Final “As-Built” Report

Team name:
LangLens

Faculty Mentor
Italo Santos

Sponsor
Dr. Okim Kang

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

Version 1.0

May 8, 2023

Contents

1

Introduction

One of the most crucial aspects of humans is language. It is an essential tool we have that we use to make connections through verbal and written communication, which helps people from different backgrounds to be able to understand and connect with others. Additionally, when it comes to speaking different languages one must first be able to master a new foreign language in order to transcend through the language barrier. At first, it would take someone years to learn and master a new language by practicing, memorizing, and writing endless lists of vocabulary by taking classes or going through textbooks. Now, technology has helped us to the point where anyone can have access to learning any new language they want. One way is by creating language learning tools that help people learn new languages by focusing on the key elements of language learning being: meaning, use, and form. This tool is what our team, LangLens, has created.

Team LangLens has created a language learning tool that focuses on the key elements of language learning as well as the crucial audible and visual elements. Our product is also capable of utilizing augmented reality in the forms of scanning text and detecting objects, therefore we have named it EducationalAR. Our client Dr.Okim Kang, who is a professor in the department of english at Northern Arizona University, wanted us to create such a tool because she noticed the lack of tools that use augmented reality for language learning purposes. With that being said, our product does this by having two modes: one for scanning text and another for detecting objects, which are then translated to their appropriate language selected by the user. So far, the languages our product can translate to are English, Spanish, French, Korean, and Chinese. Now, our client has also noticed that the applications that do use augmented reality only fo-

cus on scanning text or objects, but rarely use both, and if they do they are usually locked behind expensive paywalls or subscriptions. As a result, EducationalAR is a free product that is easy to use, as we provide instructions on how to properly scan objects and text. Our product also redirects learners to Collins Dictionary where they are given a definition of the word, as well as how to pronounce it and use it in a sentence. EducationalAR is meant to be a free and easy-to-use language learning tool to be used by learners of all levels to be able to learn new languages through the use of augmented reality.

2

Process Overview

Our team didn't follow a traditional scrum like approach for our development life-cycle. Since the project was not given to us with an already established codebase, we had to start from scratch. Due to this, a big challenge we faced was figuring out what needed to be done, and in what order. As a result, upon figuring out our goals and requirements, we would estimate a timeframe needed to complete them, and work through them in that order.

The tools used to develop our project included:

- GitHub for version control and issue tracking.
- PyCharm as our IDE of choice.
- Roboflow for training the object detection models.

Our team had five primary roles with specific duties assigned to each team member:

- Team Lead - duties include delegating tasks and making sure everyone is staying on track
- Recorder - in charge of writing down meeting minutes
- Architect - responsible for designing the overall structure and functionality of our software system

- Customer communicator - served as the bridge between both our client, and end users, mostly for the purposes of testing the software, and the rest of the development team
- Release manager - making sure that our code and deliverables are up to standard and delivered on time. Was also in charge of merging pull requests.

Our primary channel of communication was a Discord server dedicated to our Capstone project.

3

Requirements

TThis section will summarize the project's requirement acquisition process and the obtained functional, non-functional, and environmental requirements.

In summary, while we did not necessarily follow a traditional SCRUM development process from the start, we did base our requirement acquisition process on manageable 2-4 week sprints. This process meant each requirement was reviewed based on our MVP goals, rated on technological feasibility, refined to a single sprint, given acceptance criteria for testing, and then assigned to a team member with a time estimation and deadline. We also met with our client, Dr. Okim Kang, at the beginning of the project to discuss all of her desired features and high-level requirements where she ultimately gave us the final say on design choices.

From these meetings and the above requirement acquisition process, we obtained the following **Key Project Requirements**:

- The web-app will be freely accessible from any mobile device with internet access and a camera.
- The web-app will be packaged in a simple, easy to understand, user interface capable of quickly and securely scanning objects or retrieving text from an image.
- The web-app will offer the ability to choose a target language for translation from English to Spanish, French, English and/or Korean.
- The web-app will be capable of toggling between two detection modes: Object Detection and Text Recognition.

- The web-app will offer the ability to restart the scanning process.
- The web-app will scan objects and texts in a live viewing mode for real time detection.
- The web-app will display clickable links after each scan to an external learning page that provides the correct definition, word-in-use guide, and pronunciation example.

The requirements above have been divided into the following categories:

- Functional Requirements.
- Non-Functional Requirements.
- Environmental Requirements.

3.1 | Functional Requirements

3.1.1 | User Actions

The User Actions consist of all the specific user functions associated with the UIs. Our target users for this system include anyone who would like to learn a new language including, but not limited to, young people (children), monolingual individuals, and people who struggle with text. Users will have multiple utilities accessible to them when interacting with the UIs that serve as the core functions for the application. With that said, users must be able to execute the following actions:

- Allow or deny the application permission to access their mobile device's camera
- Select a target language for translation offered in Spanish, French, English, or Korean via a target language selection toggle and drop down options of the currently available languages
- Toggle between two detection modes: Object Mode and Text Mode
  - Object Mode: the user can scan objects in their environment and see the translation in their chosen target language.

  - Text Mode: the user can scan words in their environment and see the translation in their chosen target language.

- Click on any of detected objects or words and be redirected to an external learning page.
  - The learning page must provide the correct definition, sentence usage, and audible pronunciation of the detected object / word.
- Restart the scanning process as many times as desired by clicking a "Restart" button.

3.1.2 | Object Detection

Object detection is a crucial portion of EducationalAR. Team LangLens is using YOLOv5, a Python-based object detection algorithm that utilizes a single deep neural network to perform real-time object detection with trainable models. Object detection functions must adhere to the following functional requirements:

- Detect objects from a picture taken by the user of their current environment.
- Show the user's desired scanned objects in a visual bounding boxes.
- Display the objects' translated labels above the detect objects that matches the language selected from the target language selector.

3.1.3 | Text Recognition

Text recognition is another important portion of what makes EducationalAR distinguished from other augmented reality language learning applications, as few of these existing AR applications have both text recognition and object detection within the same system. Text recognition will be powered by Google's Tesseract OCR engine, specifically the Python wrapped version, pyTesseract, to perform optical character recognition and extraction. Text recognition must adhere to the following functional requirements.

