



iLegalSelfHelp: Legal Aid Made Easy

Project Documentation

Version: 1.1

Revision Date: 11.30.2017

Stakeholder: Yadira Davis

Pooja Chebolu

Peter Lu

Sean Yuan

Jennifer Zheng



TABLE OF CONTENTS

TABLE OF CONTENTS	2
HIGH LEVEL REQUIREMENTS	3
DELIVERABLES SCHEDULE	4
USER INTERFACE + WIREFRAME	6
Android	6
iOS	10
Front End	12
CLASSES OVERVIEW	15
Android	15
iOS	16
Front End	17
Back End	17
DATABASE SCHEMA	18
TESTING NEW FEATURES	19
ADMIN PORTAL OVERVIEW	21
How to Upload a PDF	22
How to Configure Push Notifications	24
How to Configure PDF Mapping Tool Fields	25
CIVIL CODE LIST	27
PDF LIST	28
ACCOUNTS INFORMATION	30
ENVIRONMENT SETUP + APP DEPLOYMENT	31
Java Servlet	31
Android	32
iOS	32
FEATURES BACKLOG + SUGGESTIONS FOR NEXT TEAM	33
CONTACT INFORMATION	34
PREVIOUS DOCUMENTATION	35



HIGH LEVEL REQUIREMENTS

Stakeholder: Yadira Davis <yadira40@yahoo.com>

Team Name: iLegalSelfHelp

Estimated Team Size: 3

Technologies Expected: iOS, Android

Background Requested: iOS or Android development

Description: Continuing the development of the first non-profit self-help app for both Android and iOS, this app will allow millions of Angelinos to search, fill, and file court documents within the Los Angeles County. This project is a continuation of a previous 401 project. The app is already built with the ability to upload and generate fillable PDFs.

I am looking to incorporate a few features, such as, but not limited to:

- “live-chat” via instant messages.
- notifications
- email connection
- payment method
- updating PDF submission

This app is geared toward assisting the underrepresented communities in Los Angeles County and offers, each who downloads, the opportunity for legal learning.

DELIVERABLES SCHEDULE

#	Date	What We Promise	What Was Delivered
1	09/01/2017	Deliverables Schedule Software Development Planning	Previous Teams' Work Comprehension and Scheduling of Deliverables as Outlined
2	09/15/2017	Email Connection Logo (including phrase) + Splash Screen Source Code Comprehension and Refactoring	Email Connection and Recover Password Functionality Splash Screen, Remember Me Functionality, User Interface (UI) Working Updates Account Creation Workflow Updates and Profile Edit/Update Features Source Code Comprehension and Preliminary Refactoring for Optimized Performance
3	09/29/2017	Push Notifications Live Chat	Android Push Notifications Functionality PDF Submission and Backend Workflow Mapping Preliminary Functionality Updates
4	10/13/2017	Update PDF Submission and Backend Workflow Mapping	iOS Push Notifications Admin Portal Frontend Updates PDF Submission and Mapping Frontend Working Functionality Updates

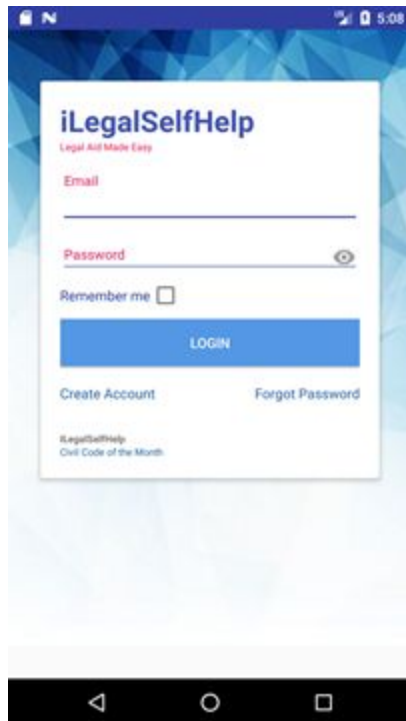
5	10/27/2017	Payment Methods User Interface (UI) Updates	<p>Live Chat Architecture and Functionality for Android and iOS</p> <p>Push Notifications Admin Console Updates (One-Time and Recurring Notification Scheduler)</p> <p>PDF Submission and Mapping Backend Working Functionality Updates</p> <p>Payment Method Integration Was Tabled (a donation website or crowdfunding page may be added within the app via a link/push notifications)</p>
6	11/10/2017	User Tutorials Testing	<p>PDF Submission and Mapping Tool Functionality Completion</p> <p>Push Notifications One-Time and Recurring Scheduler Completion</p> <p>Preliminary User Tutorials (App Workflow + Deployment Instructions) Are Included Within This Documentation Package; Comprehensive Tutorials/Videos Have Been Tabled For the Final Team That Works on This Project</p>
7	11/30/2017	Complete Documentation	<p>Project Documentation Package + Accounts Information With Link to GitHub for Source Code Access</p>

USER INTERFACE + WIREFRAME

Android

#	Description
1	Landing Page/Login Screen
2	User Registration Screen
3	Password Recovery (Forgot Password) Screen
4	Edit/Update User Profile Screen
5	Reset Password Screen
6	User Home/PDF Categories Screen
7	PDF Specific Category Screen
8	Fill Out PDF Initial Screen
9	Fill Out PDF Fields Screen
10	Fill Out PDF Confirmation Screen
11	Saved PDFs Screen
12	Email Support Screen
13	Chat Screen
14	Admin Active Chat User List Screen
15	Push Notification

1



iLegalSelfHelp
Legal Aid Made Easy

Email

Password

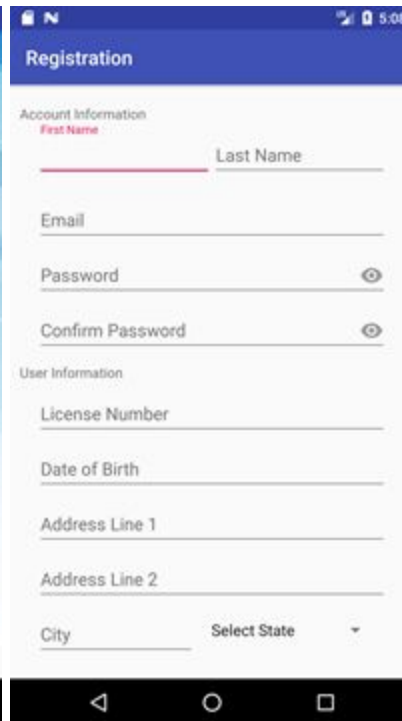
Remember me ☐

LOGIN

Create Account Forgot Password

iLegalSelfHelp
Civil Code of the Month

2



Registration

Account Information

First Name

Last Name

Email

Password

Confirm Password

User Information

License Number

Date of Birth

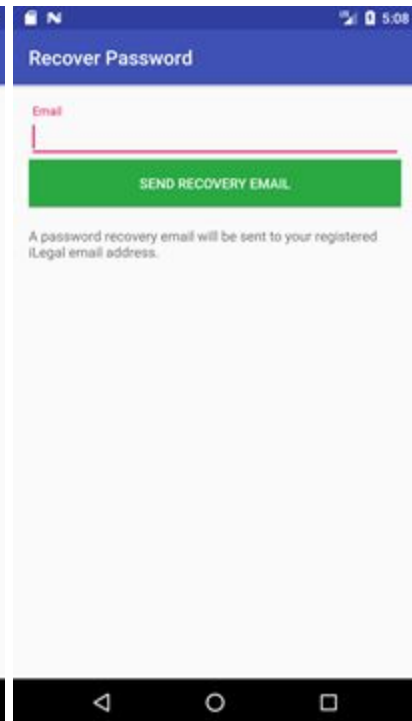
Address Line 1

Address Line 2

City

Select State

3



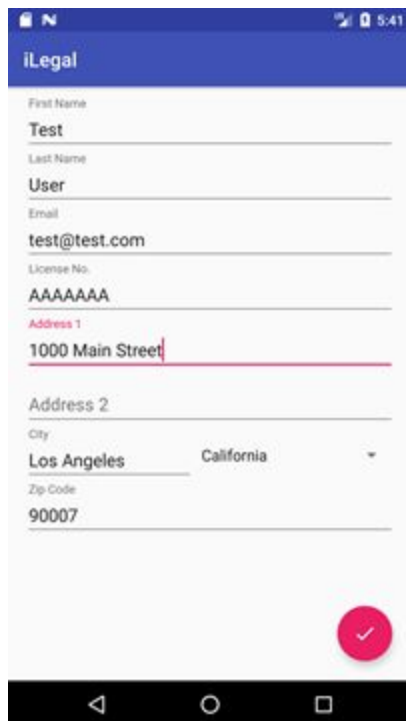
Recover Password

Email

SEND RECOVERY EMAIL

A password recovery email will be sent to your registered iLegal email address.

4



iLegal

First Name

Test

Last Name

User

Email

test@test.com

License No.

