**Software Design**

**Document**

**for**

**Library Card**

**Registration System**

**Version 1.0**

**Prepared by Maurice Mejia, Alvin Tsui, Christopher Lee, Brent Hamada, Jose Aparicio**

**LA County Library**

**4/14/2018**

Table of Contents................................................................................................................... <pg 2>

Revision History..................................................................................................................... <pg 4>

1. Introduction................................................................................................................ <pg 5>
   1. Purpose........................................................................................................... <pg 5>
   2. Document Conventions……………………................................................. <pg 5>
   3. Intended Audience and Reading Suggestions................................................ <pg 5>
   4. System Overview........................................................................................... <pg 5>
2. Design Considerations........................................................................................................... <pg 6>
   1. Assumptions and dependencies..................................................................... <pg 6>
   2. General Constraints........................................................................................<pg 6>
   3. Goals and Guidelines.....................................................................................<pg 6>
   4. Development Methods................................................................................... <pg 6>
3. Architectural Strategies.......................................................................................................... <pg 7>
4. System Architecture............................................................................................................... <pg 7>
   1. Level 0 DFD.................................................................................................. <pg 7>
   2. Level 1 DFD.................................................................................................. <pg 8>
5. Policies and Tactics..............................................................................................................<pg 10>
   1. Specific Products Used................................................................................<pg 10>
   2. Requirements traceability............................................................................<pg 10>
   3. Testing the software.....................................................................................<pg 10>
   4. Engineering trade-offs.................................................................................<pg 10>
   5. Guidelines and conventions.........................................................................<pg 10>
   6. Protocols......................................................................................................<pg 10>
   7. Maintaining the software.............................................................................<pg 11>
   8. Interfaces......................................................................................................<pg 11>
      1. Structure…...................................................................................................<pg 12>
   9. System's deliverables...................................................................................<pg 13>
   10. Abstraction...................................................................................................<pg 13>
6. Detailed System Design.......................................................................................................<pg 14>

6.1 Gui……………...........................................................................................<pg 14>

6.1.1 Responsibilities....................................................................................<pg 14>

6.1.2 Constraints...........................................................................................<pg 14>

6.1.3 Composition.........................................................................................<pg 14>

6.1.4 Uses/Interactions..................................................................................<pg 14>

6.1.5 Resources.............................................................................................<pg 14>

6.1.6 Interface/Exports..................................................................................<pg 14>

6.2 Voice Assist...…..........................................................................................<pg 14>

6.2.1 Responsibilities....................................................................................<pg 14>

6.2.2 Constraints...........................................................................................<pg 14>

6.3 OCR..…………..........................................................................................<pg 14>

6.3.1 Responsibilities....................................................................................<pg 14>

6.3.2 Constraints...........................................................................................<pg 15>

6.3.3 Composition.........................................................................................<pg 15>

6.3.4 Uses/Interactions..................................................................................<pg 15>

1. 6.3.5 Resources.............................................................................................<pg 15>