- Detect text from a picture taken by the user of their current environment.
- Show the user's desired scanned words in visual bounding boxes.

- Display the words' translated labels above the detected words that matches the language selected from the target language selector.

3.1.4 | Translation

Translation will occur after the text or objects have been detected. The words or objects will be translated from the user's base language of English to their selected target language.

- Display the translated label of detected objects to the chosen target language.
- Display the translated label of detected text to the chosen target language.

3.1.5 | External Learning Page

After any successful scan, the bounding boxes around the detected objects or words will be clickable links that redirect the user to an external learning page (Dr. Okim requested Collins Dictionary). The contents of the learning page will be provided by Collins Dictionary and must contain the correct definition, word-in-use, and pronunciation for any of the detected objects or words in the result image. The point of the learning page is for the user to be able to immerse into the key elements of language learning being meaning, usage, and form. The learning page must adhere to the following functional requirements:

- Is displayed by clicking on any detected object's or word's bounding box.
  - Bounding boxes are externally linked to Collins Dictionary.
- Provides the correct definition, sentence usage, and pronunciation for any requested object or word.

3.1.6 | Frontend

The frontend of the web-app must have a user interface for the homepage, Object Mode, and Text Mode, all designed and developed using Bootstrap. Users will interact with these UIs to perform the following tasks:

- **Homepage UI**

  - Two buttons for toggling between the detection modes: Object Mode and Text Mode.

- **Object / Text Mode UIs**
  - Requests camera access from the user displayed as a permission prompt.
  - A toggle for target language selection.
    * Displays the preliminary options in a dropdown menu of Spanish, French, English and Korean.
  - Displays clickable bounding boxes around any detected objects / words that redirects to the external learning page after each scan.
  - A restart button at the bottom of the UIs that allows the user to restart the scanning process.

3.1.7 | Camera Actions

Requirements are needed from the camera to access our application to create a seamless language learning experience. The camera must do the following:

- Use the camera to detect object.
  - Using a live-video stream, the user can take a picture which is used as input for the object detection engine (YOLOv5).
- Use the camera to detect words.
  - Using a live-video stream, the user can take a picture which is used as input for the text recognition engine (Tesseract OCR).

3.2 | Non-Functional Requirements

While in the section above we outlined the functional requirements our application must follow, in this section we will discuss how the application will be expected to perform.

3.2.1 | Speed

In terms of speed, object detection and text recognition must be relatively fast when it comes to detection and recognition. The application should not take more than 5 seconds to successfully detect and scan an object / word. The result image displayed after each scan falls into this 5 second window and should not take more than 2 seconds. The backend must also be highly responsive as it is responsible for storing and retrieving image data, and therefore, it should not take more than 1-2 seconds to accomplish either. Lastly, when the user clicks on any of the detected objects' or words' bounding boxes, the application must redirect the user to the Collins Dictionary learning page in no more than 3 seconds.

3.2.2 | Accuracy

Both object detection and text recognition accuracy should not fall below 65%. This is a large buffer, but necessary as we are using free detection software and is prone to experience at least one error in every 6 out 10 detections. When it comes to translation, since the YOLOv5 object models have to be trained in each available language offered on the web-app, the accuracy of the translation of the objects in our custom dataset should be no lower than 90% accurate. This is because there is no API being used to translate them during each scan and instead Google Translate is used to manually translate each object in our custom dataset during the model training. Therefore, the buffer of 90% is given as Google Translate cannot be fully relied on for 100% linguistic accuracy in any use case. The translation for Text Mode is being handled by Google-Trans API and in this case is being done during each scan. Therefore, accuracy of the translation is given a larger room for error at a minimum of 65% since each translation is live and relies on the accuracy of the OCR engine which can vary case by case when using the application. Finally, the search query sent to Collins Dictionary when the user clicks on any of the bounding boxes should be 95% accurate with a 5% buffer for the occasional detection errors that our chosen free engines are prone to make.

3.2.3 | Usability

The frontend of the application must provide an intuitive and easy to use interface. Users should not need any training to use and navigate the application, but there is a "Guide" button on both detection mode UIs that provides an explanation on how to properly use either of the modes for further clarification.

3.3 | Environmental Requirements

The following section will describe the environment requirements related to the constraints imposed upon the application by the client and the chosen solution software and hardware.

3.3.1 | Web-Based Mobile Application

As requested by our Dr. Okim, EducationalAR will be a web-based mobile application and will be accessible via any type of web browser (i.e. Google Chrome, Mozilla Firefox, Safari, etc.) on any type of mobile operating system (i.e. Apple iOS, Google Android, Microsoft's Windows Phone OS, etc.). Due to the nature of the application, the mobile device that is accessing the EducationalAR web-app will need to have a stable internet connection in order to use the application.

3.3.2 | Free Software

EducationalAR is required to utilize only freely available software and tools and will not utilize any paid services or platforms for development. Dr. Okim has specifically requested that EducationalAR must be free-to-use by any user and should not require the use of a developer's license for distribution of the application via an app store.

3.3.3 | Mobile Device with Camera

A hardware requirement that was not explicitly stated by the client but is implied by the functionality of the application is that EducationalAR will require the use of a mobile device camera. This means that any user accessing the web-app must have a mobile device with access to a camera or if they are accessing the website from a PC or laptop, they must have a webcam.

4

Architecture and Implementation

The following section will provide a high-level overview of the EducationalAR web-app architecture, as well as insight into the responsibilities of its three main components, their data control flows, architectural styles and illustrations of their typical use cases.

EducationalAR follows a three-tier, client-server architecture using Bootstrap for the frontend, which uses HTML, CSS, and JavaScript, and Django for the backend, which uses Python and HTML. The system consists of three major components: Main UI, Object Detection App and Text Recognition App. The Object Detection App and Text Recognition App components communicate directly with the server's static files to store and retrieve image data from the user's live video stream.