AAAAAAA

Address 1

1000 Main Street

Address 2

City

Los Angeles California

Zip Code

90007

5



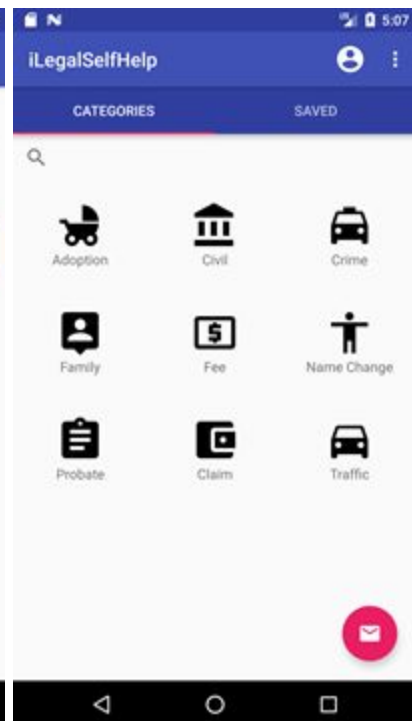
iLegal

New Password

Confirm password

RESET

6



iLegalSelfHelp

CATEGORIES SAVED

Adoption

Civil

Crime

Family

Fee

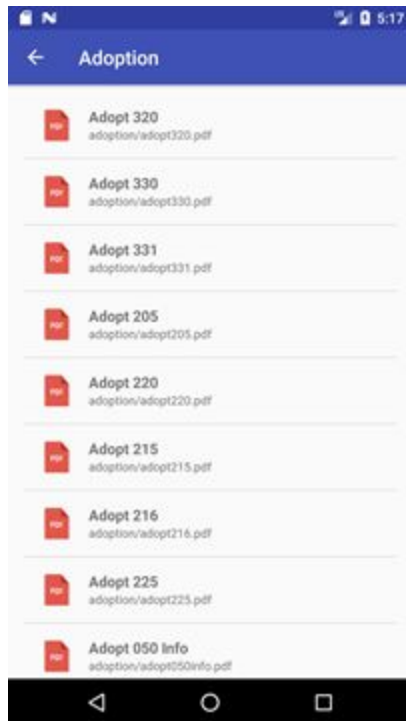
Name Change

Probate

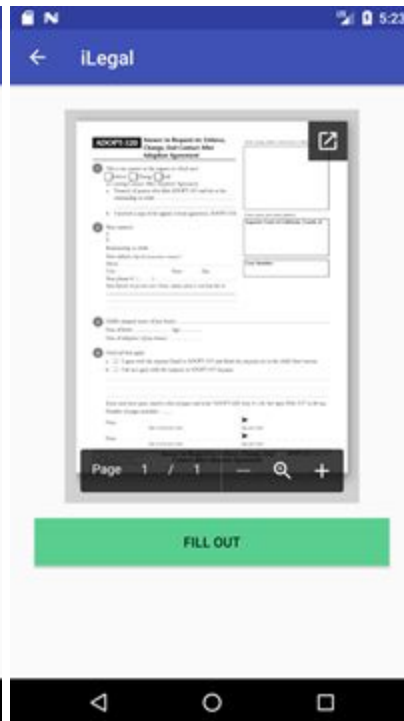
Claim

Traffic

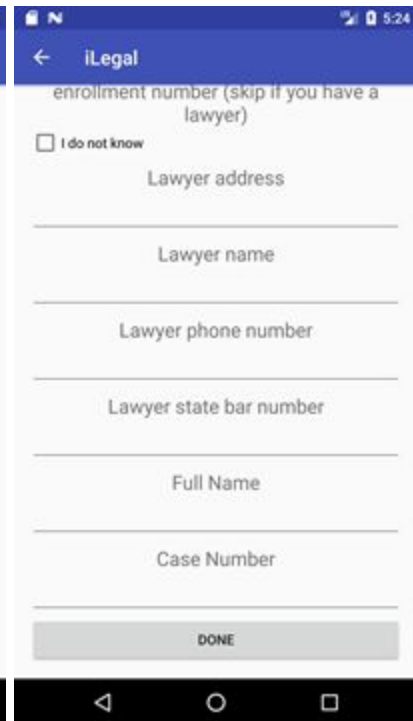
7



8



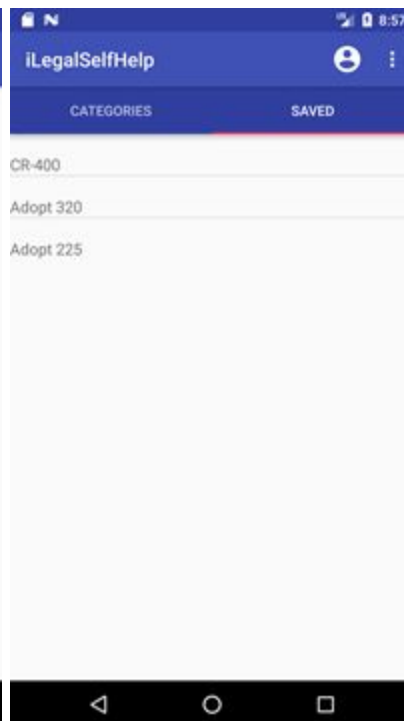
9



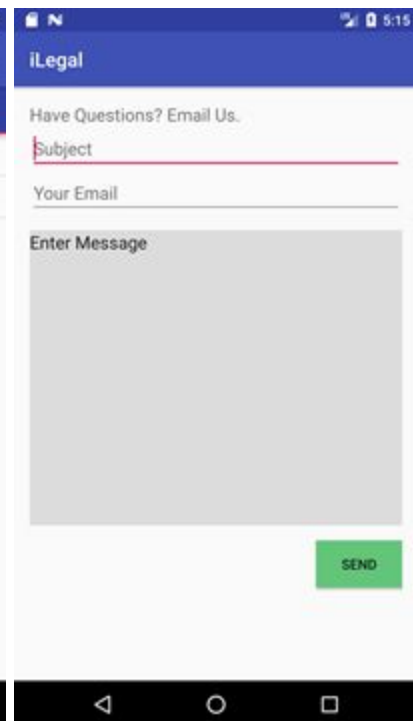
10



11



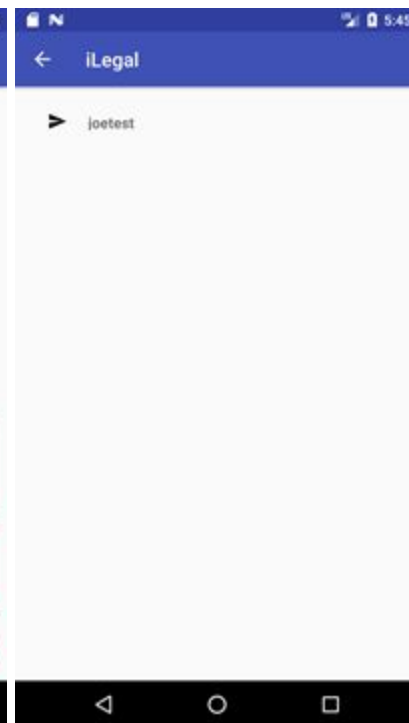
12



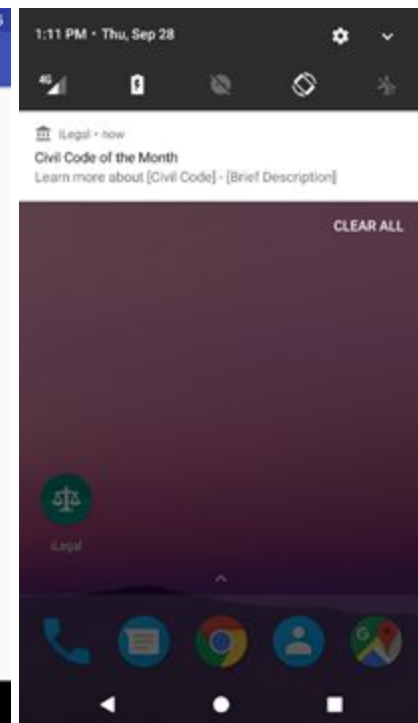
13



14



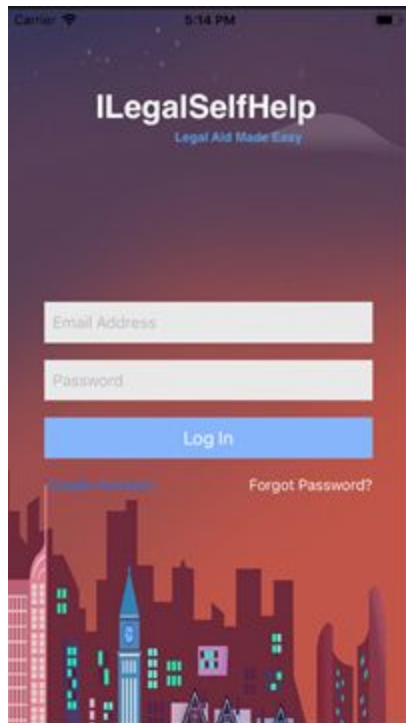
15



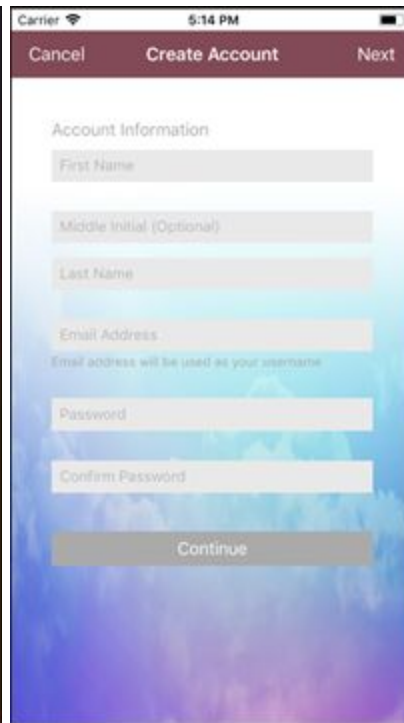
iOS

#	Description
1	Landing Page/Login Screen
2	User Registration Screen
3	Password Recovery (Forgot Password) Screen
4	User Profile Screen
5	PDF Categories Screen
6	PDF Specific Category Screen
7	Fill Out PDF Initial Screen
8	Fill Out PDF Fields Screen
9	Saved PDFs Screen
10	Email Support Screen
11	Chat Screen
12	Push Notification

1



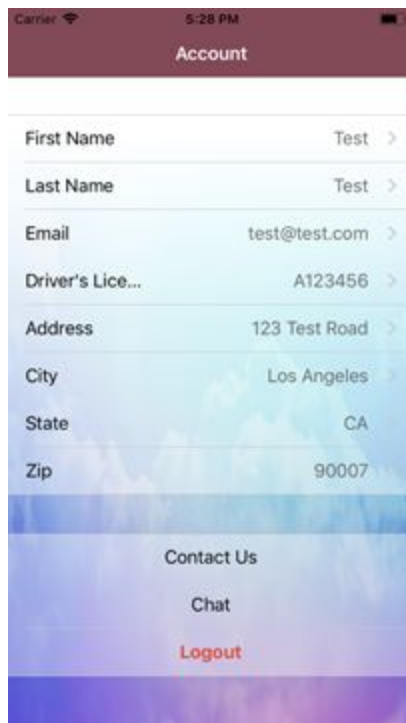
2



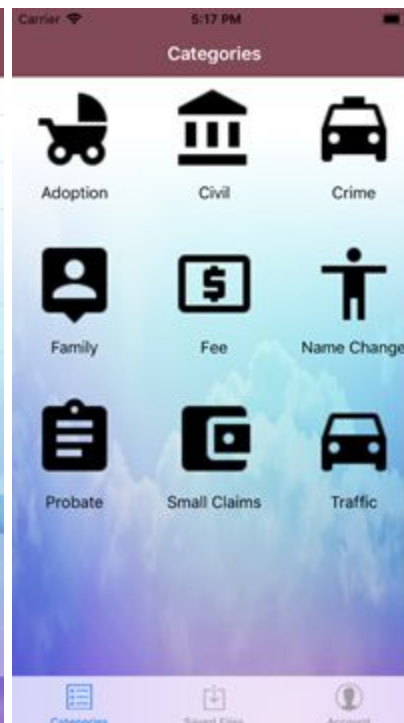
3



4



5



6



7

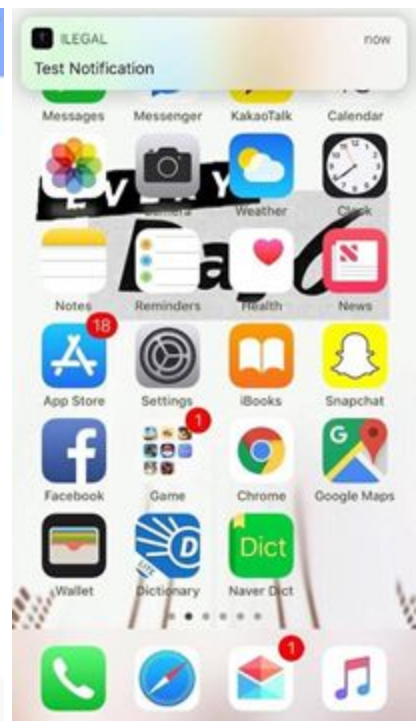
8

9

10

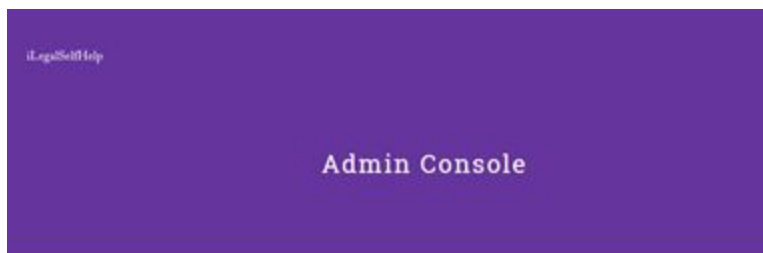
11

12



Front End

#	Description
1	Admin Portal Landing Screen
2	PDF Upload Screen
3	PDF Mapping Screen
4	PDF Upload Confirmation Screen
5	Push Notification Scheduler Screen
6	Add Mapping Field Screen



1



2

3

iLegalSelf Help

UPLOADNOTIFICATIONSMAPPING

PDF: adopt020-602PM-Category: adoption Successfully Uploaded.

How to use the mapper:

1. The fields on the left are the fields in the uploaded PDF.
2. The fields on the right are the available fields for mapping.
3. Drag the fields you wish to map from the left and drop them in the appropriate mapping field box on the right.
4. Click Map Fields.

Name(s) of person who filed ADOPT \$15.	First Name
Your relationship to child	Last Name
Case Number (if known)	Middle Name
This is my answer to the request to: (check one)(Inform)	Date of Birth
This is my answer to the request to: (check one)(Change)	Address
This is my answer to the request to: (check one)(No)	City
Full Name	Zip Code
	Phone Number
	State

4

iLegalSelf Help

UPLOADNOTIFICATIONSMAPPING

PDF Upload

Mapping Fields Success, PDF is ready to go!

5

iLegalSelf Help

UPLOADNOTIFICATIONSMAPPING

Notification Console

Schedule a recurring notification to iLegal users, or choose "One-Time" under Frequency to send a notification now.

Title	Message	Frequency	
		One-Time	<div>SCHEDULE</div>



How to add fields to the mapping tool (for auto-fill purposes):

1. Choose an appropriate category
2. Click list
3. Review the list of available mapping fields for that category
4. Enter the name of a new mapping field.
5. Click Add Field

Adoption

Adopted Child's Name

Adopted Child's Birthdate

Your Relationship to Child

Spouse Name

Child's Address

Child's Age

6

CLASSES OVERVIEW

Android

Class Name	Type	Description
AlarmReceiver	New	Receiver class for server side scheduled notifications. Creates actual notification UI bundle
CategoriesFragment	Modified	Fragment with 9 category sections
Chat	New	Chat room - send and receive message fields
ChatListViewActivity	New	Admin only - list of all ongoing chats.
ChatMessages	New	Individual chat bubble class
CivilCodeAlarm	New	Receiver class for Civil Code notifications
CompletedPDFActivity	Modified	Displays newly filled PDF
CreateAccountActivity	Modified	Register account + schedule to monthly civil codes (Firebase)
DocsListViewActivity	Modified	List of PDFs after a category is selected
Email	New	Serializable email class
EmailActivity	New	Message compose class
EmailSender	New	Parse email and route to admin
FillPDFActivity	Modified	Parsed text from PDF, allows user to fill out info and save
FirebaseIDService	New	Firebase token requester

FirebaseNotifications	New	Receiver class for one-time server notifications. Creates actual notification UI bundle
ForgotPasswordActivity	New	Request temporary password for reset
ForgotPasswordEmailSender	New	Send temporary password information to requester
HashPassword	Modified	Password hashing for server
HighLevelCategories	Modified	View holder for PDF categories
MainActivity	Modified	Log-in screen
MainNavigationActivity	Modified	Main view for categories and menu
MessageAdapter	New	Chat room - display chat bubble based on user ID
PagerAdapter	Modified	Adapter for PDF preview
PdfInfo	Modified	Singleton for imported PDF
PreviewPDFActivity	Modified	Preview PDF holder
ProfileActivity	New	Holds user profile information and sends request to server when updating
ProfileFragment	Modified	No longer used
ResetPasswordActivity	New	Framework for resetting password information through email
SavedPdfsListviewFragment	Modified	List of saved PDF's
SplashActivity	Modified	Splashscreen - unused
UserSingleton	Modified	User object

iOS

Class Name	Type	Description
AccountViewController	Modified	Displays account information and methods for contacting admins/support
AppDelegate	Modified	Setup and configure Firebase
BackEnd	Modified	Backend functions to retrieve user data and saved pdfs
CategoriesViewController	Unchanged	Main screen with all form categories
CategoryCell	Unchanged	CollectionView cell for category
Chat	New	Chat class
ChatViewController	New	Chat room with send and receive message fields
CheckBox	Unchanged	Check box class
CheckBoxGroup	Unchanged	Wrapper with array of check boxes

ContactUsViewController	Modified	Sets up mail view composer
Constants	New	Constants used to improve readability of Firebase queries
CreateAccountViewController	Unchanged	Registering new account screen
DocumentsViewController	Unchanged	Displays documents within a category
DonePDFViewController	Unchanged	Shows finalized PDF
DropDownPickerView	Unchanged	Dropdown menu
EditPDFViewController	Unchanged	Screen for editing PDF
Extensions	Unchanged	Functions for strings
ForgotPasswordViewController	Modified	Request temporary password for reset
Form	Unchanged	Form class
ListViewController	New	Admin only - List of all ongoing chats
PDFTextField	Unchanged	Text field in PDF
PreviewPDFViewController	Unchanged	Shows original PDF before editing
SavedFilesViewController	Unchanged	Shows user's saved files
UpdateUserViewController	Unchanged	Edit user's personal information
User	Unchanged	User class
UserPropertyCell	Unchanged	TableViewCell for user property
ValueCell	Unchanged	TableViewCell for PDF value field
ViewController	Unchanged	Login screen
ViewPDFViewController	Unchanged	Shows PDF
YesNoButton	Unchanged	UIButton for selecting yes/no