6.4 Email Confirmation.....................................................................................<pg 15>

6.4.1 Responsibilities....................................................................................<pg 15>

6.4.2 Constraints............................................................................................<pg 15>

6.4.3 Composition.........................................................................................<pg 15>

6.4.4 Uses/Interactions..................................................................................<pg 15>

6.5 Address Lookup..........................................................................................<pg 15>

6.5.1 Responsibilities....................................................................................<pg 15>

6.5.2 Constraints............................................................................................<pg 15>

6.5.3 Composition.........................................................................................<pg 15>

6.5.4 Uses/Interactions..................................................................................<pg 15>

Detailed Lower Level Component Design

7.1 Speech.js......………......................................................................................<pg 16>

7.1.1 Classification........................................................................................<pg 16>

7.1.2 Design Class Hierarchy........................................................................<pg 16>

7.1.3 Restrictions/Limitations.......................................................................<pg 16>

7.1.4 Performance Issues................................................................................<pg 16>

7.1.5 Design Constraints................................................................................<pg 16>

7.2 OCR…….....………......................................................................................<pg 17>

7.2.1 Classification........................................................................................<pg 17>

7.2.2 Design Class Hierarchy........................................................................<pg 17>

7.2.3 Restrictions/Limitations.......................................................................<pg 17>

7.2.4 Performance Issues................................................................................<pg 17>

7.2.5 Design Constraints................................................................................<pg 17>

1. User Interface
   1. Overview of User Interface.........................................................................<pg 18>
   2. Screen Frameworks or Images....................................................................<pg 18>
   3. Database Design.........................................................................................<pg 19>
2. Requirements Validation and Verification............................................................................<pg 20>
3. Glossary................................................................................................................................<pg 23>
4. References............................................................................................................................<pg 23>

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Creation | 10/12/17 | Initial Changes on section 1 and section 10 | 0.1 |
|  | 10/22/17 | More requirements added to section | 0.1.1 |
|  | 12/1/17 | Final Draft Revisions | 0.5 |
|  | 12/7/17 | Update DFD’s | 0.5.1 |
| End of Fall Semester | 12/8/17 | Merged Submission and Accessibility module to GUI module and removed older DFD | 0.5.2 |
| Adjustment | 03/25/18 | Updated - changed technologies | 0.6 |
| Finished more Sections | 04/09/18 |  | 0.7 |
| Final | 4/14/18 | Final Draft | 1.0 |

**1. Introduction**

**1.1 Purpose**

The Library Card Registration System, which will now be referred to as LCRS, will allow any resident from California to register for a Los Angeles County Library Card online and immediately access online resources. The Los Angeles County Public Library will host the LCRS on their servers when it’s delivered to them.

**1.2 Document Conventions**

The organization of this document provides a natural “flow” or allocation of

requirements to each succeeding section. Details regarding the overall document

structure is discussed in section 2.

**1.3 Intended Audience and Reading Suggestions**

This documentation is intended for developers and testers. Developers can look through the documentation to understand the way the data will flow from client to database. Testers can look to this document to check if the LCRS is updating data in the correct manner. This document should be read from the following order:

1. Introduction
2. System Architecture
3. Detailed System Design
4. Detailed lower level Component Design
5. User Interface

By reading the document in that order, you will get an idea of the project in terms of its behaviors, the design, and how it should have been implemented.

**1.4 System Overview**

The LCRS will have these major modules.

1. GUI (Graphical User Interface)
2. OCR (Optical Character Recognition)
3. Voice Assist
4. Accessibility
5. Email Confirmation
6. Submission
7. Address Lookup

The GUI will provide a user-friendly application that the user will use to provide information. The OCR module will convert an image into text for the application. The Voice Assist feature will be a guide that will verbally provide instructions for the user while the user goes through the application. Accessibility will be implemented to ensure that a broader range of users can use this application. The Email Confirmation module will verify that a user’s email address exists. The Submission module will create a new entry into the Library’s database using the appropriate fields that were filled in by user input. The Address Lookup module will verify that the user’s inputted address is an address that exists in Los Angeles.

**2. Design Considerations**

**2.1 Assumptions and Dependencies**

The LCRS is coded in Python and the Django framework. This will allow the LCRS to be accessible on a multiple of devices, including, but not limited to, mobile devices, iPads, and Desktops. The LCRS will have to accommodate visually-impaired users. This means we will have to address screen reading devices and provide voice recognition for input data.

**2.2 General Constraints**

Constraints on the systems include browser support, along with accessibility support. LCRS will support all modern browsers. This includes Chrome, Firefox, and Internet Explorer 10. Currently Voice Assist will only be supported on Google Chrome. Besides supporting modern browsers, we also must make sure to make LCRS accessible by all. This means following and meeting the Web Content Accessibility Guidelines 2.0 specifically level AA. We must also keep in mind the end-user environment. LCRS will accommodate most popular devices through which the application might be accessed by. We will make sure to make the application accessible on all mainstream devices, and screen sizes.

