The County of Los Angeles has 87 libraries, and they serve 10 million people county wide. They recently have started to offer a lot of services online. That means online learning services like Lynda.com, eBooks, and even movies. All of these services can be accessed online without having to leave the comfort of your own home. That said, you still need to physically walk down to the library and sign up for a physical library card before you can access all of these wonderful and free services. The County of Los Angeles Public Library came to us so that we can build an online portal where you can sign up yourself and start taking advantage of these services at any moment, from anywhere. We are looking to help automate the current registration process. My team and I have taken this as an opportunity to implement some cool new technologies to make signing up a fun and easy experience. We want you to be able to talk to the form and register without having to touch a keyboard at all. Along with that we thought it would be awesome if you could snap a pic of your driver's license and we'll handle everything else and get you going as quick as possible My team is having a lot of fun building and implementing these features. Here's what we did the past year. We had to create and host our own registration database. Unfortunately, we didn't have access to the library's main database to test and build our code on so we had to create our own database. Next the library wanted us to validate all user input. That means making sure the email address and the physical address are both valid inputs. Finally we decided to automate the sign up process to make it as accessible to people as possible.

The server we built and designed is an Apache server running on Ubuntu OS. Apache is a server software used industry wide for many applications. We coded our project in the Python3 language. This lets us use the Django framework. Django is a great framework that lets us connect and manage SQL databases.

First of all, the Library wants only people who live in the state of California to sign up for a library card, so initially, we implemented the Google Address Autocomplete API, where users only need to enter a part of their address before seeing a list of complete addresses. Once the user clicks on an address, the information of the selected address will be used to automatically fill in the related fields, such as the city and state. However, we realized that this API can only be used before submission, so we also implemented the USPS Web Tools API. This API will take the user's entered address and return the best-matching address, and this address will be used to compare with the entered address.

For email validation, once the form has been submitted, an unique id will be generated and it should be appended to the library's website's URL. This link will be sent to the user's entered email. Once the user clicks on the link, the corresponding flag in the database should be updated.

The library needs to accommodate for people who speak different languages, so we implemented Google Language Support on our form. The user only needs to click on a dropdown box and click on their primary language, and then the form should be properly translated. reCAPTCHA is a security feature that we implemented to keep bots away from signing up for a library card. This appears as a simple checkbox, but if the API cannot determine that the user is a human, simple puzzles will be continuously deployed.

Optical Character Recognition Module (OCR Module):

While thinking of ideas for modules that would be useful, we addressed the difficulty of using a smartphone to fill out online forms. It’s difficult for many people to select the text field they are aiming for. It’s also difficult for many people to type using their small smartphone keyboard. We wanted to make it easier for users to fill out an application with smartphone.

We created the OCR Module so that users can use their smartphone to take a picture of their California Drivers License or California ID and upload it to have their information automatically entered for them. In order to get information from a Driver's License to form fields, we needed to use Optical Character Recognition which extracts text from an image.

Many features of this module were successfully implemented.

- The user can upload images to fill out a library card application.
- If they are using a smartphone, their camera will automatically open prompting them to take a picture of their Drivers License. If they are using their desktop, they still have the option to use this module, but they will be uploading directly from their hard drive.
- This module uses Google Cloud Vision API for image processing and text detection. We tested many libraries available for OCR and we found that Google Cloud Vision API had the highest accuracy and was the best documented. We chose Google Cloud Vision API because it is necessary to have the users' information as correct as possible.
- There is a lot of irrelevant information on a Driver's License such as hair color or eye color and we need that filtered out. OCR Module takes all the text and determines where the first name, last name, address, birth date, and Drivers License number are all located in the text.
- The application is then filled out with the corresponding text for each field. The user's address is checked to make sure that it is a real address in California. Any fields where something went wrong are left blank and marked on the application for manual completion.

Voice Assist

At the beginning of the project we had our main objective of creating a form that was user-friendly, hassle-free, and simple. We wanted users to be able to get going with their library services quickly. To do this we needed to come up with all of the features our form would need to have to do this. We looked around to see what best practices were being used so we could implement them in our form. Once we finished this part of the project we started to come up with ideas that would make the form a fun and a more accessible experience. Voice Assist, one of the ideas, allowed us to give the users the ability to fill out the form via voice. This helps users who are typing on smartphone screens with small keyboards, or people who are visually-impaired.

The Voice Assist module is implemented by using Google's browser speech api that allows developers to translate speech to text, and vice-versa. First we tried using a wrapper library Artyom.js that brought a lot of useful functions that made writing Voice Assist a lot easier. However, we had issues with that library because it was very heavy for the things we needed. We ended up writing our own api based on the browser speech api that would let us listen for commands and talk to the user.

We could have written the Voice Assist module without the built-in browser api but that would have some challenges. We would have to turn speech-to-text using some other method and that would have to run on our own backend service. That means we would need to figure out how to build it in an efficient way that would allow mobile users to communicate with it while using the form. Since users would have to record audio snippets and upload it to our backend service that would lead to latency and it would be very hard to optimize.

By using the browser speech api built-in Chrome and Firefox we focused on the voice user experience. We built commands that allow users to fill out every field by saying the field of interest. We also built a mode that walks through the form and asks users for the input. We also built ways for users to go back in the form until they are satisfied with the result. The browser speech api allowed us to build a form that could talk to users and still be flexible.

OCR Challenges

High Resolution images

Modern cameras are starting to have a higher megapixel count which so the images taken from these cameras will result in a larger image file. If many users are uploading large image files at the same time, it will use a lot of the bandwidth of the servers which will result in a slowdown of the servers.

Image Quality

If the image is Blurry, Too Light/Dark, Noise, Low Resolution, the API will have a difficult time to identify the text on the image.

Watermarks and glare

Watermarks and light glares will also cause problems for the API because it can mask off some text on the image which will also cause problems for the API to pick up the text where the watermarks and glare are.

Voice Challenges

Web speech API

At first we used artyom on top of the google web speech API because it allowed us to create and use many of the built in functions, but after some testing we noticed that there was a few seconds

of lag that will throw the user off. We decided to only use Web Speech API because it was actually more accurate and the few seconds of lag was cut down to only a few milliseconds of lag.

Voice driven application

When writing the program we wanted the application to be free flowing. We wanted the user to be able to go to any field just by saying the field name and answering the question. We had different problems we had to overcome. One problem we had to deal with is that Google Chrome will only use the microphone for a few seconds before turning of the microphone. After we overcame this problem we were able to make the application free flowing with voice.

Browser compatibility

When creating this project we wanted to make the Voice application wort with all the browsers, but only Google Chrome and Firefox have the Web Speech API built into the browser. Google Chrome is the only browser that has the API enabled by default while Firefox has this API disabled. We do not think that it will be practical for a normal user to go into the settings for Firefox just to turn on the API just to sign up for a Library card, so we are only going to have Google Chrome support the voice application.

Performance

When testing the application locally there is no latency but when testing the application on the test server there is a slight delay on when the user can speak. There is a slight delay because we need to have a SSL configuration for the voice application to work on the server. Since the website has to keep telling Google that there is a SSL configuration if causes a small delay on when the user can reply to the question.