Front End

File Name	Type	Description
index.html	Modified	Admin portal landing page (login to access admin features)
mapping.html	New	Add new mapping fields for PDF categories
notification.html	New	Schedule push notifications
upload.html	Modified	Upload and map new PDF

Back End

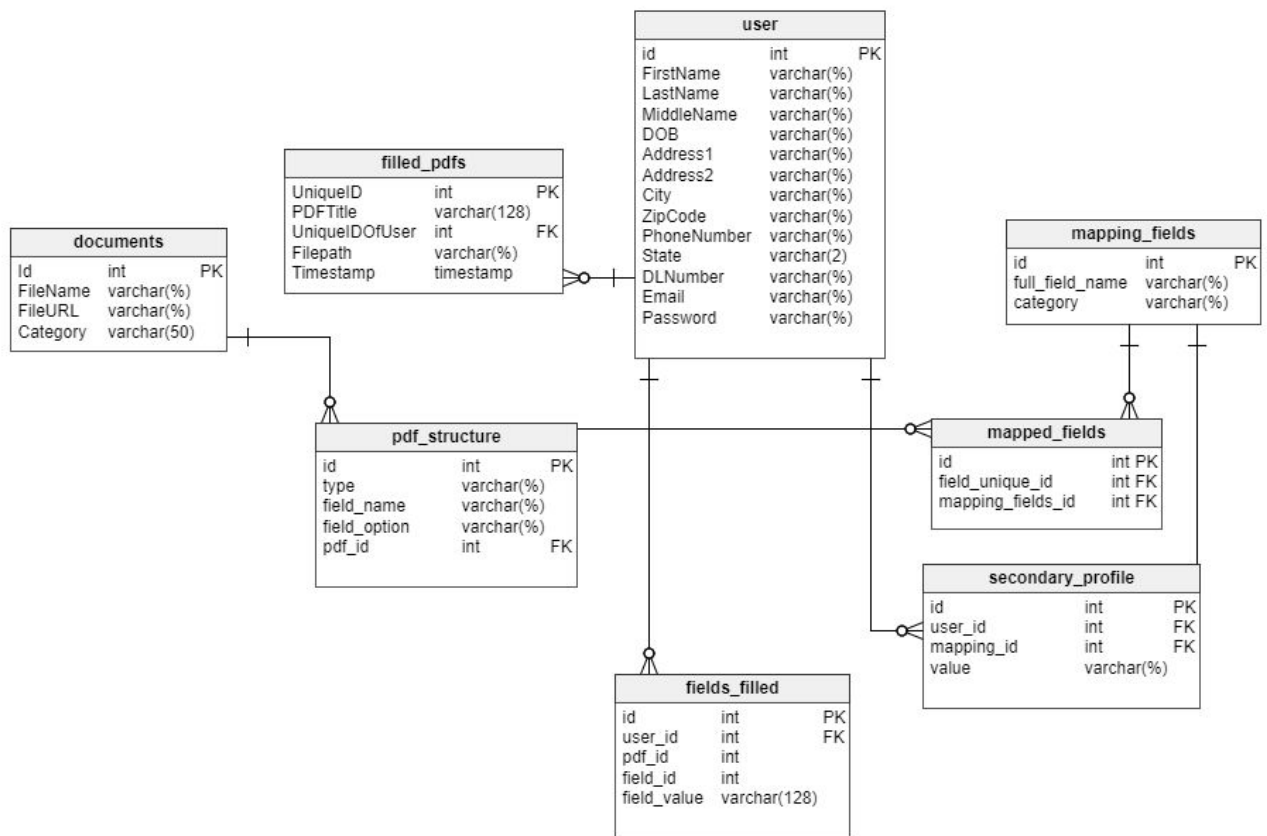
Class Name	Type	Description
DB	Unchanged	Functions to find user by ID and update user by ID
DropPDF	Unchanged	Deletes PDF from filled_pdfs DB table by unique ID

FillPDF	Unchanged	Fill PDF and push to filled_pdfs DB table
GenPDF	Unchanged	Generate JSON with the requested PDF's field names, options, and types
GetMappedField	New	Returns the autofill value given the PDF ID, field name, and user ID, which is saved in either the user (general) or secondary_profile (category specific) tables
GetMappingFields	New	Returns list of mapping fields for the specified category
HelperFunctions	Modified	Returns a list of the fields in the requested PDF file
InsertMappedFields	New	Insert new pairing between PDF field and autofill mapped field into mapped_fields DB
InsertMappingField	New	Insert new autofill mapping field for specified category into mapping_fields DB
ListPDF	Unchanged	Selects specific PDFs from various DB tables
Main	Unchanged	Creates new account in user DB table
PDFLoad	Unchanged	Inserts new PDF row into documents DB table and inserts each of the PDF's fields into the pdf_structure DB table
SignIn	Unchanged	Authenticate sign in request with user DB table and returns User singleton data if successful
UpdateSecondary	New	Either updates to override autofill field value or inserts new autofill field value into secondary_profile DB table
UpdateUser	Modified	Update user with new profile value
Web.xml	Modified	Maps the servlet name with the servlet class (Java class name) and the servlet name with the url pattern. Must update this file in order for the servlet to properly map to the corresponding Java class



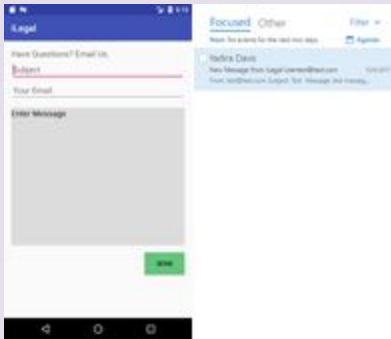
DATABASE SCHEMA

Table Name	Type	Description
documents	Unchanged	PDF general information (name, category, filepath)
fields_filled	Unchanged	User-entered values for each field in his/her filled PDFs
filled_pdfs	Unchanged	Filled PDFs general information (title, filepath, time filled)
mapped_fields	New	Mappings between unique pdf fields and available mapping fields
mapping_fields	New	Available mapping fields general information (name,

		category)
pdf_structure	Unchanged	PDF structure general information (field name, field type)
secondary_profile	New	User-specific secondary profile information
user	Unchanged	General profile information (name, birthdate, address, phone, driver's license number, email, encrypted password)



TESTING NEW FEATURES

Feature	Interface	Tests
Live Chat		<ol style="list-style-type: none"> Admin selects menu <ol style="list-style-type: none"> Expected result - chat list User selects menu <ol style="list-style-type: none"> Expected result - private chat Sending blank chat <ol style="list-style-type: none"> Expected result - error message
Push Notifications		<ol style="list-style-type: none"> Sending one-time notification (Admin Portal) <ol style="list-style-type: none"> Blank message <ol style="list-style-type: none"> Expected result - fail Blank title <ol style="list-style-type: none"> Expected result - fail Sending recurring notification (Admin Portal) <ol style="list-style-type: none"> Blank message <ol style="list-style-type: none"> Expected result - fail Blank title <ol style="list-style-type: none"> Expected result - fail Civil Code of Month notification <ol style="list-style-type: none"> Select notification <ol style="list-style-type: none"> Expected result - open application with updated Civil Code Send blank message/title <ol style="list-style-type: none"> Expected result - fail
Email Connection		<ol style="list-style-type: none"> User selects menu <ol style="list-style-type: none"> Expected result - email form Sending email with blank email subject/sender's email/message <ol style="list-style-type: none"> Expected result - fail Sending filled email <ol style="list-style-type: none"> Expected result - send email to admin Admin accessing email <ol style="list-style-type: none"> Expected result - sent email seen in inbox

Admin Portal - Mapping



1. Admin loads PDF with blank name/category
 - a. Expected result - fail
2. Admin loads PDF with name/category
 - a. Expected result - PDF mapping screen
3. Admin maps desired fields
 - a. Expected result - PDF upload confirmation screen
4. Admin selects category in Mapping tab
 - a. Expected result - currently available mapping fields displayed
5. Admin adds new mapping field and selects category
 - a. Expected result - successful addition of mapping field message

PDF Submission Process



1. User selects PDF
 - a. Expected result - PDF preview
2. User selects Fill Out
 - a. Expected result - PDF Fillable Screen
3. User fills out desired fields and presses Done
 - a. Expected result - PDF fill confirmation; option to email/return

ADMIN PORTAL OVERVIEW

Link: <http://159.203.67.188:8080/html/index.html>

Login: See accounts information section

The administrator portal gives the stakeholder the ability to upload PDFs, add mapped autofill fields, and configure push notifications. Originally, the portal only had the functionality to upload a fillable PDF to the backend server. Furthermore, the front-end was bare bones and lacked usability. The interface of the portal was redesigned with HTML5/CSS and Javascript to provide a user-friendly design for the administrator. In addition to uploading a fillable PDF, the administrator can also map the fields on the PDF. They will have a choice of mapping a field from the PDF with any autofill fields from the general category or the specific category of the PDF.

How to Upload a PDF

Link: <http://159.203.67.188:8080/html/upload.html>

1. Follow the instructions on the upload page to choose a PDF file, title, and category, as shown below.
 - a. Note: These PDF files must have fillable fields (we created these files manually using Adobe Acrobat Pro DC [further instructions can be found in the Spring 2017 Team Documentation])



How to use the uploader:

1. Choose a new PDF by clicking the browse button
2. Choose an appropriate title
3. Choose an appropriate category
4. Click Go

Choose File adopt320.pdf
adopt320 Adoption GO!

2. Press GO! to upload the PDF file and move on to mapping page.

PDF Upload

PDF: adopt320 Category: adoption Successfully Uploaded.

How to use the mapper:

1. The fields on the left are the fillable fields in the uploaded PDF.
2. The fields on the right are the available fields for mapping.
3. Drag the fields you wish to map from the left and drop them in the appropriate mapping field box on the right.
4. Click Map Fields

City
State
Your lawyer if you have one Name address phone and State Bar 1
Your lawyer if you have one Name

First Name
Last Name
Middle Name
Date of Birth

On the mapping page, there are two columns of draggable fields. The left column includes all the fields scraped from the fillable PDF. The right columns includes the available autofill fields the PDF fields can be mapped to. In the right column, the first 11 fields are general profile fields and the following fields are mapping fields specific to the PDF's category.

1. The fields on the left are the fillable fields in the uploaded PDF.
2. The fields on the right are the available fields for mapping.
3. Drag the fields you wish to map from the left and drop them in the appropriate mapping field box on the right.
4. Click Map Fields

City
State
Your lawyer if you have one Name address phone and State Bar 1
Your lawyer if you have one Name address phone and State Bar 2
Your lawyer if you have one Name address phone and State Bar 3
Childs adopted name if you know
Name(s) of person who filed ADOPT 315:
relationship to child:
Your relationship to child:
Area Code
Phone Number
Child DOB
Child Age
Date of adoption

First Name
Last Name
Middle Name
Date of Birth
Address
City
Zip Code
Phone Number
State
Driver's License Number
Email
Adopted Child's Name
Adopted Child's Birthday
Your Relationship to Child
Spouse Name

3. Drag a PDF field from the left and drop it into the appropriate mapping field box on the right. Repeat for all desired mapped fields.



iLegalSelfHelp

1. The fields on the left are the fillable fields in the uploaded PDF.

2. The fields on the right are the available fields for mapping.

3. Drag the fields you wish to map from the left and drop them in the appropriate mapping field box on the right.

4. Click Map Fields

Your lawyer if you have one Name
address phone and State Bar 1

Your lawyer if you have one Name
address phone and State Bar 2

Your lawyer if you have one Name
address phone and State Bar 3

Childs adopted name if you know

Name(s) of person who filed ADOPT
315:

relationship to child:

Your relationship to child:

Child DOB

Child Age

Date of adoption

Date

Your Name (2)

Date (2)

First Name

Last Name

Middle Name

Date of Birth

Address

City

City

Zip Code

Area Code

Phone Number

Phone Number

State

State

Driver's License Number

Email

4. Scroll down and press the Map Fields! button to complete the process.

How to Configure Push Notifications

Link: <http://159.203.67.188:8080/html/notification.html>

1. Set title and message for notification
2. Set frequency for notification between one-time, weekly, or monthly.

iLegalSelfHelp

UPLOAD

NOTIFICATIONS

MAPPING

Notification Console

Schedule a recurring notification to iLegal users, or choose "One-Time" under Frequency to send a notification now.

Title	Message	Frequency	
Civil Code §23103 (a)	A person who drives a vehicle in willful or war	<div>✓ One-Time</div> <div>Weekly</div> <div>Monthly</div>	<div>SCHEDULE</div>

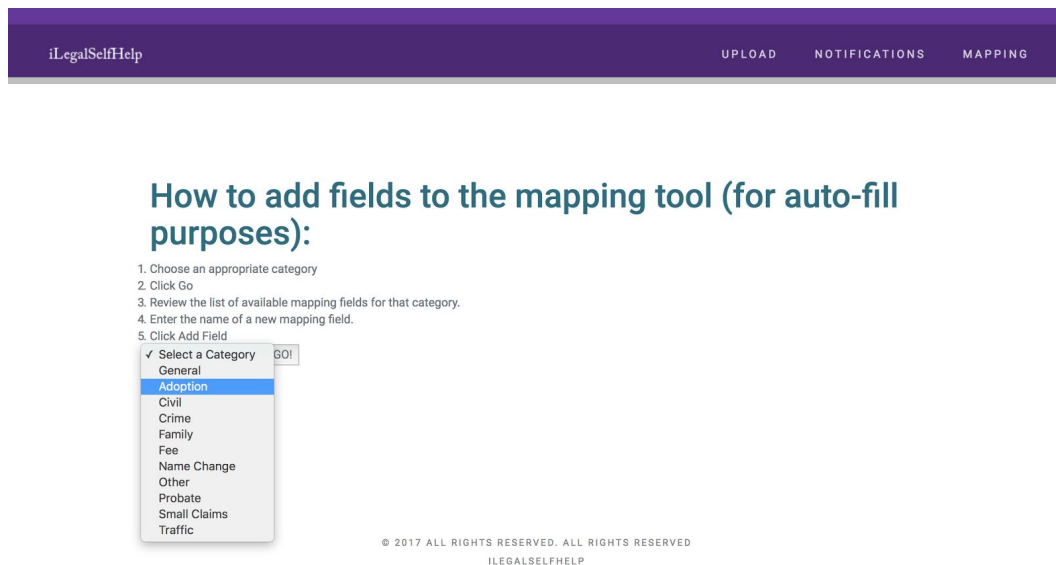
23

3. Press Schedule button to submit new notification.

How to Configure PDF Mapping Tool Fields

Link: <http://159.203.67.188:8080/html/mapping.html>

1. Choose the category that you would like to add mapping fields to from the dropdown menu.



2. Press Go! This will update the page with a list of the existing mapping fields for that category.
 - a. Note: You can list the existing mapping fields for the general category, however you will be unable to add any mapping fields to the general category.

