:**
    - Valid Language Data includes any of the approved target languages for our system (i.e. 'language' : 'korean', 'language' : 'spanish', 'language' : 'french')
    - Invalid Language Data includes any language not approved as a target language for our system (i.e. 'language' : 'german', 'language' : 'arabic')

- **Boundary Values:**
  - **Min Input:** Empty image payload
  - **Max Input:** Maximum size image payload (2.5MB)

Unit Test	**Description**	**Sample Input**	**Expected Outcome**
Valid POST request with image data, language, and AJAX header	Test a valid POST request made from user containing correct base64 encoded png image, language data, and AJAX header	{'img': 'data:image/png; based64,ivBORw 0KGg...==', 'language': 'spanish'}	JSON response with status code 200 containing image URL, language, and coordinates of detected text
Valid POST request with invalid image data, language, and AJAX header	Test a valid POST request made from user containing incorrect image data, language data, and AJAX header	{'img': 'data:image/jpeg', 'language': 'spanish' } {'img':' ', 'language': 'korean' } {'language': 'korean' } image larger than 2.5MB	JSON response with status code 500 containing an invalid image error
Invalid GET request with image data, language, and AJAX header	Test a invalid GET request made from user containing image data, language data, and AJAX header	{'img': 'data:image/png; based64,ivBORw0 KGg...==', 'language': 'spanish' }	JSON response with status code 405 containing an invalid request method error
POST request with no AJAX header	Test a POST request made from user containing image data, language data, but no AJAX header	$.ajax({ type: "POST", url: "text_mode", headers: { }, data: { img: url, language: language.value}, dataType: "json".....	JSON response with status code 405 containing an invalid request method error

The next set of test cases for the Text_Mode unit were designed based on the following equivalence partitions and boundary values to test the OCR / translation processes of the view:

- **Equivalence Partitions:**
  - **Valid Image:** a PNG image with clear text or no text
  - **Invalid Image:** a JPEG, GIF or any other data type
  - **Valid Language:** a 2 character string representing a valid language code approved for our system (i.e.'en' = English, 'es' = Spanish, 'fr' = French, 'ko' = Korean)
  - **Invalid Language:** a string representing an invalid language code (i.e. 'arab')
- **Boundary Values:**
  - **Min Input:** An image with no text or a single character
  - **Max Input:** An image with more than 100 words

Unit Test	**Description**	**Sample Input**	**Expected Outcome**
OCR Image with Text	Tests for valid OCR of an image with English text	pytesseract. image_to_string (testImg.png) PNG image with one word, 'Restaurant' PNG image with 50 words	Passed assertion test with correct OCR of the text in the image
OCR Image with No Text	Tests for valid OCR of an image with no Text	pytesseract. image_to_string (testImg.png) PNG image with no text	Passed assertion test with zero words detected
OCR invalid image type	Tests for invalid image type input	pytesseract. image_to_string (testImg.jpg) JPEG image with text GIF image with text	Error response with status code 415 for unsupported image type
Translate OCR text from English to valid target language	Test for valid translation of text obtained from pytesseract from English to valid target language	translator.translate (ocr_text, dest='es').text PNG image with one word and language code 'es' PNG image with 100 words and language code 'fr'	Passed assertion test with all words correct detected and translated to the target language
Translate OCR image from English to invalid target language	Test for invalid translation of text obtained from pytesseract from English to invalid target language	translator.translate (ocr_text, dest='foo').text PNG image with one word and language code 'foo' PNG image with 100 words and language code 'arab'	Error response with status code 400 for invalid language code specified
Translate OCR image with no text from English to valid target language	Tests for translation of no text	translator.translate (" ", dest='es').text PNG image with no text	Passed assertion test with zero words detected

2.2 | Yolo_View

Yolo_View is the main function for the Object Detection Mode. When the user accesses the Object Detection mode, the view will render the yolov5 template, which will apply for camera access, and once accepted, will display the camera using the whole size of the screen. The user can select a desired language, and take a picture of their environment, pressing the take picture button, which will generate an Ajax POST request, sending the image and the selected language in the request. The view will receive the request, checking if it is an Ajax request, and will retrieve the values, process them, and send back a JsonResponse with the url of the resulting image, the selected language and the coordinates of the scanned objects.