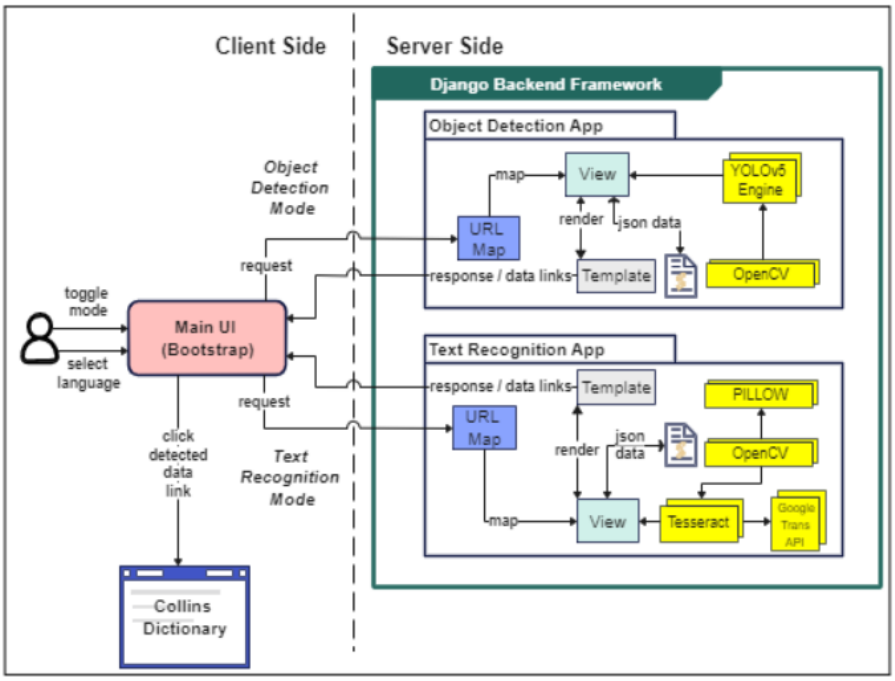


Figure 4.1: System Architecture.

4.1 | Main UI

- **Key Responsibilities:** The Main UI, or "homepage", serves as the client-side interface when accessing the EducationalAR web-app. It houses the two main user-operated features, Object Mode and Text Mode, which are the core functions for the entire web application. The user is also able to change the entire web-app's native language if English is not their primary language.

- **Control Flow:** The Main UI is deployed through Django, a fully-featured server-side Python web framework that allows for the use of both Python and HTML to create web-based applications. The Main UI serves as a pipeline between the user's HTTP requests and the Django server by relaying these requests via URL mapping to the targeted app's View file. The View file interacts with the targeted app's Python programs and renders an HTTP response as an HTML Template file which is received and dispatched to the Main UI.

- **Architectural Style:** The overall influence for the Main UI's architecture is based on its involvement with the system's overall three-tier application architecture. In

a modular client-server architecture, there are three layers: a presentation layer, an application layer, and a data layer. The Main UI serves as the presentation layer for this style as it is the user interface for the web application.

Figure 4.2: Main UI

4.2 | Object Detection App

- **Key Responsibilities:** The Object Detection App is one of the two major detection mode modules used within the Django backend framework that contains the View, Template, and URL files that execute the Object Mode. When the user toggles to Object Detection Mode, the App package is executed on the server-side and dispatched to the user as a separate UI page designed specifically for object detection mode. Once rendered, the user is able to start the scanning process, choose a target language for translation, click on an object's bounding box for redirection to the learning page, restart, or stop the scanning process completely and return to the Main UI.

- **Control Flow:** Each App package created for Django requests, handles, and responds the same way. First, an HTTP request from the user using the Object Detection Mode URL is received by the Django URL Mapper. The mapper iden-

tifies the request for this mode and redirects it to the appropriate View file . The user then chooses a target language, aims their camera at their desired object and hits the Detect button. This sends an AJAX POST request from the Template to the View containing the base64 encoded image of the object and their target language. The View preprocesses the image with OpenCV and sends it to YoloV5 to identify the object with one of our custom trained object models (depending on the target language selected). The YOLOv5 algorithm then returns the detected object's bounding box coordinates, confidence score, and result image as a JSON response. The response data is rendered back to the Template after the frontend JavaScript transforms the bounding boxes around the object in the result image as a clickable link to the external learning page (Collins Dictionary).

- **Architectural Style:** The architectural style for the Object Detection App is component-based to allow for the decomposition of the package into logical components. This is the most efficient and modular use of the components within the application as it opens up the possibility for new feature implementation and seamless collaboration.

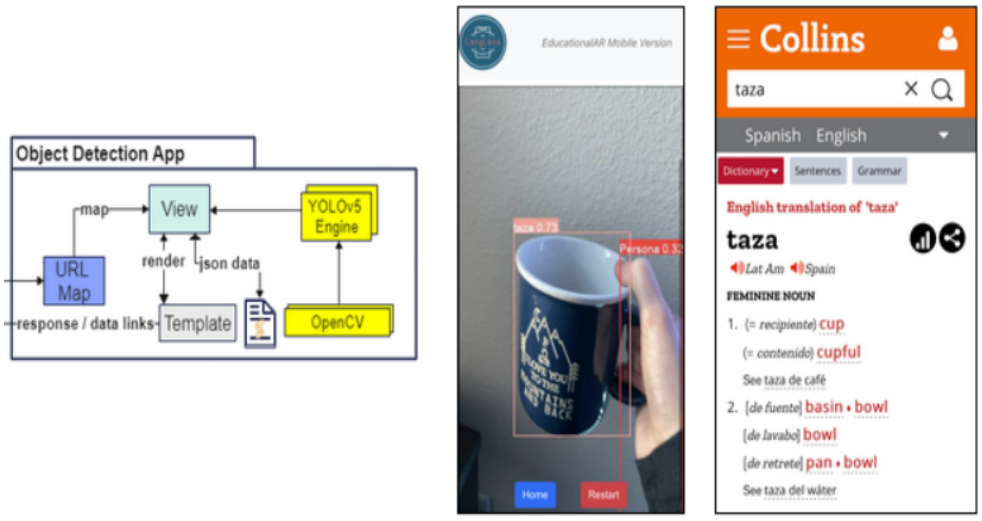


Figure 4.3: Object Detection App Component and Use Case Images

4.3 | Text Recognition App

- **Key Responsibilities:** The Text Recognition App, similar to the Object Detection App, is the Python App package used within the Django backend framework that contains the Model, View, Template, and URL files that execute Text Mode. The Text Recognition tasks are very similar to Object Detection, but instead uses Google's Tesseract OCR package, specifically the Python wrapped version, pyTesseract, to perform optical character recognition and extraction. Simply put, pyTesseract uses a pre-trained neural network of Leptonica's language-specific training data and a collection of machine learning algorithms to detect characters and recognize full words.
- **Control Flow:** As stated in section 4.2, each App package created for the Django backend requests, handles, and responds the same way. So very similarly to our Object Detection App, the user's image is again preprocessed with OpenCV, sent to pyTesseract for text recognition and extraction, and then returned as an array with any detected words in the image and their bounding box coordinates. The detected text is then translated to the user's target language with GoogleTrans API and drawn onto the original image with PILLOW using the obtained bounding box coordinates. From here, the control flow of this App package is the exact same and only differs in computational logic and the Template page response which would be the Text Recognition UI instead.
- **Architectural Style:** The architectural style is component-based, as previously described for section 4.2, for reusability and easy reconfiguration of logical components.