**2.3 Goals and Guidelines**

The goal of LCRS is to make an application that is easy to use and simple at the same time. It should be able to assist with entering data. Our mandatory delivery date is yet to be discussed but it should be by May 2018.

**2.4 Development Methods**

As we work on this project we will be following the Agile development software design method. We will meet every week to describe our progress during the last work period. We will assign each other small tasks that should get completed week by week. We will be checking in with each other daily to update our teammates on our progress with that week’s tasks. This will allow us to work through and build the LCRS while focusing on quality the entire development cycle

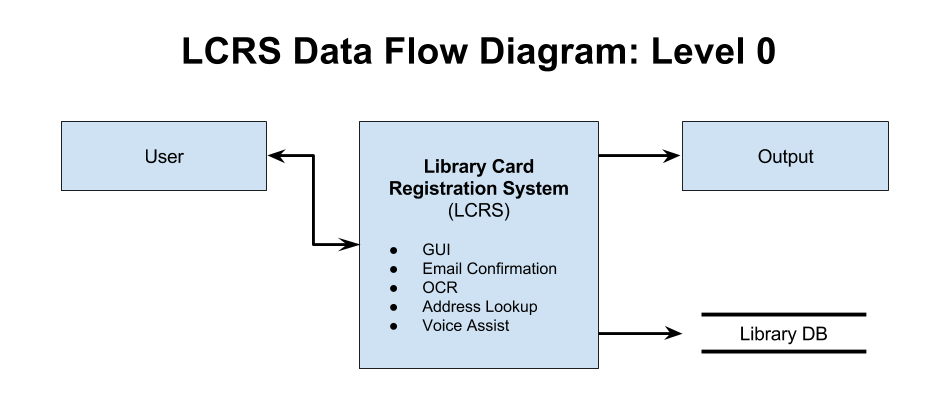
**3. Architectural Strategies**

Python and Django framework was used to build the LCRS because it allows a quality quick development. Python was used for writing clean code while Django was used for having the project be self-contained. Javascript was used for Voice Assist because it allowed us to have voice input on the client side which will put less stress on the server and mitigate latency of voice inputs.

**4. System Architecture**

The LCRS has been broken up into a few segments to allow us to create a more logical and concise application. To start, at the highest level (DFD 0), the application has been broken up into four pieces. The user will enter the LCRS and submitting the application will result in a thank you screen and a request to the library’s API.

**Level 0 DFD**



Next, we will discuss each module and how it will work together with the other modules. Below, you can see the level 1 DFD.

**Level 1 DFD**



The (LCRS) Control module will handle all the core logic. This module will be able to handle core logic such as routing different services for the user to fill the application. For example, upon selecting the Voice Assist option in the application, the user will bring up the Voice Assist service. Control will also be able to make the appropriate requests to submit an entry, have logic to verify during application entry that a user’s address is indeed a valid address, and have an option to automatically fill in certain fields via OCR (which Control will be able to provide access to).

The GUI module is the only way for a user to input data and submit the application. It will consist of the necessary fields to make a submission. It should be built so that it works on all screen sizes through responsive design.

The OCR module will be able to extract data from a scan or photo of the user’s California Driver’s License. This module will upon being called, request a picture from the user. Then it will be able to scan the relevant data, and autofill appropriate fields.

Voice Assist module will be able to take control of the application and guide the user through the required fields. It will use voice to ask the user for field data, and autofill the relevant field. It should be enabled through the GUI module via a button.

The Submission module has one purpose, to submit a valid complete application to the endpoint. It will return a message to Control that will detail the success of the entry, or failure.

The Email Confirmation module is in charge of verifying a user has a valid email address. An address will be deemed valid if a user has verified their email through a  link sent by the validation module.

The Accessibility module will allow screen readers and other accessibility services to use the web site.

The Address Lookup will assist in helping the user input their address. When filling out their address the user will have a drop down of valid suggested addresses from the Google Maps API that the user can choose from. The address is then sent to the USPS API to further confirm if the address has not been tampered and is a valid California address.