How to add fields to the mapping tool (for auto-fill purposes):

1. Choose an appropriate category
2. Click Go
3. Review the list of available mapping fields for that category.
4. Enter the name of a new mapping field.
5. Click Add Field

Adoption GO!

Adopted Child's Name
Adopted Child's Birthday
Your Relationship to Child
Spouse Name
Child's Address
Child's Age

Select a Category Add Field!

3. Then you can type in the new mapping field in the text field.
4. Select the corresponding category that you would like to add the new mapping field to.
5. Press the Add Field! Button to submit the new mapping.

How to add fields to the mapping tool (for auto-fill purposes):

1. Choose an appropriate category
2. Click Go
3. Review the list of available mapping fields for that category.
4. Enter the name of a new mapping field.
5. Click Add Field

Adoption GO!

Adopted Child's Name
Adopted Child's Birthday
Your Relationship to Child
Spouse Name
Child's Address
Child's Age

Adopted Child's Phone Adoption Add Field!

CIVIL CODE LIST

Following is the list of civil codes that are used for push notifications (currently, one civil code is selected at random and pushed monthly):

Code	Description
§23103 (a)	A person who drives a vehicle in willful or wanton disregard is guilty of reckless driving.
§23140 (a)	It is unlawful for a person under the age of 21 years who has 0.05 percent or more, by weight, of alcohol in his or her blood to drive a vehicle.
§23249.50 (1)	Driving under the influence of alcoholic or drug is a serious problem, constituting the largest group of misdemeanor violations in many counties.
§23249.50 (2)	Studies of first offenders found that more than half of first offenders are alcoholics or problem drinkers.
§21200 (a)	A person riding a bicycle or pedicab has all the rights and is subject to all the provisions applicable to the driver of a vehicle.
§21205	No person operating a bicycle shall carry any package, bundle or article which prevents the operator from keeping at least one hand upon the handlebars.
§21213 (a)	A person under 16 years of age shall not operate a class 3 electric bicycle.
§21221	Every person operating a motorized scooter upon a highway has all the rights and is subject to all the provisions applicable to the driver of a vehicle
§21800 (a)	The driver of a vehicle approaching an intersection shall yield the right-of-way to any vehicle which has entered the intersection from a different highway.
§21952	The driver of any motor vehicle, prior to driving over or upon any sidewalk, shall yield the right-of-way to any pedestrian approaching thereon.
§21957	No person shall stand in a roadway for the purpose of soliciting a ride from the driver of any vehicle.

§21966	No pedestrian shall proceed along a bicycle path or lane where there is an adjacent adequate pedestrian facility.
§21970 (a)	No person may stop a vehicle unnecessarily in a manner that causes the vehicle to block a marked or unmarked crosswalk or sidewalk.
§298.5 (a)	Two persons desiring to become domestic partners may complete and file a Declaration of Domestic Partnership with the Secretary of State.
§30	The parties to crimes are classified as Principals and Accessories.
§208 (a)	Kidnapping is punishable by imprisonment in the state prison for three, five, or eight years.
§215 (b)	Carjacking is punishable by imprisonment in the state prison for a term of three, five, or nine years.
§236	False imprisonment is the unlawful violation of the personal liberty of another
§236.1 (g) (6)	“Great bodily injury” means a significant or substantial physical injury.
§236.1 (g) (7)	“Minor” means a person less than 18 years of age.
§240	An assault is an unlawful attempt, coupled with a present ability, to commit a violent injury on the person of another.
§242	A battery is any willful and unlawful use of force or violence upon the person of another.

PDF LIST

Following is the list of PDFs currently available for filling within the app:

#	Type	Category	Name
1	Unchanged	Adoption	Adopt-050-INFO
2	Unchanged	Adoption	ADOPT-205
3	Unchanged	Adoption	ADOPT-215
4	Unchanged	Adoption	ADOPT-216
5	Unchanged	Adoption	ADOPT-220
6	Unchanged	Adoption	ADOPT-225
7	Unchanged	Adoption	ADOPT-330

8	Unchanged	Adoption	ADOPT-331
9	New	Civil	APP-103
10	New	Civil	MC-001
11	New	Civil	MC-010
12	New	Civil	MC-275
13	New	Civil	MC-410
14	New	Claim	CP-10
15	New	Claim	DE-111
16	New	Claim	DE-142 DE-111(A-3e)
17	New	Claim	SC-100-INFO
18	New	Claim	SC-104C
19	New	Claim	WG-005
20	Modified	Crime	CR-400
21	Modified	Crime	CR-401
22	New	Crime	CR-402
23	New	Crime	CR-403
24	New	Crime	DV-800; JV-252
25	New	Crime	JV-060
26	New	Crime	JV-595-INFO
27	New	Crime	JV-596-INFO
28	New	Crime	JV-596
29	New	Crime	JV-600
30	New	Crime	JV-642
31	New	Crime	JV-710
32	New	Crime	JV-732
33	New	Crime	JV-735
34	New	Crime	JV-744
35	New	Crime	JV-744A
36	New	Crime	JV-745
37	New	Crime	JV-746
38	New	Crime	JV-794
39	New	Crime	SV-600 2018-01-01
40	New	Family	FL-303

41	New	Family	FL-306
42	New	Family	FL-307
43	New	Family	FL-323-INFO
44	New	Family	FL-510
45	New	Family	FL-570
46	New	Family	FL-590A
47	New	Family	FL-592
48	New	Family	FL-676-INFO
49	New	Family	FL-800
50	New	Family	FL-810
51	New	Family	FL-950
52	New	Family	FL-955-INFO
53	New	Family	FL-955
54	New	Family	FL-596
55	New	Family	FL-957
56	New	Family	FL-958
57	Unchanged	Fee	FW-001
58	Unchanged	Fee	FW-002
59	New	Fee	FW-008
60	Unchanged	Fee	FW-010
61	New	Probate	FL-200

*Note: We used our best judgment in assigning categories for certain PDFs; these categories may be changed either through the database or by removing & re-uploading the PDF with the appropriate category (new categories may also be programmed into the app to reflect any missing categories)

ACCOUNTS INFORMATION

Account	Credentials	Description
Admin Portal	Link: http://159.203.67.188:8080/html/index.html Password: ilegal	Access to administrator portal for uploading PDFs, mapping fields, and sending push notifications.
Firebase/ Gmail	Link: https://console.firebase.google.com/u/0/project/illegal-b9b07/overview	Access to Firebase Project (push notifications, chat)

	Email: ilegalselfhelp@gmail.com Password: Fighton17	
GitHub	Link: https://github.com/iLegalSelfHelpApp Email: info@ilegalselfhelp.com Password: Fighton17	Access to source code and application files
Database	Link: http://159.203.67.188/phpmyadmin/ Username: root Password: Trojans17	Access to MySQL GUI
DigitalOcean	Link: https://cloud.digitalocean.com/login Username: info@ilegalselfhelp.com Password: 2017Trojans	Access to project database host
Apple Developer	Email: Yadirao40@yahoo.com Password: **My06171986	Access to Apple Developer features
Java Servlet	ssh root@159.203.67.188 Password: Trojans17	Access to Java Servlet /opt/tomcat/webapps/ Access to admin portal files /opt/tomcat/webapps/html/
Tomcat Web Application Manager	Link: http://159.203.67.188:8080/manager/html/ Username: tomcat Password: pw	Portal to deploy the back end WAR file
GoDaddy	Link: https://sso.godaddy.com/?realm=idp&app=mya&path=&isc=gdbb3376 Username: Yadirao40 Password: 2017Trojans	Hosts the domain for ilegalselfhelp.com
Office 365 Email	Link: https://sso.godaddy.com/?app=o365&realm=pass&username=info@ilegalselfhelp.com Username: info@ilegalselfhelp.com Password: 2017Trojans!	For the administrator, as customer service emails from the app will be sent to this email. Email is also connected to the GitHub.

ENVIRONMENT SETUP + APP DEPLOYMENT

Java Servlet

Software Required - Eclipse Java EE IDE for Web Developers



How to Import Back End Project:

1. Install Eclipse Java EE IDE
2. Clone back end code from Github master repository
3. Open Eclipse
4. Import
5. Existing Projects Into Workspace
6. Select root directory of local GitHub
7. Finish

How to Export and Deploy Back End Project:

1. Open Eclipse
2. Export
3. Web
4. WAR file
5. Web project: Dev
6. Set WAR file destination
7. Finish
8. ssh root@159.203.67.188
9. Change directory to /opt/tomcat/webapps/
10. Remove Dev.war file
11. Log into Tomcat Web Application Manager
12. Upload WAR file to Deploy > WAR file to deploy
13. Deploy!

Use the following tutorial for additional reference:

<http://www.codejava.net/ides/eclipse/how-to-create-deploy-and-run-java-servlet-in-eclipse>

Android

Software Required - Android Studio

1. Install Android Studio
2. Clone Android code from Github master repository
3. Open existing project in Android Studio

iOS

Software Required - Xcode, CocoaPods

1. Install Xcode.
2. Install CocoaPods.
3. Clone iOS code from Github master repository
4. Do “pod install” in project directory.
5. Use “illegal.xcworkspace” to open ilegal project.

FEATURES BACKLOG + SUGGESTIONS FOR NEXT TEAM

#	Feature Update/Addition
1	Add functionality to delete uploaded PDFs from the administrator portal. Currently, in order to delete a PDF from the back-end, the administrator would need to physically remove it from the server's file system and clear its corresponding rows in the database.
2	Add functionality to allow users to delete PDFs from their saved list as well as edit/update PDFs they've previously filled (can use fields_filled table in database; currently unused).
3	Update the UI for the text-scraped PDF (screen which allows users to actually fill the PDF).
4	Add functionality to edit the list of civil codes for monthly notifications.
5	Rather than manually create each fillable PDF with Adobe Acrobat Pro DC, request access to unlocked fillable PDF forms from Court office (in place of the unlocked plain-text forms) -> these forms have already been generated by the Court and may be accessed here: http://www.courts.ca.gov/formnumber.htm
6	Add email verification to disallow generation of multiple spam accounts.
7	Store sensitive personal information using encryption (ex. Driver's License Number)
8	Make the mapping drag and drop tool more user friendly by clearly defining which autofill mapping fields on the right column are general and category specific and automatically mapping fields with the same name.
9	Reconcile the differences between the Android and iOS apps. Update iOS user interface to match the Android's.
10	Refactor code for optimized performance.

11	Thoroughly test and debug app (as this project has been developed over the course of 3 semesters, there are still outstanding bugs from each previous team). Test the apps on actual mobile devices and ensure UI/features translate to different devices appropriately (Samsung Galaxy s8, iPhone X, etc). For example, certain functionality cannot be tested on the emulator such as saving the PDF or sending it over AirDrop.
12	Get the iOS and Android apps approved on the corresponding App Stores.
13	Add payment infrastructure for donations if the application is non-profit or discuss with the stakeholder if the app should be paid.

CONTACT INFORMATION

Team Member	Email
Pooja Chebolu	pchebolu@usc.edu
Peter Lu	peterjlu@usc.edu
Sean Yuan	seanyuan@usc.edu
Jennifer Zheng	zhengjen@usc.edu



iLegalSelfHelp

Detailed Design Document

Version 1.0

1/17/2017

Stakeholder:Yadira Ojeda

Developers: Marek Foster, Curtis Johnson, Matt Rigdon, Himmat Singh

Table of Contents

High Level Requirements	(2)
Deliverables Schedule	(3)
User Interface and WireFrame Design	(4-6)
Backend Methods Added	(7-8)
Testing/Error Catching	(9-12)
1. Login Page Error Checking	
2. Account Info Error Checking	
3. Documents List Error Checking	
4. PDF Filler Error Checking	
Uploading PDF to App	(13-16)
1. Create PDF	(13-14)
2. Upload PDF	(15)
3. Database	(16)
Technology and Softwares Used	(17-18)
Web Server Full Stack	(19)
Accounts Information	(20)
iLegalSelfHelp Logo	(21)
Previous Documentation	(22-67)
BackEnd Logic & Method Implementation	

High Level Requirements

StakeHolder: Yadira Davis <yadirao40@yahoo.com>

Team Name: iLegalSelfHelp

Team Size: 3-4 people

Expectations: iOS and Android Development (Smartphone development)

Description:

(Continuing the) Creation of a the first non-profit self-help legal app on both Android and iOS platforms. The app will allow Angelenos to search, fill, and file court documents within Los Angeles right on their mobile phone. The app will assist underrepresented communities in LA County and offers the opportunity for legal learning.