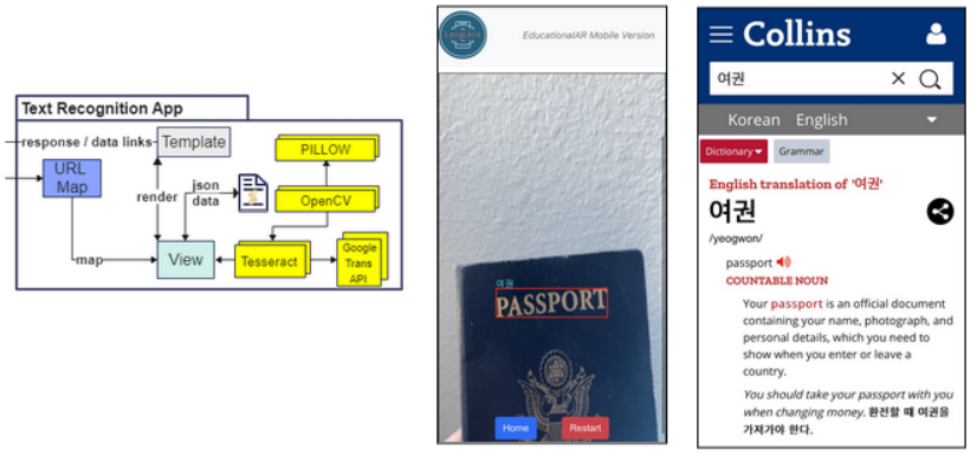


Figure 4.4: Text Recognition App Component and Use Case Images

5

Testing

In this section, we will discuss our overall testing strategies and results that validated our implementation and improved EducationalAR.

Testing is a crucial part when developing software, as it helps to make sure that everything works as expected and discover possible bugs. For our overall testing plan we conducted three kinds of tests:

- **Unit Tests:** A software testing technique that focuses on testing individual components or units of code independent from an entire system.
- **Integrations Tests:** A type of software testing that focuses on testing the interactions between different modules or components of a software system.
- **Usability Tests:** To ensure that the application is intuitive, easy to use, and aesthetically pleasing for end-users.

5.1 | Unit Tests

The following section will discuss the two critical components of the web application, Text Mode and Object Mode, their related unit tests and results. The EducationalAR 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 was 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.

Team LangLens was 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 was 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.

5.1.1 | Text Mode

Text Mode is the main method in the Text Recognition 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 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 to test the request and responses of the View function:

Unit Test	**Expected Outcome**	**Results**
Valid POST request with image data, language, and AJAX header	JSON response with status code 200 containing image URL, language, and coordinates of detected text	Pass
Valid POST request with invalid image data, language, and AJAX header	JSON response with status code 500 containing an invalid image error	Pass
Invalid GET request with image data, language, and AJAX header	JSON response with status code 405 containing an invalid request method error	Pass
POST request with no AJAX header	JSON response with status code 405 containing an invalid request method error	Pass

The next set of test cases for the Text Mode unit were designed to test the OCR engine and translation processes of the View function:

Unit Test	**Expected Outcome**	**Results**
OCR Image with Text	Correct OCR of the text in the image	Pass
OCR Image with No Text	Zero words detected	Pass
OCR invalid image type	Error response with status code 415 for unsupported image type	Pass
Translate OCR text from English to valid target language	All words correctly detected and translated to the target language	Pass
Translate OCR image from English to invalid target language	Error response with status code 400 for invalid language code specified	Pass
Translate OCR image with no text from English to valid target language	Passed assertion test with zero words detected	Pass

5.1.2 | Yolo_View

Yolo_View is the main function for the Object Detection Mode. When the user accesses Object Mode, the View will render the yolov5.html template, which will request camera access, and once accepted, displays the camera using the whole size of the device's screen. The user can select a target language and take a picture of their environment, pressing the Detect button, which will generate an AJAX POST request that sends the image and the selected language to the View. The View will receive the request, checking if it is an AJAX request, and will retrieve the values, process them, and return them as a JSON response 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 to test the request and responses of the View function:

Unit Test	Expected Outcome	Results
Valid POST request with image data, language, and AJAX header	JSON response with status code 200 containing image URL, language, and coordinates of detected objects	Pass
Valid Language Selection	HTTP response with 200 status code with 'yolov5.html' template correctly rendered containing the correct language as context	Pass
Invalid Language Selection	HTTP response with 404 status code containing a message warning that the application does not support the language	Pass

Although it was not explicitly tested in these units, we did find through these tests that the response times for these two main View functions could have been faster. To reduce the round-trip time (RTT), we condensed the View functions to have fewer image processing steps when using OpenCV which brought the RTT down to less than 5 seconds on average.

5.2 | Integration Tests

The following section will discuss the interactions between the critical components of the web application, their related integration tests, and results.

Team LangLens was focused on connecting the two main modules, Object Mode and Text Mode, to the frontend 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 Object Mode and Text Mode to make EducationalAR a complete application. The test harnesses for integration testing were to compare the data of the results of the integration tests to the expected outcomes. The following integration tests show the interactions between each of these components and were necessary for the success of the product.

5.2.1 | Integration Tests

Users on the application will be able to access the camera 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 EducationalAR provides and without it the application would be rendered functionless.