**5. Policies and Tactics**

**5.1 Choice of which specific products used**

There are a few libraries needed to create this application. Starting with Python, 3.5 or greater, Django 2 or greater, and Ubuntu as the deployment environment. To build, a text editor was used along with a virtual machine running Ubuntu for testing.

**5.2 Plans for ensuring requirements traceability**

The software requirements will be updated throughout the process, keeping revisions of this document along with corresponding version of the application for ease of traceability.

**5.3 Plans for testing the software**

The software will be tested for three major components. Functionality, design, and compatibility.

Functionality defined as functions behaving as expected. Functionality will be tested by testing edge cases, confirming expected output. This will be needed to tested for upon each update, and code update.

Design shall be tested by using cross browser testing, to make sure that all modern and supported browsers scale appropriately.

**5.4 Engineering trade-offs**

Voice Assist

Voice Assist could have been built by creating a backend service that streams audio from the client to the server. We would have needed to implement client-side recording and a way stream it. We felt that although this would have been the most robust solution, it would have been very time-consuming and there might have unaccountable input delays from the user.

Therefore, we decided to go with Google’s Speech API. It handles all the streaming and recognition. Allowing us to focus on the functionality and user experience of filling out the form via voice.

During implementation control flow is one way. Limiting how the users can move around the form via voice. This has meant needing to implement user verification in obvious ways.

OCR

For OCR if the user uploads a low-quality image they might have to manually fill out the application or correct the fields.

**5.5 Coding guidelines and conventions**

The code will follow the standards of the WCAG 2.0

**5.6 The protocol of one or more subsystems, modules, or subroutines**

All modules are independent of each other, so there is no protocol between the modules.

All the modules will submit the user information to the same database.

**5.7 Plans for maintaining the software**

Maintaining the software will be the responsibility of the customer

**5.8 Interface for end-users, software, hardware and communications**

When the user will use the online application, they will be how they would want to use the regular application, Voice Assist application, or the OCR application. The regular application is the regular application users will normally see when signing up for various web sites. The voice assisted application will be the standard application but will enable voice inputs. The user’s microphone will be used to communicate with the application. The OCR application will be the standard application, but the user will have to take a picture of their state ID or state Driver License and the application will be filled out. The user will have to have access to a camera or have a digital image of their ID for the OCR to work properly.

**5.8.1 Hierarchical organization of the source code**

djangoProject

|

|   manage.py

|   output.doc

|

+---djangoProject

|       settings.py

|       urls.py

|       wsgi.py

|

+---media

|       googleocr.py

|

+---static

|   \---website

|           addressValidation.js

|           javascript.js

|           stylesheet.css

|           voiceAssist.js

|

\---website

   | admin.py

   | apps.py

   | models.py

   | tests.py

   | urls.py

   | views.py

   |

   \---templates

       \---website

               htmlfiles.html

**5.9 Deliverables**

Voice

In order to link speech module, you need to link the Javascript source code into the html page where voice is needed. Assuming the html page has required tags and ids, linking the Javascript is all that is required to build the project.

OCR

Django’s media directory must be enabled for OCR to handle image uploads. The source code is exported with the Django application. Google Vision requires an API key to enable character recognition. Creating an API key is handled in Google APIs Console. Once the key is created, download the key and copy it to a secure location on the server that can only be accessed by administrators and Django. In the OCR application source code, import the key file with “os.environ[“GOOGLE\_APPLICATION\_CREDENTIALS”] = “(Ex. /Absolute/Path/File.json)”. The application might take a few minutes to authorize with Google Cloud

**5.10 Abstraction**

There is a generic SQL database with basic fields so that when changing the database will just be a matter of changing the User database class.

**6. Detailed System Design**

**6.1  GUI**

**6.1.1 Responsibilities**

The primary role of this module will be to allow users to enter data. It will give users access to an interface for users to select other services. This component will allow text entry, voice entry, and OCR entry.