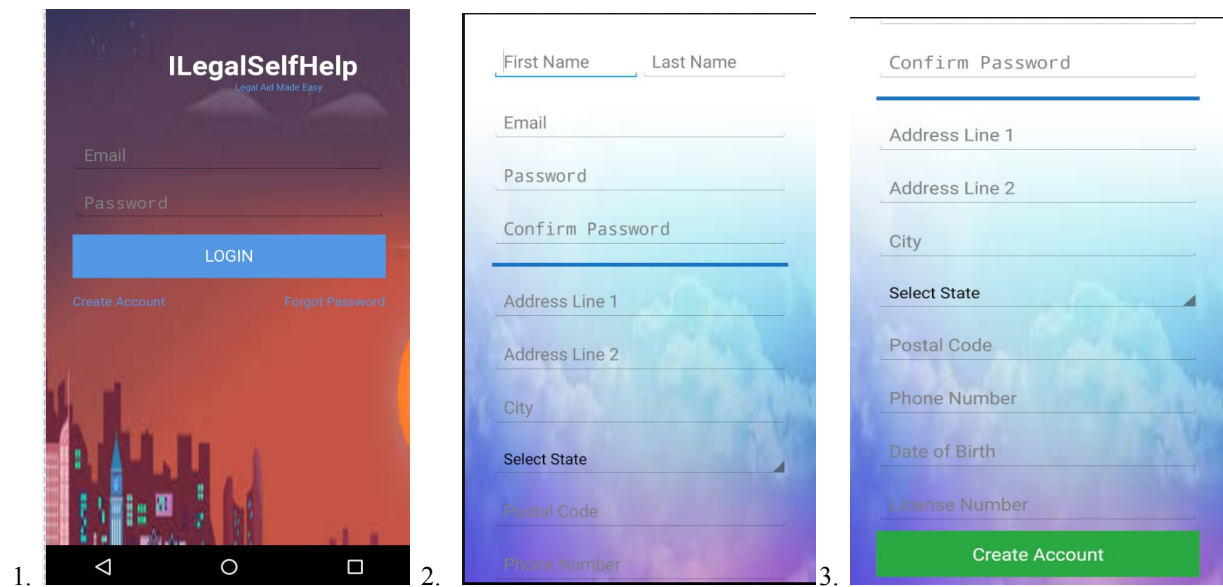
The app contains a large array of PDFs well-known and filled out in the LA county, each separated into Categories in-app for ease of access

Deliverable Schedule

Deliverable	Date	What we promise	What was delivered
1	1/17	Meeting/Initial Guidelines Set	More In-Depth Project Description/Contact info from last team since this is a continuation effort. Greater Understanding of the UI requirements. Great understanding for need of the project. Clearing up what has already been done and what needs to be done. Laying out the following meeting protocols/contacting freq
2	1/30	Code Base, architecture for the app- what technology we are going to be using, github setup, overhead things set out and modified and ready to use	Received code base from previous group. Set up github account and all groups are on it. Still missing some login information from the previous group, so we have to track that down. The first draft of the UI is drawn up and shown to Yadira, a more specific outline will be done after the meeting with her.
3	2/13	Layout for the specific UI, Mockups, restructuring of forms	Wireframe is more solidified, thought out, and screens are leading to one another. Passed by yadira and it's good to start designing in app.
4	2/27	Database up and running, and model design (Login, Database accepts new users and recognizes old users)	Database is connected to and a trial pdf has been uploaded to the adoption categories. New users are being accepted where before they weren't. UI bugs have arised, the flow isn't as clean as it was, must fix.
5	3/13	Complete app UI	UI is complete, bugs are fixed, some changes need to be made- such as the progress bar for filling out PDFs as Yadira did not like that. PDFs need to be uploaded completely. The prospect of a Logo had been introduced last week and now is being created, first iteration was not accepted by Yadira- it was too 3D
6	3/27	Debugging and Testing Flow of App	Logo has been created and approved by Yadira, flat black and white. The color scheme for the whole app matches and flows well with each other, both on IOs and Android. The skyline of the DTLA for the Login page is inspiration for color theme and Login page design. UI is up to what Yadira likes, it flows nice and Debugging process has begun.
7	4/10	Complete documentation and App with all PDF's and categories listed out	Uploaded as many PDFs as possible using adobe acrobat pro DC. Documentation cleaned up and sent to both Prof Miller and Yadira.

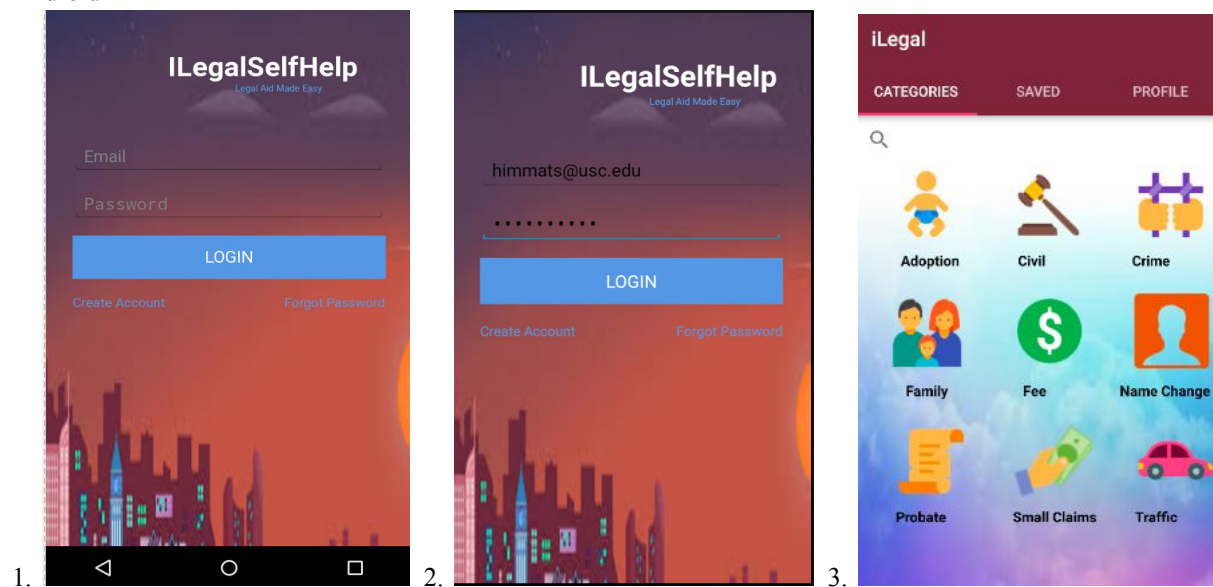
User Interface + WireFrame

Android and iOS



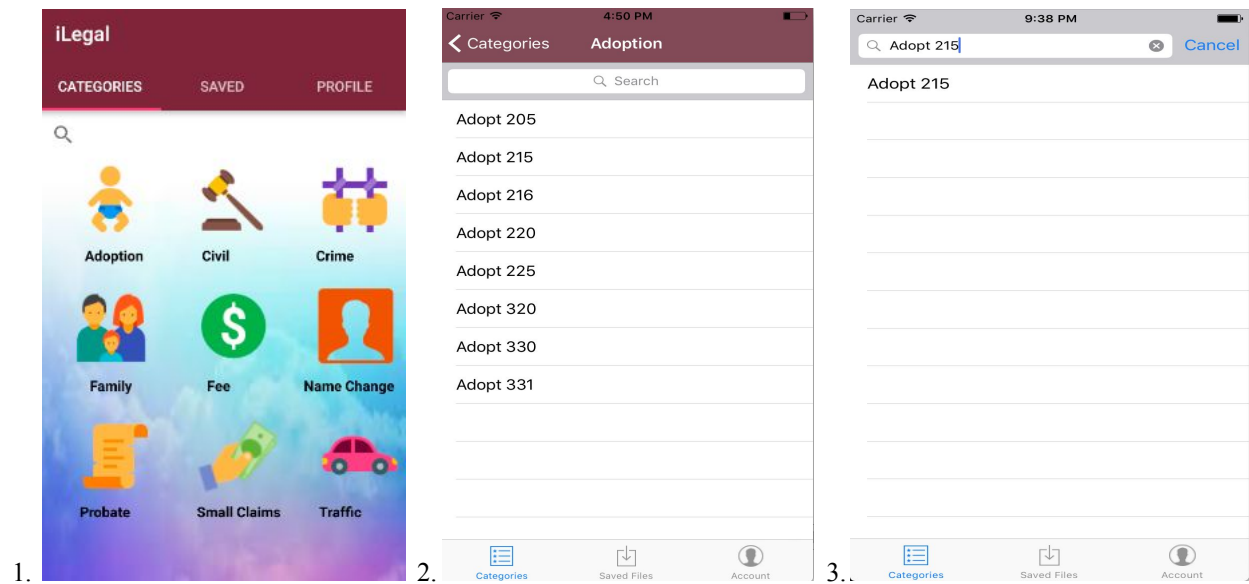
Flow: 1. Login Screen → Create Account Button Pressed → (2. & 3.) Create Account Screen

Android



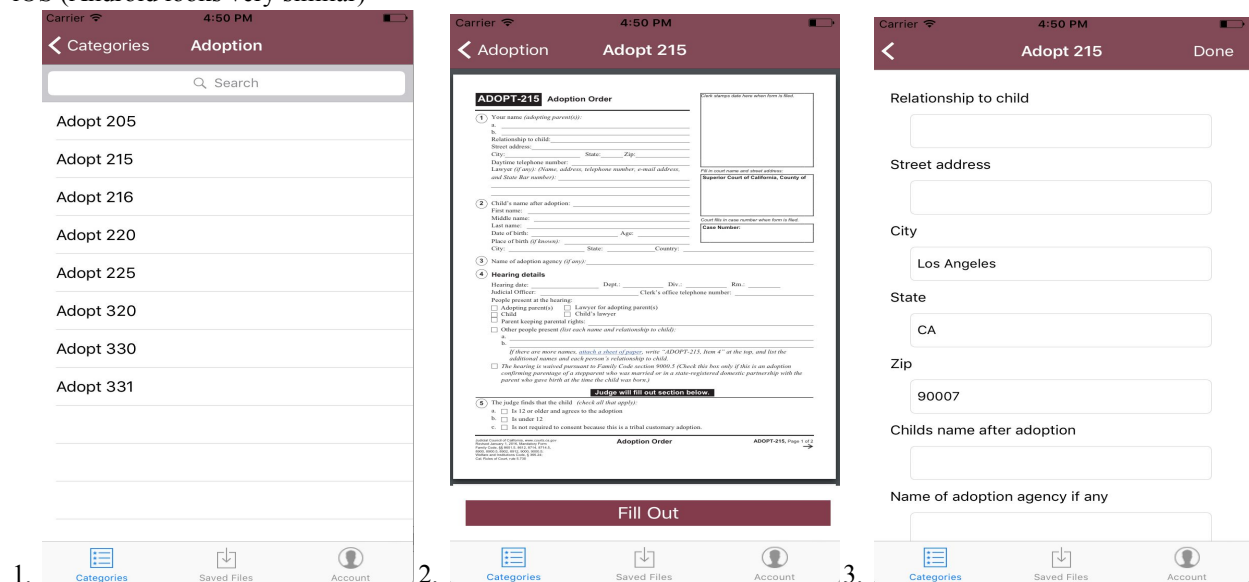
Flow: 1.Login Screen → Correct Info put in → 2.Email and Pswd Shown → Login Button Clicked→ 3. Document Categories Screen Shown

Android and iOS



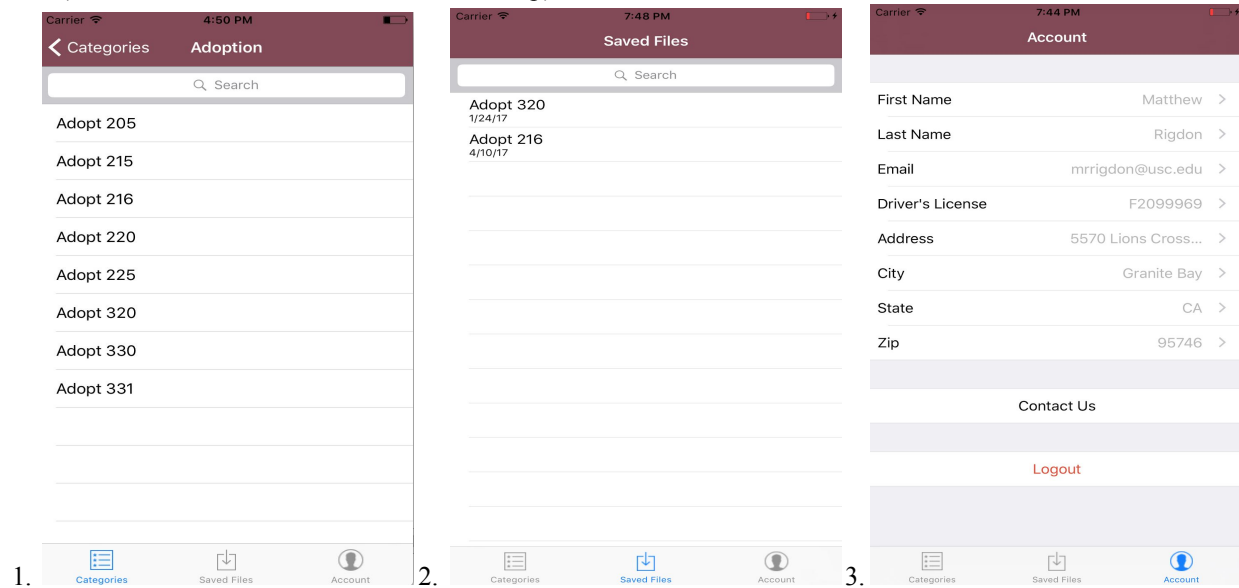
Flow: 1. Documents Grid Screen → Click Adoption Image → 2. List of Adoption Forms to Fill out → Search box at top searchable in case need a specific pdf → 3. Searched PDF

iOS (Android looks very similar)