Integration Test	Expected Outcome	Results
Camera Permission Accept	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.	Pass
Camera Permission Deny	When the user denies permission for the camera to be used, there should not be any image.	Pass
Take Picture	When the user pushes the Detect button, it should show the user the photo they have taken, and then send that to their respective modes.	Pass
Disable Camera	When the user disables the camera, they should no longer see from the camera's point of view.	Pass

5.2.2 | 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 wish to learn. The language selection changes the results of the object detection, text recognition and the learning page, so accurate results are of the utmost importance.

Integration Test	Expected Outcome	Results
Change Native lLanguage	The text of the instructions for the homepage are correctly translated into the desired native language.	Pass
Change Target Language	The object detection and text recognition correctly translate to the desired target language (Spanish, French, English, Korean, or Chinese).	Pass

5.2.3 | Object Mode Integration

The Object Mode is able to detect objects within a photo from a live camera feed. The expected outcome is that a bounding box with an accurate translated 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	**Expected Outcome**	**Results**
Receive photo	The photo displayed to the user is the same as the photo that the camera had taken.	Pass
Detect Object	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.	Pass

5.2.4 | Text Mode Integration

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

Integration Test	**Expected Outcome**	**Results**
Receive User Photo	The photo displayed to the user is the same as the photo that the camera had taken.	Pass
Recognize Full Words and Correctly Translate	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.	Pass

5.2.5 | External Learning Page Integration

The external learning page integration is the last step within the EductionalAR web application, as after either the detected object or recognized word has its bounding box drawn, interacting with the bounding box should redirect the user to the external Collins-Dictionary learning page with the correct definition, sentence usage, and pronunciation.

Integration Test	Expected Outcome	Results
User access learning page	The user is taken to a Collins-dictionary page of the accurate translation according to their own language and their desired translation language.	Pass

It is worth noting that during the integration testing for Text Mode and Object Mode we found a broken pipe error if either of the detection engines didn't successfully detect an object/word. This would result in the user having to refresh the entire web-app to use either of the modes again which was not ideal. To fix this bug, we implemented a fail catch for both modes that ensures that if either of the detection results contain empty query dictionaries, then the user is shown a "Failed to Detect " page. From this failure page, the user is shown examples of how to get a successful scan and then the option to either restart the scanning process or return to the homepage all without causing a broken pipe error on the server side.

5.3 | Usability Tests

In this final testing section, we will discuss the usability testing of EducationalAR and its results.

As EducationalAR is a web 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 needed to be able to get feedback from not only our client, but also general users of our application, in order to really understand the overall 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 were conducted in a random selection of 10 users (friends, family and students at NAU) to assess the overall usability and aesthetics of EducationalAR on a scale of 1-5 with 1 being unusable to 5 being completely usable:

- **Main Menu Usability:** Evaluated how easily users can access the text recognition and object detection modes from the main menu, change the native language of the site, as well as how aesthetically pleasing the main menu's user interface was.
  - **Average Rating: 4.5**
- **Text Mode Usability:** Evaluated how easily users can open Text Mode, select a target language, detect text, and view the correctly translated labels.
  - **Average Rating: 4**
- **Object Mode Usability:** Evaluated how easily users can open Object Mode, select a target language, detect objects, and view the correctly translated labels.
  - **Average Rating: 4**
- **External Learning Page Redirect Usability:** Evaluated how easily users can click on an object / text bounding box, be redirected to Collins Dictionary, and whether or not the definition, sentence usage, and pronunciation were correct.
  - **Average Rating: 4**
- **User Feedback:** Gathered 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.
  - **General Feedback:**
    - A few users suggested we implement a loading spinner to visually show when an image is processing as they felt it was unclear if they need to retake a photo or simply wait for the result to appear.
    - One user suggested that we limited the detection to one object / word at a time to reduce the amount of bounding boxes. We had to explain that this was not feasible given our chosen technologies but added a line to

our guide explaining that to reduce the amount of boxes users should try to get as close to their desired object / word as possible.

- **Overall Usability:** Evaluated the overall ease of use and user-friendliness of the app, including its interface design and workflow.
  - **Average Rating: 4**

The testing process was structured in a linear way going down the list provided above. The participants were sat down with one of the LangLens team members and given the local link to the web application. The LangLens team member worked with the participant to walk through the ordered list of usability tests. The LangLens team members then gathered 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 was evaluated, including its interface design and workflow.

Overall everyone in the testing group agreed the application was intuitive and easy to use, but a few users suggested adding a loading spinner to visually show that an image is processing. They also had a general consensus that they wished the detection results were a little more accurate. We had to explain that because we are using free software for the detection engines that there is not a feasible way to achieve higher than 65

6

Project Timeline

In this section, we will review the project's major milestones for the entirety of the Capstone experience. The Fall 2022 semester was mainly focused on the research phase of the project including requirement gathering and technological feasibility. The Spring 2023 semester was heavily focused on the development, testing, and implementation phases of the project and their associated documentation.

6.1 | Fall 2022 Semester

Regarding the Fall semester schedule, we carried out our research phase, hitting five major milestones: Team Website, Technological Feasibility, Requirement Specification Document, Tech Demonstration, and Design Review I.