The following test cases for the Yolo_view unit were designed based on the following equivalence partitions and boundary values to test the request and responses of the view:

- **Equivalence Partitions:**
  - **Valid Image Data vs. Invalid Image Data:**
    - Valid Image Data is an image/png data url which is base64 encoded.
    - Invalid Image Data includes the other variety of image types (i.e. JPEG, PDF, GIF)
  - **POST Request vs. GET Request**
  - **Present AJAX Header vs. No AJAX Header:**
    - The AJAX Header should contain the type, url, header types, data img, language, dataType, and success flag.
  - **Valid Language Data vs. Invalid Language Data:**
    - Valid Language Data includes any of the approved target languages for our system (i.e. 'language' : 'korean', 'language' : 'spanish', 'language' : 'french')
    - Invalid Language Data includes any language not approved as a target language for our system (i.e. 'language' : 'german', 'language' : 'arabic')
- **Boundary Values:**
  - **Min Input:** Empty image payload

– **Max Input:** Maximum size image payload (10MB)

Unit Test	**Description**	**Sample Input**	**Expected Outcome**
Valid POST request with image data, language, and AJAX header	Test a valid POST request made from user containing correct base64 encoded png image, language data, and AJAX header	{'img': 'data:image/png; based64,ivBORw 0KGg. . . ==', 'language': 'spanish'}	JSON response with status code 200 containing image URL, language, and coordinates of detected text
Valid POST request with invalid image data, language, and AJAX header	Test a valid POST request made from user containing incorrect image data, language data, and AJAX header	{'img': 'data:image/jpeg', 'language': 'spanish' } {'img':' ', 'language': 'korean' } {'language': 'korean' } image larger than 2.5MB	JSON response with status code 500 containing an invalid image error
Invalid GET request with image data, language, and AJAX header	Test a invalid GET request made from user containing image data, language data, and AJAX header	{'img': 'data:image/png; based64,ivBORw0 KGg. . . ==', 'language': 'spanish' }	JSON response with status code 405 containing an invalid request method error
POST request with no AJAX header	Test a POST request made from user containing image data, language data, but no AJAX header	$.ajax({ type: "POST", url: "text_mode", headers: { }, data: { img: url, language: language.value}, dataType: "json".....	JSON response with status code 405 containing an invalid request method error

The tests for the YoloV5 view and Text Mode are the same since both receive the same requests, and return the same response

3

Integration Testing

Integration testing is a type of software testing that focuses on testing the interactions between different modules or components of a software system. The goal of integration testing is to ensure that these different parts of the overall system work together correctly when combined. Additionally, it is also used to detect bugs and defects in the interfaces and interactions between these different components which ultimately saves extra time and money put into reducing these potential defects.

Our team is focused on connecting the two main modules, being Object Detection and Text Recognition, to the other parts of the application. These parts of the application must use the data that has been received from the object and text recognition portion to be functional. The camera, the language and the learning page all interact with the object detection mode and text recognition mode to make LangLens the full package. The test harnesses for the integration testing will compare the data of the results of the integration tests to the expected outcomes. The following integration tests will show the interactions between each part of the application and will be necessary to the success of the product.

3.1 | Camera Integration

Users on the application will be able to access the camera to be able to make use of the object detection and text recognition portions of the application. This part of the application will allow for immersive language learning through a visual element, being the camera. The activation of the camera is the start of the entire process that LangLens provides.

Integration Test	Description	Expected Outcome
Camera Permission Accept	Tests for ability for user to give the application authorization to use their device's camera	When the user gives permission for the camera to be used, the user will be able to see from the camera's point of view
Camera Permission Deny	Tests for ability for user to deny the application authorization to use their device's camera	When the user denies permission for the camera to be used, there should not be any image
Take Picture	Tests for the ability for the user to take a picture with the camera using the picture button	When the user pushes the picture button, it should show the user the photo they have taken, and then send that to their respective modes
Disable Camera	Tests for the ability to turn off the camera	When the user disables the camera, they should no longer see from the camera's point of view

3.2 | Language Integration

User's on the application will be able to change the language to fit not only what language they know, but also what language they want to know. The language portion change the results of the object detection, text recognition and the learning page, so accurate results will be the utmost importance

Integration Test	Description	Expected Outcome
Change base language	Tests for the ability to change the language of the instruction text	The text of the instructions for the homepage, object detection and text recognition are correctly translated into the desired language
Change Translation Language	Tests for the ability to change what language the Object Detection and character recognition translate	The object detection and text recognition correctly translate to the desired language

3.3 | Object Detection Integration

The object detection module is able to detect objects within a photo from a camera feed. The expected outcome is that a bounding box with an accurate label is drawn around the object. Verification that the app can successfully detect the object is crucial so that the main functionality of the application can take place, as well as communicating with the backend and other modules.