**6.1.2 Constraints**

Only constraint is that this module must support IE10. This module must only make a request to the one endpoint. It must make sure to send valid data every time. Constraints are detailed within the WCAG specification.

**6.1.3 Uses/Interactions**

Will be used by the user to enter and select entry option. Will be a single file, built from Django with labels that describe the html appropriately.

**6.1.4 Resources**

This is the main module that will allow the user to select other services, and other modules. Without this module, a user will not be able to submit.

**6.2 Voice Assist**

**6.2.1 Responsibilities**

The primary responsibilities of this module is to assist users with disabilities fill out the application. It will provide voice feedback for each component of the application. It will also provide voice input so that the user will be able to fill out the application.

**6.2.2 Constraints**

The constraints for this component will be the quality of the user’s microphone. Depending on the user’s microphone quality the user’s voice input might not be able to pick up the exact words of the users.

**6.3 OCR**

**6.3.1 Responsibilities**

The primary responsibilities of this module are to enable entering information using images of a user’s driver’s license. This module shall allow the user to use their phones built in camera to take a picture of their driver's license or allow a user to upload a scanned image file. This module shall scan the image of the driver's license and extract the first name, last name, and home address.

**6.3.2 Constraints**

This module’s effectiveness is strongly dependent on both the accuracy of the OCR algorithm of this module and the quality of the input image. Colors other than black and white must be taken into consideration. Quality of the image limits the accuracy of this module

**6.3.3 Composition**

This module contains subcomponents for accessing the camera on the user’s phone, sending the image to the algorithm, performing Optical Character Recognition on the image, filtering out the unwanted information (i.e. hair color, eye color), and sending the extracted text to fill the application on the user’s end.

**6.3.4 Uses/Interactions**

OCR module only interacts with the user’s device and the fields of the application.

**6.3.5 Resources**

OCR module shall include API keys to access Google Cloud Vision API.

**6.4  Email Confirmation**

**6.4.1 Responsibilities**

This module is in charge of validating the user input and the email address given by the user. This module shall connect with appropriate Google services to autofill and autocomplete the addresses.

**6.4.2 Constraints**

This module will have a constraint in regard to how many calls are allowed to the Google services. We are constrained at 1000 calls per day of application run time.

**6.4.3 Composition**

This will be a function that will activate once the user starts to enter an address. This function will be inside of the main definition.

**6.4.4 Uses/Interactions**

This will only interact with the field in the sense that it will autofill the address field.

**6.5 Address Lookup**

**6.5.1 Responsibilities**

This module is in charge of validating the user input and the address given by the user. This module shall connect with appropriate Google services to autofill and autocomplete the addresses.

**6.5.2 Constraints**

This module will have a constraint in regard to how many calls are allowed to the Google services. We are constrained at 1000 calls per day of application run time.

**6.5.3 Composition**

This will be a function that will activate once the user starts to enter an address. This function will be inside of the main definition.

**6.5.4 Uses/Interactions**

This will only interact with the field in the sense that it will autofill the address field.

**7. Detailed Lower Level Component Design**

**7.1 Speech.js**

**7.1.1 Classification**

Voice Assist functions:

* onend
* sayMessage
* onresult
* insertValue
* cleanupEmail
* saidYes
* concatBirth
* clearInputs
* startSignUp
* askUserQuestion
* confirmInput
* asyncCall
* onload

**7.1.2 Design Class Hierarchy**

There is no class hierarchy in this application.

**7.1.3 Restrictions/Limitations**

Google Chrome is currently the only web browser that is compatible with Voice Assist.

**7.1.4. Performance Issues**

Voice Assist currently has some latency issues on when the user will be able to speak and when the application will pick up the user voice input

**7.1.5 Design Constraints**

Currently for Voice Assist we are only able to stay on the field and move forward. We currently are not able to field navigate the voice form freely.