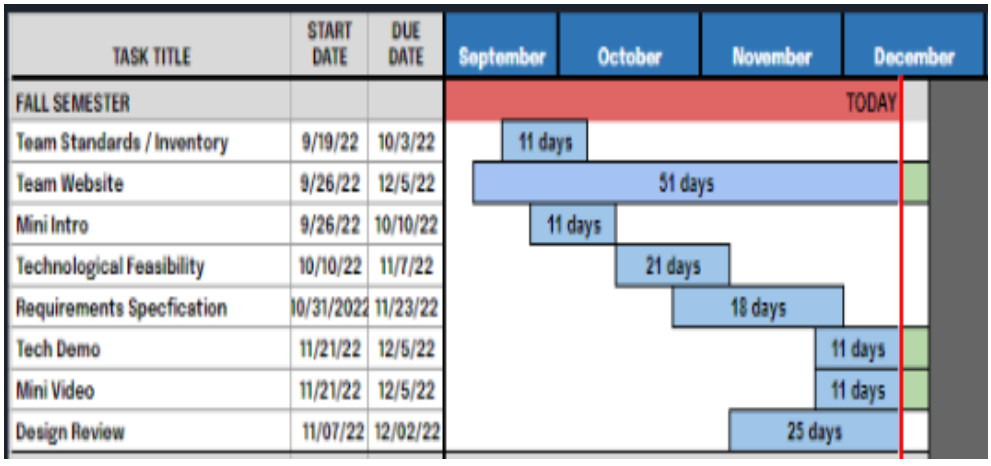


Figure 6.1: Fall 2022 Schedule

The Team Website was considered a major milestone that lasted throughout the entirety of the Fall 2022 semester. It was constantly updated and needed to be complete with all project info, requirements, documentation, envisioned architecture, mini video, and schedule by the end of the term. The Technological Feasibility document was the next major milestone and was primarily focused on research and proving the feasibility of the chosen technologies we were planning to use to develop EducationalAR. The Requirements Specification Document was arguably our most important milestone for the middle of the semester as it outlined and described in detail all the functional, non-functional, and environmental requirements that would be necessary to include in our final product. This document was edited at the end of the Fall semester after a meeting with our client who wanted to adjust some of the requirements originally asked of us regarding the learning page. The second to last milestone completed was the Tech Demonstration which was a live demonstration to our mentor and client that proved our chosen technologies were capable of being implemented into the development of the final product. The final major milestone was the Design Review I presentation that was held live at the FEST 1 Engineering Conference at NAU to explain our first look at the project's design choices and envisioned solution. It was the most important wrap up to the Fall semester and ensured that the team was ready to tackle implementation in the Spring.

6.2 | Spring 2023 Semester

The Spring 2023 semester was the beginning of development, testing, and implementation phases. It was primarily focused on following our system design plan and architecture to develop EducationalAR and be ready for final deployment at the conclusion of the Capstone experience in May.

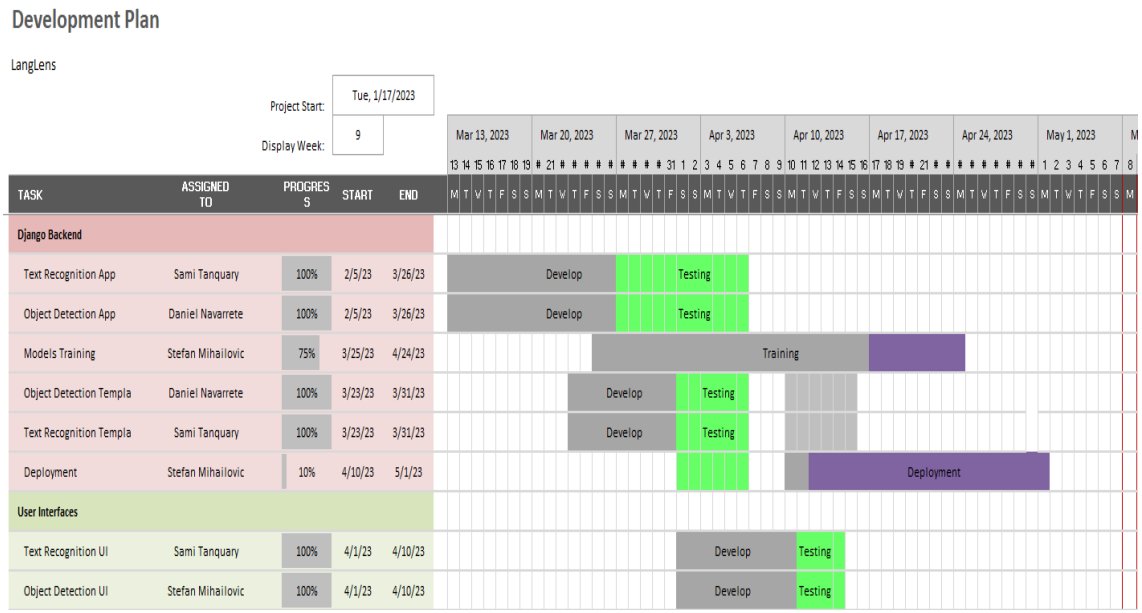


Figure 6.2: Spring 2023 Schedule

Following the Gaant chart above, we first set up our Django project configuration and started working on the implementation of the two main modules of the web application, Object Mode and Text mode. At the same time, we were designing and implementing the user interfaces and training the language models used in the YOLOv5 algorithm for Object Mode. Once everything was implemented and working, we could finally deploy the project on the server. After each major milestone was developed, we followed our specified testing plan previously discussed in section 5.

7

Future Work

In this section, we will discuss the future work that lies ahead for EducationalAR. While we were able to create a distinguished first version, EducationalAR is capable of much more than we had time to implement during the course of this year.

Once Team LangLens has graduated, future work on this project will be handled by Dr. Okim and her PhD candidate, Kevin Hirschi, as we have given them access to and ownership of the application's code base. While these are not all of the possibilities for the future of EducationalAR, these outline the next steps to take regarding the development of version 2.0.

7.1 | Internal Learning Page

Dr.Okim has future plans for integrating the learning page from Collins Dictionary internally as well as other learning resources so that the user is not redirected to a different website. This would involve setting up a Model in the Django backend that is capable of creating unique learning page templates that can be filled in with the detect object or word's definition, sentence usage, pronunciation and any other future learning resources that Dr. Okim desires.

7.2 | Personal User Accounts

Other future work on this project includes implementing personal user accounts with the ability to store words from detected objects and text for future language learning and practice. This was a stretch goal for our team this year, but unfortunately we

did not have enough time to implement this feature. These accounts would be set up in the backend using a Django database and would require additional security measures to ensure that user's personal information cannot be leaked or accessed in any way. With these accounts, an admin side would need to be developed in order to have access to edit, remove, or ban individual accounts if necessary as well as access the server's database. In addition to the database for the accounts themselves, a database for each detection mode would be needed as these personal accounts would allow users to save any scanned object or word into their account and review them in a flash card style game. To reduce the size of the amount of objects or words stored in the databases, modifying the process of detection of objects and words would need to include reducing the amount of detection boxes to 1, or configuring the saving of objects or words to the user's select choice. Basically this means users should not be able to save every single object or word in their photo as this would overflow the database or require a lot of additional storage that can get pretty costly over time with more users.

7.3 | Train Object Models

Additionally, more object models will be trained in order to increase object detection accuracy. Our team used RoboFlow to train each individual language model. This process included drawing the bounding boxes manually around 10-15 similar images and providing the correct label of the object in each language available in our application. Because of this, the amount of hours that went into training each model was too much to provide more than 300 objects for the Object Mode. As of now, there are currently no freely available object models online which means that future work for this portion would require a lot of additional person hours and is something that future developers of the project might consider hiring additional developers to handle completely.