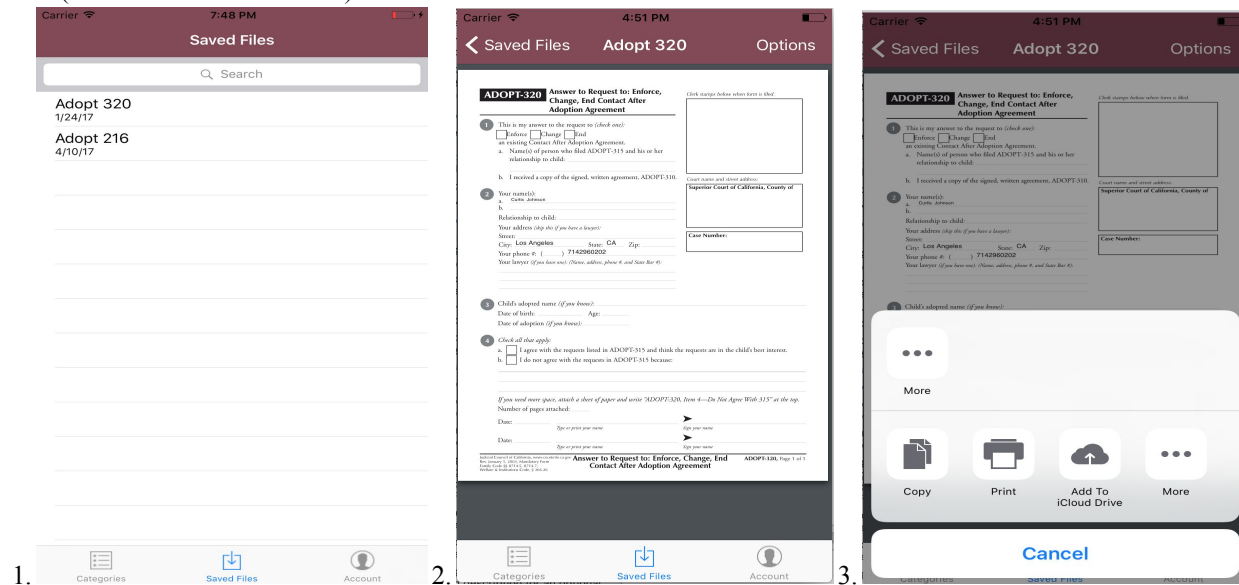
Flow: 1. List of Specific Forms → One of the Forms is clicked → 2. PDF viewer is shown of whole form → click on specific typable space → 3. Easy Access PDF filler brought up

iOS (*Android difference is Nav Bar is on the top)



Flow: 1. List of Specific Forms → “Saved Files” on bottom Nav Bar is Clicked → 2. Saved files are show → “Account” on bottom Nav Bar is Clicked → 3. Account Info is brought up

iOS (Android Flow is the same)



Flow: 1. List of Saved Forms→ Specific Form is Clicked → 2. PDF of form shown → “Options” on top right is Clicked → 3. Printable Access and Email Access are shown

iOS “Backend” Class

A wrapper class for all methods that hit the web server and some other utility functions.

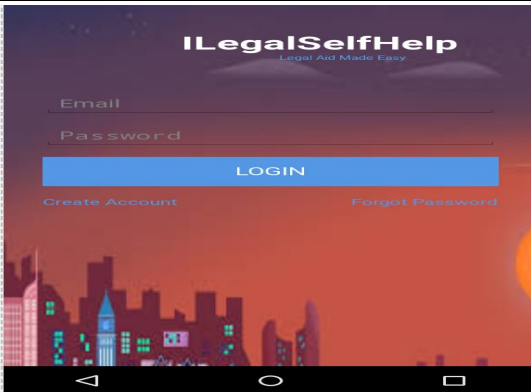
Methods:

- login
 - Description: used to log a user into the app
 - Endpoint: /SignIn
 - Params:
 - username (String) - the user's username
 - password (String) - the user's password
 - completion (Closure) - executed when the login is completed; passes back an optional error if request failed
 - Return Value: None
- saveUserLocal
 - Description: used to save the user's info locally to the iPhone
 - Endpoint: None
 - Params: None
 - Return Value: None
- clearUserLocal
 - Description: used to purge user's locally saved data
 - Endpoint: None
 - Params: None
 - Return Value: None
- getSavedFiles
 - Description: used to get all the user's saved files from the database
 - Endpoint: /ListPDF
 - Params:
 - completion (Closure) - executed when the files have been retrieved; passes back the saved files if request executed successfully
 - Return Value: None
- getCategories
 - Description: used to get the categories from a local json file
 - Endpoint: None
 - Params:
 - completion (Closure) - executed when the data has been retrieved; passes back the list of categories
 - Return Value: None
- getItems
 - Description: used to get all PDFs from a category
 - Endpoint: /ListPDF
 - Params:
 - category (String) - The category from which to get the PDFs
 - completion (Closure) - executed when the request has finished; passes back a list of the PDFs if successful

- Return Value: None
- updateUser
 - Description: used to update a field of the user
 - Endpoint: /UpdateUser
 - Params:
 - field (String) - the field to change (i.e. Email)
 - value (String) - the new value for the field to change (i.e. example@gmail.com)
 - completion (Closure) - executed when the request has finished; Passes back an optional error if the request failed
 - Return Value: None
- getPDF
 - Description: Used to get the parsed questions from a pdf from the database
 - Endpoint: /FillPDF
 - Params:
 - id (String) - The Id of the desired pdf
 - completion (Closure) - Executes when the request has finished; passes back a list of questions if the request succeeded, an error if it failed

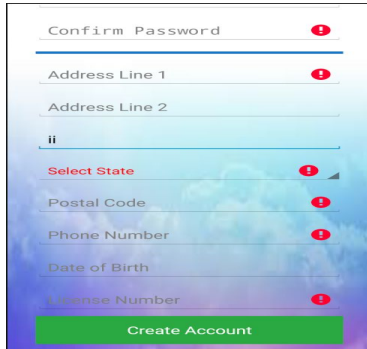
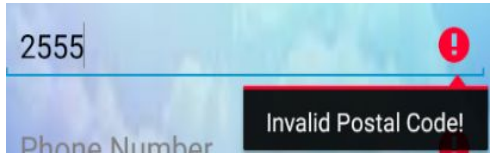
Error Testing

1. Login Window

Test #	01
Test Description	Error catching for login information
Steps to run test	<ol style="list-style-type: none"> 1. User name valid, password not 2. Username invalid, password valid 3. Does not enter anything in either field, can still click button 4. Enter multiple spaces for username 5. Enter multiple spaces for password 6. Username and Password exists, try to create account
Input	<ol style="list-style-type: none"> 1. Username: "user1" password: " " 2. Username: "user1" password: "apples" ('user1' already taken) 3. Textfields all: " " 4. Username: " " 5. Password: " " 6. Username: "user1" password: "apples" ('user1 already taken) 7. Username: "user1" password: "apples" 8. Username: "user1" password: "apples"
Expected Output	<ol style="list-style-type: none"> 1. Error: "Invalid Password" 2. Error: "Invalid Username" 3. Button still enabled 4. Error: "Invalid Username" 5. Error: "Invalid Password" 6. Error: "Username already taken" 7. Account info stored and game board choice GUI opens 8. Game board choice GUI opens
Screenshot	

2. Create Account Screen

Test #	02
Test Description	Error catching for putting in new account information

Steps to run test	<ol style="list-style-type: none"> 1. Create Button Clicked with no information 2. Invalid Postal Code 3. Invalid phone number 4. Invalid driver's license 5. Invalid email
Input	<ol style="list-style-type: none"> 1. All fields are empty, hit the create button 2. Input random numbers too short for postal code 3. Phone number too short 4. Incorrect Drivers License too short 5. Incorrect email, not correct server domain
Expected Output	<ol style="list-style-type: none"> 1. Red Flags (fig1) 2. Red Flag and Toast Message (fig2) 3. Red Flag and Toast Message 4. Red Flag and Toast Message 5. Red Flag and Toast Message
Screenshot	  <p>fig1</p> <p>fig2</p>

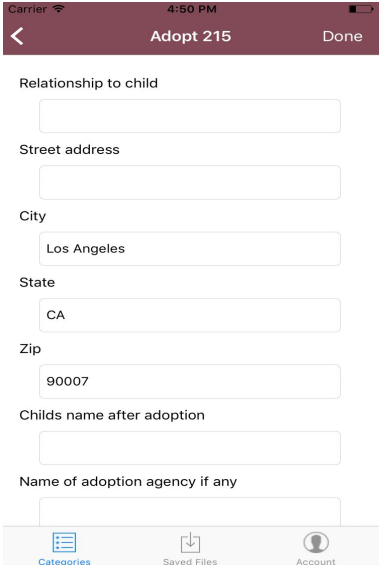
3. Documents Categories Screen

Test #	03
Test Description	Error catching for Documents Categories
Steps to run test	<ol style="list-style-type: none"> 1. Click on Icon 2. Invalid Web Request Call 3. Correct Web Request Call 4. Consistency to Categories
Input	<ol style="list-style-type: none"> 1. Click on specific icon 2. Click outside icon 3. Click on Adoption 4. Click on Nav Bar on bottom(for iOS) or top(Android) and return to doc list, should populate with categories again
Expected Output	<ol style="list-style-type: none"> 1. Listview page of documents shown 2. Nothing should happen 3. Adoption forms should populate screen from JSON request 4. Categories should populate

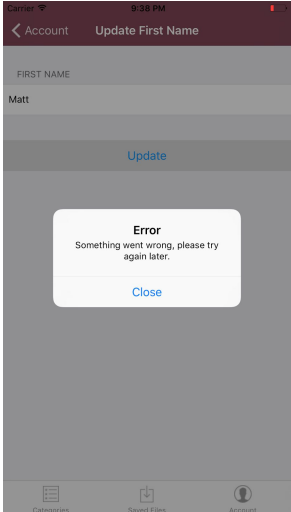
Screenshot	

4. PDF fillable Screen

Test #	04
Test Description	Error catching for pdf fillable Screen
Steps to run test	<ol style="list-style-type: none"> 1.Input Information into pdf filler 2.Invalid Web Request 3.Save Document 4.Save Document, navigate somewhere else, return to saved list 5.Open saved document
Input	<ol style="list-style-type: none"> 1. All fields are empty, click on field 2. Not connected to internet, click on pdf 3. Click on save pdf 4. Fill out doc, save, and navigate somewhere else in app, come back 5. Open saved doc
Expected Output	<ol style="list-style-type: none"> 1. Field should pop up or lead to another screen which allows it to be filled out 2. PDF filler should be blank or ask to connect to internet 3. PDF should now be accessible in the pdf view saved list 4. PDF persists in app 5. Information retained in the saved pdf version

Screenshot	 <p>The screenshot shows a mobile app interface titled 'Adopt 215'. The form includes fields for 'Relationship to child', 'Street address', 'City' (pre-filled with 'Los Angeles'), 'State' (pre-filled with 'CA'), 'Zip' (pre-filled with '90007'), 'Childs name after adoption', and 'Name of adoption agency if any'. A bottom navigation bar contains icons for 'Categories', 'Saved Files', and 'Account'.</p>
------------	--

5. Updating User Information

Test #	05
Test Description	Error catching for updating user information
Steps to run test	1. Remove Connection from internet and Press submit
Input	1. Enter new value into text field of update user info page, for any field a. Hit the submit button
Expected Output	1. An alert that says the request failed
Screenshot	 <p>The screenshot shows the 'Update First Name' screen in the app. The 'FIRST NAME' field contains the text 'Matt'. Below the field is an 'Update' button. An error dialog box is displayed in the center, stating 'Error: Something went wrong, please try again later.' with a 'Close' button. The bottom navigation bar is visible at the bottom.</p>

Creating PDF's

Before uploading forms to the app, pdf's must be properly formatted with fillable fields. We used Adobe Acrobat DC pro to do this. Below is the step-by-step approach:

1. Open pdf in adobe acrobat dc pro
2. Under tools > click prepare form
(this will automatically create most of the fields)
3. Go through each line editing/creating the appropriate field type and name
(the field name will be the label displayed in app)

Below is a screenshot of Adobe Acrobat DC pro preparing a form. Double clicking on a field will bring up the field's properties.

) This is my answer to the request to (check one):

☐ Enforce ☐ Change ☐ End

an existing Contact After Adoption Agreement.

a. Name(s) of person who filed ADOPT-315 and
relationship to child:

b. I received a copy of the signed, written agreement

) Your name(s):

a.

b.
Relationship to child:

Your address (skip this if you have a lawyer):

Street:

City: State: Zi

Your phone #: (Area Code) Phone Number

The three types of fields supported by the application are text fields, checkboxes, and radio buttons. There is also a specific syntax that must be followed when naming checkboxes and radio buttons.

By default the checkbox will appear below the field name on the app.	
Field Name: <i>Check here if you asked the court to waive your court fees for this case in the last six months</i>	<div>Check here if you asked the court to waive your court fees for this case in the last six months</div> <div><input type="checkbox"/></div>
If you want a custom answer next to the checkbox, then place a custom answer in brackets after field name.	

<p><i>Field Name: Has you financial situation improved since your last request to waive fees?[No]</i></p>	<p>Has your financial situation improved since your last Request to waive fees?</p> <p><input type="radio"/> No</p>
<p>To display multiple answers then simply use identical field names with different answers in the brackets. See screenshot above.</p>	
<p><i>Field Name 1: This is my answer to the request to (check one)[Enforce]</i> <i>Field Name 2: This is my answer to the request to (check one)[Change]</i> <i>Field Name 3: This is my answer to the request to (check one)[End]</i></p>	<p>This is my answer to the request to (check one)</p> <p><input type="radio"/> Enforce</p> <p><input type="radio"/> Change</p> <p><input type="radio"/> End</p>

Upload PDF to App

1. Front-end website for uploading pdf's directly to the app is located at:
159.203.67.188/login.html
2. Page will display a login button, which when pushed will prompt the user for a password (illegal).
3. Successfully logging in will take user's to the upload pdf dashboard
(159.203.67.188/upload.html).
4. To upload pdf's simply choose the correct pdf file, then carefully choose a name and category to use in the application and click submit.