**7.2 OCR**

**7.2.1 Classification**

OCR functions:

* upload
* scan
* delete
* filterFirstName
* filterLastName
* filterMiddleName
* filterAddress
* filterDob
* filterDlNum

**7.2.2 Design Class Hierarchy**

There is no class hierarchy in this application.

**7.2.3 Restrictions/Limitations**

OCR has a file size cap that cannot handle pictures with quality greater than 16MP. Some devices may require change in camera settings in order to upload successfully.

**7.2.4. Performance Issues**

OCR requires more time to process an image when the file size is larger. The application bottlenecks at image processing when multiple large image files are queued.

**7.2.5 Design Constraints**

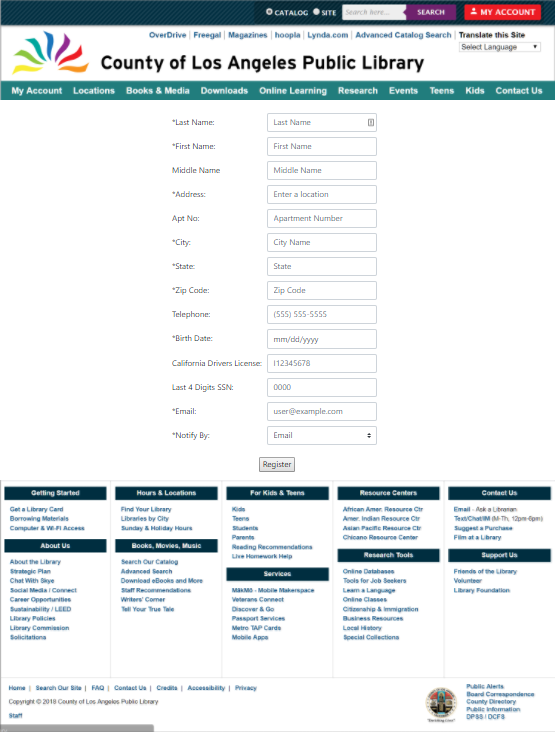
OCR at best can accurately read clear non-distorted images. If an uploaded image was taken with poor quality (low resolution, low light, excessive light, out of focus), the resulting data is not guaranteed to be correct. OCR also cannot accurately read images with glare or watermarks blocking important text.

**8. User Interface**

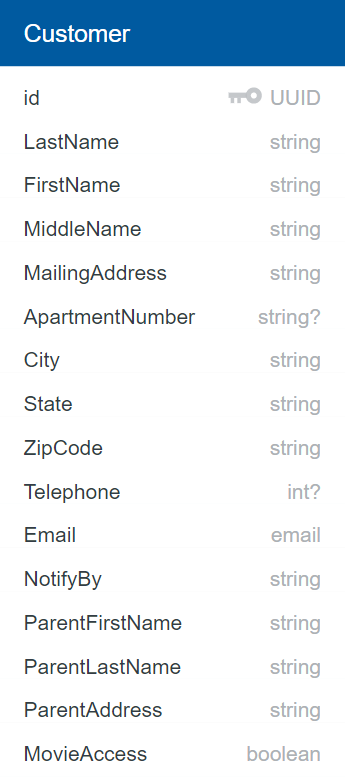
**8.1  Overview of User Interface**

The user interface, or GUI will consist of a data entry fields. It will implement responsive design to display the application for various screen sizes. There will be a button, it will turn on the Voice Assist feature. There will be a submit that will submit the entry into the Main module. It will be single page, and should be easy to read.