7.4 | Add More Target Languages

The last step for the next stages of EducationalAR include adding a variety of other languages, such as Vietnamese, in order to further Dr. Okim's language learning research. While this process is incredibly easy for Text Mode since Tesseract OCR is pre-trained in over 100 languages, it is not that easy for Object Mode, as mentioned in section 7.3. To add more languages for Object Mode, each new language has to have its own custom trained object model. This means countless person hours for manually adding every single object you want the model to include, drawing bounding

boxes manually around every object in every provided photo, and including the correct translation of the object for each object training in the model. As you can probably imagine, this process is painstakingly time consuming and therefore, future work for adding more languages is not as trivial as it may seem at first glance and any future developers should consider hiring designated members to train these models around the clock.

8

Conclusion

Dr.Okim has observed a lack of free, web-based language learning applications that have the latest augmented reality features of both object detection and text recognition that focus on the key elements of language learning being meaning, usage and form. These applications do not offer practical visual and audible links between the written forms and meanings of words. In collaboration with Dr. Okim and Kevin Hirschi, Team LangLens have created an immersive and intuitive, mobile-first, language learning web application that focuses on the key elements of language learning and offers object detection and text recognition. Our team has been able to complete all of the minimum viable product requirements, including both object detection and text recognition as well as connection to the Collins Dictionary learning page. We have implemented the MVP target languages being Spanish, English, French, Korean, and an additional stretch goal language of Chinese into our text and object detection modes. With these distinguished features combined, we have truly created a polished final product. Dr.Okim is very happy with the work we have done throughout the course and we as a team are very proud of our project. We hope we can inspire many language learners to achieve their aspirations in an immersive and interactive way, as well as furthered Dr.Okim's research for many years to come.

9

Glossary

The following are terms referenced throughout the document and their definitions.

Term	Definition
AR	Augmented Reality is an interactive experience that combines the real world and computer-generated content.
SCRUM	A framework for project management that emphasizes teamwork, accountability and iterative progress toward a well-defined goal. The framework begins with a simple premise: Start with what can be seen or known. After that, track the progress and tweak, as necessary.
MVP	Minimum Viable Product is an early, basic version of a product (typically a computer program or piece of technology) that meets the minimum necessary requirements for use but can be adapted and improved in the future, especially after customer feedback.
OCR	Optical Character Recognition is the identification of printed characters using photoelectric devices and computer vision software.
UI	User Interface is the means by which the user and a computer system interact, in particular the use of input devices and software.

10

Appendix A: Development Environment and Toolchain

10.1 | Hardware

Throughout development of EducationalAR, our team used a variety of operating systems (Linux, Windows, and Mac) as our product is not dependent on any specific OS. The two minimum hardware requirements is to have at least 5gb of disk space on your device to ensure proper installation of all project packages, and a decent graphics card on your chosen device for the YOLOv5 engine to run and detect objects smoothly. Note that an upgraded or expensive graphics card is not necessary as we were still able to run YOLOv5 on laptops that did not have powerful graphics cards, but be aware that the speed of the CPU will be a bit slower than on PCs that have powerful gaming graphics cards.

10.2 | Toolchain

The following subsection will discuss the chosen software tools for the development of EducationalAR.

- **PyCharm:** For the duration of the project, our team used PyCharm, a Python IDE, to develop the web application. We chose PyCharm because it provides code analysis, a graphical debugger, an integrated unit tester, and supports web development with Django.

- **Django:** The full stack web framework we chose to work with for the overall project was Django, which is highly popular for responsive mobile web app development. Django uses Python and HTML for all server-side development which was perfect for the implementation of our Python based YOLOv5 and Tesseract OCR packages in our backend.

- **Bootstrap:** Bootstrap is a frontend web framework that is highly popular for responsive, mobile, web-app development. Bootstrap uses HTML, CSS, and JavaScript for all client-side development. Bootstrap is not necessarily required as Django is more than capable of developing frontends, but Bootstrap is specifically designed for creating mobile web applications and made developing our UIs much easier than if we just stuck with Django for the full-stack.

- **YOLOv5:** YOLOv5 is a Python based object detection algorithm that utilizes a convolutional neural network to identify objects using customizable and trainable object models. We chose YOLOv5 because it was free to use, based in Python, and was the most accurate object detection algorithm we could find that didn't require monthly payments for detections.

- **Google's Tesseract OCR:** Google's Tesseract OCR, or more specifically for our project, the Python wrapped version pyTesseract, is an optical character recognition engine that uses uses a pre-trained neural network of Leptonica's language-specific training data and a collection of machine learning algorithms to detect characters and recognize full words. Similarly to YOLOv5, we chose this package for the text recognition feature in our application as it was free, open-source, based in Python, and the most recommended to use in a free application while still maintaining a relatively high detection accuracy rate.

- **OpenCV:** In order to use YOLOv5 and pyTesseract with the highest achievable accuracy and speed, all images need to be preprocessed with color correction and orientation fixes which is why we implemented OpenCV: a computer vision library popular for image processing. OpenCV is free and open source and provided all of the necessary image processing tools needed for both detection engines and was the most compatible for our system as it is based in Python as well.

- **RoboFlow:** RoboFlow is a developer tool for creating computer vision models. When designing our project, we neglected to realize that we would need to train an object model for every offered language in the web-app. After discovering

that we couldn't just translate YOLOv5's provided object labels, we needed to find a model training tool that could help us create our own object models and add to our existing datasets. To use RoboFlow, you upload 10-15 images of your desired new object, manually draw the bounding box around the object to identify it in each photo, provide it with a label (in each language you want to offer in a separate model for each language), and then run the images through the detection algorithm. RoboFlow will then provide you with an object model package that you can download and easily just implement into the code base's main directory for YOLOv5 to use.