Uploaded PDFs are stored under /var/www/html/pdfs/<category>/

The PDF Categories:

Adoption	Civil	Crime	Family	Fees
Name Change	Other	Probate	Small Claims	Traffic

Database

The database is a Mysql database called “capstonedb” The database has 6 tables.

1. **Documents:** This table has 4 columns.
 - a. Id - Auto-increments
 - b. FileName - String
 - c. FileURL - String with the folder and file the file is located at in the server.
 - d. Category - String corresponding the the folder name for that category and file.
2. **fields_filled**
 - a. Id- Auto-increments
 - b. Key - String (64)
 - c. Value - String (128)
3. **filled_pdfs**
 - a. UniqueID - Auto-increments
 - b. PDFTitle - String (128)
 - c. UniqueIDOfUser - Int
 - d. Filepath
 - e. Timestamp
4. **metatag**
 - a. Id
 - b. MetaTagName
 - c. UserID
5. **pdf_structure**
 - a. id
 - b. type
 - c. field_name
 - d. field_option
 - e. pdf_id
6. **User**
 - a. Id
 - b. FirstName
 - c. LastName
 - d. MiddleName
 - e. DOB
 - f. Address1
 - g. Address2
 - h. City
 - i. ZipCode
 - j. PhoneNumber
 - k. State
 - l. DLNumber
 - m. Email
 - n. Password

Technology Used

Tech: GitHub, Android SDK, iOS SDK, Apache Tomcat, Digital Ocean, Java Servlet, MySQL

Languages: XML, Java, HTML, Swift, Css, JavaScript

Github

Hosts the Android, iOS, and web server source code

Android SDK

Used to build the Android app

iOS SDK

Used to build the iOS app

Linux

Ubuntu was used as the server OS

Apache Tomcat

The framework used to run the web server

Java Servlet

The framework used to handle web requests

Digital Ocean

Hosts the Tomcat server

MySQL

The database for the platform

XML

Used to build the iOS and Android user interface layouts

Java

The language of which the Android and Servlet source code is built on

Swift

The language of which the iOS source code is built on

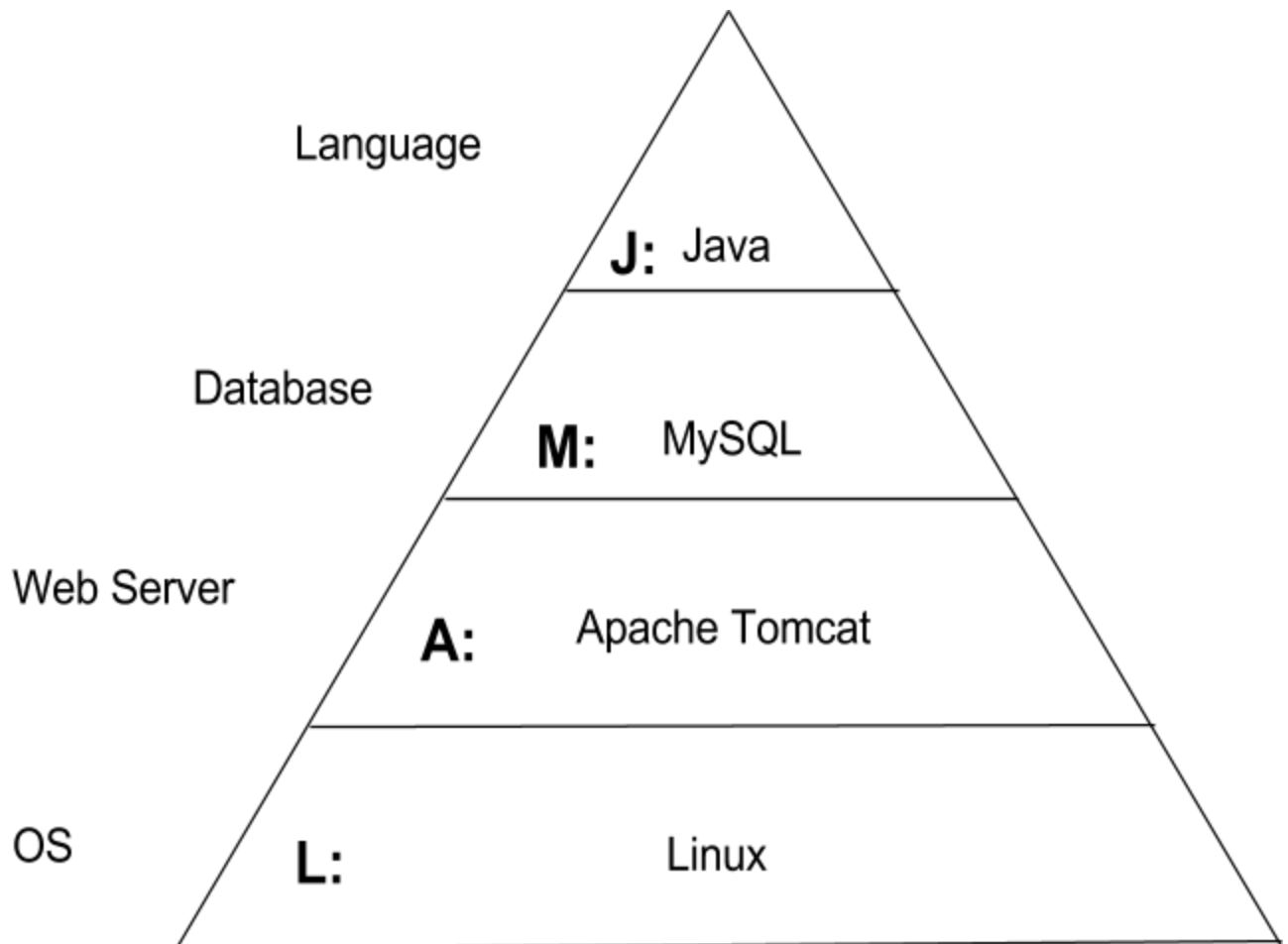
Cocoapods

The dependency manager for the iOS app source code

HTML/CSS/JavaScript

The languages of which the upload pdf and login web pages are built on

Web Server Full Stack



Accounts Information

	Link	Username	Password
Digital Ocean	https://cloud.digitalocean.com/login	Yadirao40@yahoo.com	2017Trojans
GoDaddy	https://sso.godaddy.com/?realm=idp&app=mya&path=&isc=gdbb3376	Yadirao40	2017Trojans
Admin Page (upload pdfs to app)	http://159.203.67.188/login.html		illegal
Apache Tomcat	http://159.203.67.188:8080/Dev/		
GitHub	https://github.com/iLegalSelfHelp	*invite only*	
Apple Dev Info		Yadirao40@yahoo.com	**My06171986
Group Member Info	himmats@usc.edu , mefoster@usc.edu , cbjohnso@usc.edu , mrrigdon@usc.edu		
Server Access		root	Trojans17
MYSQL	Database	root	Trojans17

Informational Videos

How to create a cloud server (droplet):

<https://www.youtube.com/watch?v=vqZ7eKM0WS8>

How to one click install LAMP & write to the URL that the server is connected to:

<https://www.youtube.com/watch?v=P8rqDzUFjGU>

How to Access MySQL server via a GUI (terminal):

<https://www.youtube.com/watch?v=4fcPxLrHH7o>

Digital-Ocean Update: walkthrough of the whole console

<https://www.youtube.com/watch?v=HWeva-PFyWg>

Logo Design



iLegalSelfHelp Logo. Self-Designed. Trademarked.

Document Category Widgets taken from: <https://icons8.com/web-app/new-icons/all>

Clouds Background Picture: <http://images-hd.net/single/1539082-calm-background.html>

LA Skyline Background Picture:

<http://www.vactualpapers.com/web/wallpapers/looking-over-the-city-abstract-hd-wallpaper/3840x2160.jpg>

iLegal Self Help

Legal Aid Made Easy

Hooman Zarrabi
Jordan Banafsheha
Grace (Yoo Jin) Lee
Nicky Guangorena

Table of Contents

Android	3
Classes and its Methods	3
UserSingleton	3
MainActivity	10
MainNavigationActivity	11
CategoriesFragment	12
DocsListviewActivity	13
PdfPreviewActivity	13
FillPdfActivity	14
CompletedPdfActivity	15
SavedPdfsListviewFragment	15
HashPassword	16
iOS	17
Classes and its Methods	17
User	17
Form	18
DropdownPickerView	18
CheckBox	19
CheckBoxGroup	19
PDFTextField	19
LogInViewController	20
ContinueAccountViewController	20
CreateAccountViewController	23
ForgotPasswordViewController	25
CategoriesViewController	26
DocumentsViewController	27
PreviewPDFViewController	29
EditPDFViewController	30
DonePDFViewController	32
SavedFilesViewController	33
ViewPDFViewController	35
Design Pattern	37
MVC (Model - View - Controller)	37
Singleton	38
Open-Source Framework	38

Alamofire	38
SwiftyJSON	38
CryptoSwift	39
PDFBox	39
Server/Database	39
Database setup	39
Server-Side Application	41
PDF Processing	41
Backend APIs	41
War Deployment	44
Front-End Website	44

Android

Classes and its Methods

UserSingleton

Extends Class: java.lang.Object

Description: Class that contains the user's basic information (address, birthday, name, email, phone number, etc.) to be utilized throughout the PDF forms. This is a singleton so it is meant to be the model that all classes get the logged in user's information from.

Member Variables:

Type	Name	Description
String	email	Email address of user
String	firstName	First name of user
String	lastName	Last name of user
String	address1	Address 1 of user
String	address2	Address 2 of user (Optional)
String	city	City of user
String	state	State of user
String	postalCode	Zip/Postal code of user
String	phoneNumber	Phone number of user
String	licenseNumber	Drivers License of user
String	dob	Birth date of user
String	ID	Unique Identifier of user
Static UserSingleton	user	Unique UserSingleton object to carry user information

Methods:

setAddress1

Description:

Sets address1 string variable

Parameters:

String address1

Returns:

N/A

setAddress2

Description:

Sets address2 string variable

Parameters:

String address2

Returns:

N/A

setCity

Description:

Sets city string variable

Parameters:

String city

Returns:

N/A

setDob

Description:

Sets dob String variable

Parameters:

String dob

Returns:

N/A

setEmail

Description:

Sets email string variable

Parameters:

String address1

Returns:

N/A

setFirstName**Description:**

Sets firstName string variable

Parameters:

String firstName

Returns:

N/A

setAddress1**Description:**

Sets address1 string variable

Parameters:

String address1

Returns:

N/A

setID**Description:**

Sets ID string variable

Parameters:

String ID

Returns:

N/A

setLastName

Description:

Sets lastName string variable

Parameters:

String lastName

Returns:

N/A

setLicenseNumber**Description:**

Sets licenseNumber string variable

Parameters:

String licenseNumber

Returns:

N/A

setPhoneNumber**Description:**

Sets phoneNumber string variable

Parameters:

String phoneNumber

Returns:

N/A

setPostalCode**Description:**

Sets postalCode string variable

Parameters:

String postalCode

Returns:

N/A

setState

Description:

Sets state string variable

Parameters:

String state

Returns:

N/A

getAddress1**Description:**

Gets address1 string variable

Parameters:

N/A

Returns:

address1

getAddress2**Description:**

Gets address2 string variable

Parameters:

N/A

Returns:

address2

getCity**Description:**

Gets city string variable

Parameters:

N/A

Returns:

city

getDob

Description:

Gets dob string variable

Parameters:

N/A

Returns:

dob

getEmail**Description:**

Gets email string variable

Parameters:

N/A

Returns:

email

setFirstName**Description:**

Gets firstName string variable

Parameters:

N/A

Returns:

firstName

getID**Description:**

Gets IDstring variable

Parameters:

N/A

Returns:

ID

getLastName

Description:

Gets lastName string variable

Parameters:

N/A

Returns:

lastName

getLicenseNumber**Description:**

Gets licenseNumber string variable

Parameters:

N/A

Returns:

licenseNumber

setPhoneNumber**Description:**

Gets phoneNumber string variable

Parameters:

N/A

Returns:

phoneNumber

getPostalCode**Description:**

Gets postCode string variable

Parameters:

N/A

Returns:

postalCode

getState

Description:

Gets state string variable

Parameters:

N/A

Returns:

state

MainActivity

Extends Class: android.app.Activity

Description: Activity where user logs in with an email and password.

Type	Name	Description
EditText	email	Field where users enters email address
EditText	password	Field where user enters password
Textview	createAccount	Text that forwards user to CreateAccountActivity
Textview	forgotPassword	Text that forwards user to ForgotPasswordActivity
Button	loginButton	Button that logs user in and forward user to MainNavigationActivity

isValidEmail

Description:

Verifies whether the entered email is correctly formatted

Parameters:

Charsequence target

Returns:

boolean

MainNavigationActivity

Extends Class: android.support.v7.app.AppCompatActivity

Implements: android.support.design.widget.NavigationView.OnNavigationItemSelectedListener

Description: Class that is responsible for the overarching navigation of the app. Implements navigation drawer that uses CategoriesFragment as well as SavedPdfsListViewFragment.

onBackPressed

Description:

Specifies that the navigation drawer will close if back is pressed. Otherwise app will navigate back.

Parameters:

N/A

Returns:

void

onCreateOptionsMenu

Description:

Inflates the mainNavigation resource file

Parameters:

Menu menu

Returns:

boolean

onOptionsItemSelected

Description:

Handle action bar item clicks here. The action bar will automatically handle clicks on the Home/Up button, so long as you specify a parent activity in AndroidManifest.xml.

Parameters:

MenuItem item

Returns:

boolean

onNavigationItemSelected

Description:

Handles the selection and displaying of navigation drawer items. Will also display respective fragment on click of navigation item

Parameters:

MenuItem item

Returns:

boolean

CategoriesFragment

Extends Class: android.support.v4.app.Fragment

Description: Fragment that will display the available categories of PDFs based on documents database.