**8.2  Screen Frameworks or Images**



**8.3 Database Design**



**9. Requirements Validation and Verification**

|  |  |
| --- | --- |
| Requirements Related to Design Module 9.1 GUI | |
| Requirement No. |  |
| 9.1-1 | The GUI shall consist of input elements for: First name, Middle name, Last name, Address 1, Address 2, City, State, Zip code, Apartment number, Residence address, Birthday, Phone number, California Driver License Number, last four digits of Social Security Number, E-mail, Electronic signature. |
| 9.1-2 | The GUI shall have a Voice Assist function that will read all text field prompts |
| 9.1-3 | The GUI shall use the Google Translate API to translate application language to languages that are not English or Spanish |
| 9.1-4 | The GUI shall have an Email Confirmation function |
| 9.1-5 | The GUI shall have a submit button that will send the user input to a database. |
| 9.1-6 | The GUI shall have image upload option that will allow users to upload an image of their driver's license |
| 9.1-7 | The GUI shall have an Address validation that will confirm if the Address is located in California |
| 9.1.8 | The GUI module shall ensure that the application will comply with the WCAG requirements. |
| 9.1-9 | The GUI shall have a reCAPTCHA checkbox |

|  |  |
| --- | --- |
| Requirements Related to Design Module 9.2 Voice Assist | |
| Requirement No. |  |
| 9.2-1 | The Voice Assist shall be able to guide user through input fields |
| 9.2-2. | The Voice Assist shall automatically fill fields based on user voice response |
| 9.2-3 | The Voice Assist shall confirm each input field with user |

|  |  |
| --- | --- |
| Requirements Related to Design Module 9.3 Optical Character Recognition (OCR) | |
| Requirement No. |  |
| 9.3-1. | The OCR shall enable access to the camera of the user’s phone or allow an image type file upload. |
| 9.3-2. | The OCR shall scan an uploaded image file of a driver’s license to recognize the characters. |
| 9.3-3 | The OCR shall be able to fill for the following fields: First Name, Last Name, Middle Initial, Birthdate, and Address. |
| 9.3-4 | The OCR shall determine which data is associated with which field. An address indicated on a driver’s license shall be directed to the address field. |
| 9.3-5 | The OCR shall communicate with the Address Lookup module to verify the address field filled by OCR. |

|  |  |
| --- | --- |
| Requirements Related to Design Module 9.4 Email Confirmation | |
| Requirement No. |  |
| 9.4-1 | The Email Confirmation shall send an email to user after submission to verify email |
| 9.4-2 | The Email Confirmation shall send unique codes to each email provided |
| 9.4-3 | The Email Confirmation shall store these unique codes in the database for each user. |
| 9.4-4 | The Email Confirmation shall update the database if the user does verify the user’s email by comparing the code the user provides, with the code in the database. |
| 9.4-5 | The Email Confirmation shall have an API that the link will make a request to that will update, and tag user as valid. |
| 9.4.6 | The Email Confirmation module shall make calls to the Library API to record the data provided |
| 9.4.7 | The Email Confirmation module shall generate a login number after the user clicks the confirmation URL link in their email |

|  |  |
| --- | --- |
| Requirements Related to Design Module 9.5 Address Lookup | |
| Requirement No. |  |
| 9.5-1 | The Address Lookup shall query Google Maps API with user input or OCR input. |
| 9.5-2 | The Address Lookup shall accept only valid California street addresses. |
| 9.5-3 | The Address Lookup shall return an error if a valid California street address is not found. |
| 9.5-4 | The Address Lookup shall automatically complete form fields with the validated information obtained from Google Maps API |

**11. Glossary**

* LCRS - Library Card Registration System
* OCR - Optical Character Recognition
* Voice Assist - Voice Assist
* reCAPTCHA - a CAPTCHA-like system designed to establish that a computer user is human (normally in order to protect websites from bots) and, at the same time, assist in the digitization of books.
* CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart
* EULA - End User License Agreement
* API - Application programming interface

**12. References**

Django

* <https://www.djangoproject.com/>
* <https://docs.djangoproject.com/en/2.0/intro/tutorial01/>
* <http://www.django-rest-framework.org/tutorial/1-serialization/>

Google

* <https://cloud.google.com/translate/docs/>
* <https://developers.google.com/maps/>
* <https://developers.google.com/maps/documentation/javascript/places-autocomplete>
* <https://cloud.google.com/vision/>
* <https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API>

Digital Ocean

* <https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04>
* <https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-16-04>

USPS

* <https://www.usps.com/business/web-tools-apis/welcome.htm>