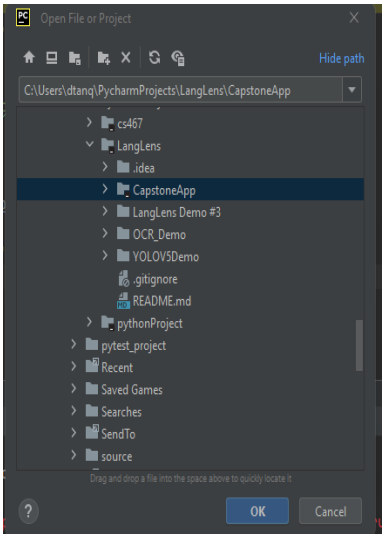
10.3 | Setup

This subsection will walkthrough how to install the project to your local device for future development.

To install EducationalAR on any device for future development or deployment on another platform, the first step is to ensure that you have a Python IDE installed on your local device. We recommend PyCharm (**download**) as it was the IDE we chose to develop the project with for this year.

Once you have PyCharm installed on your device, the next step is to download the .zip folder containing the actual source code for EducationalAR from the GitHub repository provided by Dr. Okim.

After the project code is fully downloaded on your device, you will need to unzip the project source code folder and open the folder named **"CapstoneApp"** within PyCharm.



NOTE: The folder name "CapstoneApp" should have a black terminal box symbol next to it indicating that it is a Python compatible project.

Once the project is opened in PyCharm, the next step is to open the terminal, which is located at the bottom of the IDE window, in order to create a Python virtual environment.

10.3.1 | Creating the Virtual Environment

1. Open the terminal in PyCharm and check if Python3 is installed. You should see *Python3 [current version number]* if your device has Python3 installed.
   Python3 –version
   NOTE: If you do NOT see a version number appear, you will need to install Python3 before moving onto the next steps. (`https://www.python.org/downloads/`)

2. Create the new virtual environment.
   python3 -m venv educationalAR-env

3. Activate the virtual environment.
   educationalAR-env\Scripts\activate.bat

4. Install Django package.
   pip install django

5. Install all the requirements in the virtual environment.
   python -m pip install -r requirements.txt
   NOTE: If any of the packages in the requirements do not successfully install, you can manually install them via the terminal using pip.
   pip install [package name]

10.3.2 | Running the server locally (for development)

1. Open the terminal and make sure you are inside the project's directory.
   . . . \LangLens\CapstoneApp
   NOTE: If you are not in the CapstoneApp directory, you will need to *cd* into the project directory wherever it is stored on your device using the terminal command line.

2. Run the ssl server in the terminal.
   python manage.py runsslserver <your ip address>:8000
   NOTE: To connect to the server from your mobile device, make sure you are connected to the same WiFi network as your laptop/PC and enter the following URL into your mobile phone's browser. It is important to make sure to use https and not http or else your phone will not have access to video elements used in EducationalAR.
   https://<your ip address>:8000/

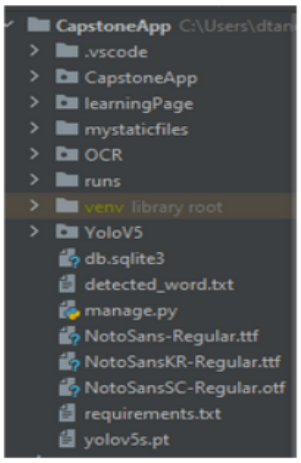
10.4 | Production Cycle

In this subsection, we will walkthrough how to edit, compile, and deploy EducationalAR for development and testing.

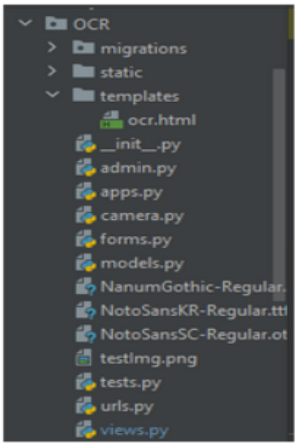
10.4.1 | Editing

1. Open the project in PyCharm. Make sure you are opening the directory name "CapstoneApp" and not any of the demo directories.

2. You should see the project's entire directory on the left hand side of the IDE.

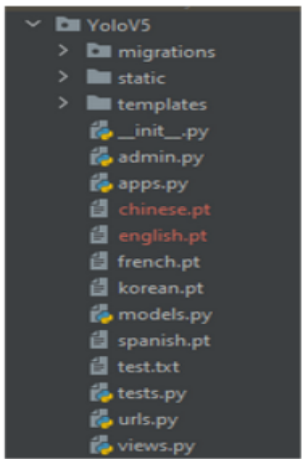


3. To make changes to the Text Mode feature of the application, you will want to enter the "OCR" directory. Inside, you will find all of the HTML, CSS, and JavaScript files under "Static" where you can make any frontend changes to the UI. The other most important file is the "views.py" file where you will find all of the code for image processing and OCR engine. This is the file that you will make all of your backend edits to when modifying the processes that run Text Mode.



4. To make changes to the Object Mode feature of the application, you will want to enter the "YoloV5" directory. Inside, you will find all of the HTML, CSS, and JavaScript files under "Static" where you can make any frontend changes to the UI. The other most important file is the "views.py" file where you will find all of the code for image processing and the YOLOv5 engine. This is the file that you

will make all of your backend edits to when modifying the processes that run Object Mode.



- Note: You will also see in this folder the object models that end in ".pt". These are the trained object models for each language offered in the application. To create you own, review the RoboFlow section in 10.2 or review RoboFlow's documentation (RoboFlow)

10.4.2 | Compiling

1. Once all your changes have been made, save the project. The next steps follow the end of Setup subsection in 10.3.

2. Open the terminal at the bottom of the PyCharm IDE and make sure you are in the current project's main directory.
   . . . \LangLens\CapstoneApp>

3. Run the SSL server in the terminal. It is imperative that you run the SSL server and not just "runserver" as the SSL certificate will allow you to access the local website on your mobile device for testing.
   python manage.py runsslserver <your ip address>:8000
   NOTE: To connect to the server from your mobile device, make sure you are connected to the same WiFi network as your laptop/PC and enter the following URL into your mobile phone's browser. It is important to make sure to use https and not http or else your phone will not have access to video elements used in Educa-

tionalAR.
https://<your ip address>:8000/

10.4.3 | Deploying

To deploy the project in the production environment, you will need the login access from Dr. Okim for the Hostwinds server. Once logged in, you will need to SSH into the server using an ssh client such as PuTTY. Inside the server, you will see the existing project files and you can just update the existing project files with any changes you made using a file transfer tool such as WinSCP.