Integration Test	Description	Expected Outcome
Receive photo	Tests for the ability to receive the captured photo from the camera feed	The photo displayed to the user is the same as the photo that the camera had taken
Detect Object	Tests for the ability to detect the object in the camera feed and return accurate results	A bounding box is drawn around the detected object. An accurate label containing the name of the detected object is shown within the bounding box.

3.4 | Text Recognition Integration

The Text Recognition Mode is able to recognize characters within a photo from a camera feed. The expected outcome is that a bounding box with an accurate label is drawn around the character or characters. Verifying that the application can successfully recognize a character or characters is integral to the main functionality of the application.

Integration Test	Description	Expected Outcome
Receive photo	Tests for the ability to receive the captured photo from the camera feed	The photo displayed to the user is the same as the photo that the camera had taken
Recognize Character or Characters	Tests for the ability to recognize the character or characters in the camera feed and return accurate results	A bounding box is drawn around the recognized characters. An accurate label containing the name of the recognized characters is shown within the bounding box

3.5 | Learning Page Integration

The learning page integration is the last step within the LangLens web application, as after either the detected object or recognized word has its bounding box drawn, interacting with the bounding box should take the user to the external Collins-Dictionary learning page.

Integration Test	Description	Expected Outcome
User access learning page	Tests for the ability to interact with the bounding box of detected object or recognized word for each respective mode	The user is taken to a Collins-dictionary page of the accurate translation according to their own language and their desired translation language

4

Usability Testing

Usability testing is the final testing strategy in our plan, and it involves incorporating the end-users into the testing process. The main purpose of usability testing is to ensure that the software is intuitive and easy to use for the end-users. Due to the nature of usability testing, it will not involve code based test cases, and instead will focus on acquiring feedback on the user interface, as well as the workflow of the system.

As LangLens is an application that is focused on entry level language learning, a simple and easy to use interface with a modern design is crucial to the success of the application. We need to be able to get feedback from not only our client, but also general users of our application, so we can take a step back and really understand the usability of our product. Not only do we need our application to be usable, but we also need our application to be aesthetically pleasing. While aesthetics are technically subjective, good aesthetics are clearly easier to evaluate than bad aesthetics. The following usability tests will be necessary to the success of the design of the product:

- **Main Menu Usability:** Evaluate how easily users can access the text recognition and object detection modes from the main menu, how easily users can change the language of the site, as well as evaluating how aesthetically pleasing the main menu's user interface is.
- **Text Recognition Mode Usability:** Evaluate how easily users can open text recognition mode and use the app to scan and recognize text.
- **Object Detection Mode Usability:** Evaluate how easily users can open object detection mode and use the app to scan and detect text.

- **Learning Page Redirect Usability:** Evaluate how easily users can understand to open the redirect to the Collins-dictionary page.
- **User Feedback:** Gather feedback from users about their experience using the app, including any suggestions or issues they encountered, as well as their preference to the aesthetics of the site.
- **Overall Usability:** Evaluate the overall ease of use and user-friendliness of the app, including its interface design and workflow.

To conduct our usability testing, we will be having participants ranging from friends and family, to roommates and classmates. The testing process will be structured in a linear way, going down the list as provided above. The participants will be sat down with one of the LangLens team members, and will be given the link to the web application. The present LangLens team member will work with the participant to walk through the usability tests, starting from Main Menu, to Learning page test. Finally, the LangLens team members will gather feedback from the participants concerning their overall experience using the app, as well as any suggestions or particular issues they encountered. Finally, the overall ease of use and user-friendliness of the app will be evaluated, including its interface design and workflow, and the results will, in addition to integration and unit testing, provide a more clear picture regarding the quality of our web application.

As mentioned earlier, usability testing will be the final software testing conducted, allowing us to make any final adjustments necessary, before our web application is handed over to our client.

5

Conclusion

To conclude, as we mentioned before we want our tests to make sure that our product is working efficiently and properly. The unit testing will help us understand what it can and cannot process by having the tests give valid and invalid inputs. The actions of the product such as the scanning, translating, and retrieving the learning page will be tested with the integration tests that we have for it. Finally, by having people who are unaware of our product use the product and provide feedback to us we will be able to gauge the usability of our product. The purpose of these tests is to help us understand the limitations of our product, therefore these tests will serve that purpose and will help us make any fixes that must be done for the product. That way we can ensure that LangLens performs as the expected language learning tool that we have designed it to be.