Type	Name	Description
View	view	Used to inflate the Categories Fragment Layout
ListView	mainListView	ListView that displays the categories
ArrayAdapter	listAdapter	Adapter that connects the array of categories to the mainListView

DocsListViewActivity

Extends Class: android.support.v7.app.AppCompatActivity

Description: Activity that displays the respective documents of the category selected in the previous CategoriesFragment.

Type	Name	Description
ListView	mainListView	ListView that displays the documents
ArrayList<PdfInfo>	pdfs	Array list of PdfInfo objects.
PdfAdapter	pdfAdapter	Adapter that connects the array of PdfInfo objects to the mainListView

Inner Class:

PdfAdapter: class that uses PdfInfo object to set text of listview

PdfPreviewActivity

Extends Class: android.support.v7.app.AppCompatActivity

Description: Activity that displays the generic selected pdf in a webview. Here you can scroll through the pdf if you choose that you want to fill out the pdf you press the fill out button.

Type	Name	Description
WebView	pdfPreview	WebView that displays generic pdf
Button	editButton	Button that directs you to FillPdfActivity to fill respective PDF
PdfInfo	selectedPdf	Object that holds info of selected PDF

FillPdfActivity

Extends Class: android.support.v7.app.AppCompatActivity

Description: Displays fields of PDF natively on the app and allows you to fill in information. Once you fill your information you can then submit the information.

Type	Name	Description
Button	fillButton	Button that fills pdf with info and directs user to FilledPDFActivity
LinearLayout	ll	Linear layout of the Activity that holds dynamically created view
PdfInfo	selectedPdf	Object that holds info of selected PDF
JSONArray	fieldsArray	JSONArray that holds field info of PDF
ArrayList<Spinner>	spinnerArrayList	Arraylist that hold references to spinners created for pdf
ArrayList<Checkbox>	checkBoxArrayList	Arraylist that hold references to checkBoxes created for pdf
ArrayList<EditText>	editTextArrayList	Arraylist that hold references to editTexts created for pdf

CompletedPdfActivity

Extends Class: android.support.v7.app.AppCompatActivity

Description: Displays the filled out PDF in a webview and gives option to email it.

Type	Name	Description
Webview	filledPdfWebview	Webview that displays filled PDF
String	filledPdfUrl	String that holds URL of filled PDF
Button	emailButton	Button that allows user to email PDF
Button	doneButton	Button that returns user to MainNavigationActivity
File	outputDir	File object that holds path to create pdf file from server
PdfInfo	filledPDFInfo	Object that holds info of selected pdf

SavedPdfsListviewFragment

Extends Class: android.support.v7.app.AppCompatActivity

Description: Listview that displays the filled out pdfs. If you press on a name you are directed to the CompletedPdfActivity to see your PDF.

Type	Name	Description
View	view	Used to inflate the SavedPdfsListviewFragment Layout
ListView	mainListView	ListView that displays the saved PDFs
ArrayAdapter<String>	listAdapter	Adapter that connects the array of savedPDFs to the mainListView

HashPassword

Extends Class: java.lang.Object

Description: Static class that is used to encrypt passwords to SHA1.

encryptPassword

Description:

Static method that takes a plain text passwords and encrypts it to SHA1 to be stored in database.

Parameters:

String password

Returns:

String

byteToHex

Description:

Helper method used in the encryptPassword method that turns a byte array to a hashed string.

Parameters:

Final byte[] hash

Returns:

String

Classes and its Methods

User

Extends Class: NSObject

Description: Class of User containing all their basic information (address, birthday, name, email, phone number, etc.) to be utilized throughout the PDF forms.

Member Variables:

Type	Name	Description
String	email	Email address of user
String	password	SHA1-hashed password of user
String	firstName	First name of user
String	lastName	Last name of user
String	middleInitial	Middle initial of user (Optional)
String	addressOne	Address 1 of user
String	addressTwo	Address 2 of user (Optional)
String	city	City of user
String	state	State of user
String	zipcode	Zip/Postal code of user
String	phone	Phone number of user
String	license	Drivers License of user
Bool	active	Status whether the user has logged in or not
String	DOB	Birth date of user
String	id	Unique Identifier of user
Form []	myFiles	List of saved files by the user

Methods:

setBirthday
Description: Composes a string in format of “YYYY-MM-DD” for user birthday
Parameters: month - birth month of user day - birth day of user year - birth year of user
Returns: N/A

Form

Extends Class: NSObject

Description: Class storing the location, name, and unique ID for each PDF form

Member Variables:

Type	Name	Description
String	title	Title of the PDF form
String	subtitle	Description of the PDF form
String	id	Unique form ID of the form
String	location	Location of the form in the server

Methods: N/A

DropDownPickerView

Extends Class: UIPickerView

Description: UIPickerView with UITextField attached to visualize the chosen value on UI and the associated question.

Member Variables:

Type	Name	Description
UITextField	dataTF	Text Field where the value selected will be populated
String	title	Title of the question
Int	index	Index of question in the form

Methods: N/A

CheckBox

Extends Class: UISwitch

Description: UISwitch represented with its other check box options of the associated question as well as the string of check box option itself.

Member Variables:

Type	Name	Description
String	label	The title of the option
Array<CheckBox>	alternateOptions	Array of other CheckBox objects associated with the same question

Methods: N/A

CheckBoxGroup

Extends Class: NSObject

Description: Class representing the question requiring check box options, including the associated options, string of the question itself, and the index of the question on the form.

Member Variables:

Type	Name	Description
CheckBox []	buttons	Array of all CheckBox objects associated with the question
String	title	Title of the question
Int	index	Index of question on the form

Methods: N/A

PDFTextField

Extends Class: UITextField

Description: Class extending UITextField to include the string of the question..

Member Variables:

Type	Name	Description
String	title	Title of the question
Int	index	Index of question on the form

Methods: N/A

LoginViewController

Extends Class: UIViewController

Description: Main login page (start page) of the application. Handles login GET request.

Member Variables:

Type	Name	Description
UITextField	usernameTF	Text field of username
UITextField	passwordTF	Text field of password
UIButton	loginButton	Button that will trigger GET request for login

Methods:

loginButtonClicked
<p>Description: Triggers HTTP GET request to verify that user account with entered username and password exists in the server.</p> <p>Parameters: sender - UIButton that triggered the action (loginButton in this scenario)</p> <p>Returns: If the user account exists, app will proceed to CategoriesViewController. Otherwise, UIAlertController will pop up.</p>

ContinueAccountViewController

Extends Class: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate

Description: Second page of “Create Account,” prompting user for address, phone number, driver’s license, and etc.), also triggers HTTP GET request to create user account in server.

Member Variables:

Type	Name	Description
UITextField	addressOneTF	Text field requiring input of user’s address 1
UITextField	addressTwoTF	Text field requiring input of user’s address 2
UITextField	cityTF	Text field requiring input of user’s city
UITextField	stateTF	Text field requiring input of user’s state
UITextField	zipcodeTF	Text field requiring input of user’s zip/postal code

UITextField	phoneTF	Text field requiring input of user's phone number
UITextField	licenseTF	Text field requiring input of user's driver's license
UITextField	birthdayMTF	Text field requiring input of user's birthday month
UITextField	birthdayDTF	Text field requiring input of user's birthday day
UITextField	birthdayYTF	Text field requiring input of user's birthday year
UIButton	createButton	Button to finalize information and create user account
UILabel	addressLabel	Alert message displayed if addressTF is empty
UILabel	cityLabel	Alert message displayed if cityTF is empty
UILabel	stateLabel	Alert message displayed if stateTF is empty
UILabel	zipcodeLabel	Alert message displayed if zipcodeTF is empty
UILabel	phoneLabel	Alert message displayed if phoneTF is empty
UILabel	licenseLabel	Alert message displayed if licenseTF is empty
UILabel	birthdayLabel	Alert message displayed if birthdayMTF, birthdayDTF, birthdayYTF is empty

Methods:

createAccount
<p>Description: Upon click of "Create Account", check if all required fields have been populated. If populated, then trigger HTTP POST request to create account in the server. Otherwise, display error message.</p> <p>Parameters: sender - UIButton clicked to trigger IBAction (Download Button)</p> <p>Returns: N/A</p>

donePicker
<p>Description: Notifies the receiver that it has been asked to relinquish its status as first responder in its window.</p> <p>Parameters: N/A</p> <p>Returns:</p>

N/A

updateLabel

Description:

Display labels, error messages, when user fails to fill out any of the required fields.

Parameters:

sender - UIButton clicked to trigger IBAction (Create Account Button)

Returns:

N/A

pickerView

Description:

Returns the size of a row for a component.

Parameters:

pickerView - UIPickerView in scope

component - options in pickerView Data Source

Returns:

The count of options in pickerView

pickerView

Description:

Returns the view used by the picker view for a given row and component.

Parameters:

pickerView - UIPickerView in scope

row - row selected by the user

component - options in pickerView Data Source

Returns:

N/A

pickerView

Description:

Returns the value of the selected row in a given component.

Parameters:

pickerView - UIPickerView in scope

row - row selected by the user

component - options in pickerView Data Source

Returns:

The string value of option selected in pickerView

numberOfComponents
Description: Returns the number of components for the picker view. Parameters: pickerView - UIPickerView in scope Returns: The count of data source in pickerView

CreateAccountViewController

Extends Class: UIViewController, UITextFieldDelegate

Description: First page of “Create Account,” prompting user for email, password,

Member Variables:

Type	Name	Description
UITextField	firstNameTF	Text field requiring input of user's first name
UITextField	middleInitialTF	Text field requiring input of user's middle initial
UITextField	lastNameTF	Text field requiring input of user's last name
UITextField	emailTF	Text field requiring input of user's email address
UITextField	passwordTF	Text field requiring input of user's password
UITextField	confirmPasswordTF	Text field requiring input of user's password to confirm
UIButton	continueButton	Button to move to second page of create account
UILabel	pwReqLabel	Alert message displayed if passwordTF is either empty or less than six characters
UILabel	pwMatchLabel	Alert message displayed if passwordTF and confirmPasswordTF do not match
UILabel	firstNameLabel	Alert message displayed if firstNameTF is empty
UILabel	lastNameLabel	Alert message displayed if lastNameTF is empty
UILabel	emailLabel	Alert message displayed if emailTF is empty or not a valid email address

Methods:

textField

Description:

Function to only allow maximum of one character in Middle Initial

Parameters:

textField - UITextField with restricted character count

range - range of characters to be replaced

string - string entered in UITextField

Returns:

Returns true if middleInitialTF's string is shorter or equal to size of one, return false otherwise.

nextButtonClicked

Description:

Trigger continueAccount() when Next Bar Button is clicked

Parameters:

sender - UIButton clicked to trigger IBAction

Returns:

N/A

continueButtonClicked

Description:

Trigger continueAccount() when Continue Button is clicked

Parameters:

sender - UIButton clicked to trigger IBAction

Returns:

N/A

updateLabel

Description:

Display UILabels (error messages) when user fails to fill out any of the required fields.

Parameters:

sender - UIButton clicked to trigger IBAction

Returns:

N/A

validEmail

Description:

Check if input in emailTF is a valid form of email address

Parameters:

str - string of email

Returns:

Returns true if str is a valid email. Returns false otherwise.

continueAccount**Description:**

Check if all required fields have been populated by the user and segue to second page of Create Account

Parameters:

N/A

Returns:

N/A

ForgotPasswordViewController

Extends Class: UIViewController, MFMailComposeViewControllerDelegate

Description: Extension of view controller that allows user to input their first and last name along with the email address associated with the account to reset the password.

Member Variables:

Type	Name	Description
UITextField	firstNameTF	Text field requiring input of user's first name
UITextField	lastNameTF	Text field requiring input of user's last name
UITextField	emailTF	Text field requiring input of user's email address
UIButton	recoverPasswordButton	Button to move to reset password after populating all required input (first name, last name, email address)

Methods:**recoverPasswordButtonClicked****Description:**

Trigger HTTP GET request to check if there is an associated account with the first name, last name, and email address.

Parameters:

sender - UIButton clicked to trigger IBAction (Recover Password Button)

Returns:

N/A

CategoriesViewController

Extends Class: UITableViewController

Description: View controller that displays all categories of the form in a Table View

Member Variables:

Type	Name	Description
String []	categoriesList	List of all categories from the server
String []	filteredCategories	List of categories of search text from the full list of categories
UISearchController	searchController	Search bar for filtering categories

Methods:

tableView
<p>Description: Returns the count of categories, filtered or full list</p> <p>Parameters: tableView - UITableView in scope section - number of rows in section</p> <p>Returns: The count of categories in categoriesList or filteredCategories if search term exists</p>

tableView
<p>Description: Retrieve the table view cell of category selected</p> <p>Parameters: tableView - UITableView in scope indexPath - the index of selected cell in tableView</p> <p>Returns: UITableViewCell of selected cell</p>

tableView
<p>Description: Trigger segue on click of a table view cell with associated category</p> <p>Parameters: tableView - UITableView in scope indexPath - the index of selected cell in tableView</p>

Returns:
N/A

filteredContentForSearchText

Description:
Reload Table View based on the user-input of search text and filtered categories

Parameters:
searchText - string of search term to filter for
indexPath - the index of selected cell in tableView

Returns:
N/A

updateSearchResults

Description:
Trigger filtering of categories when input is typed in search bar

Parameters:
searchController - Search bar with entered search text on top of the table view

Returns:
N/A

DocumentsViewController

Extends Class: UITableViewController

Description: View list of all forms under the associated category clicked on previous CategoriesViewController

Member Variables:

Type	Name	Description
Form []	formList	List of all documents under the category from the server
Form []	filteredForms	List of documents of search text filtered from the full list of documents
UISearchController	searchController	Search bar for filtering documents

Methods:

tableView

Description:

Returns the count of categories, filtered or full list

Parameters:

tableView - UITableView in scope
section - number of rows in section

Returns:

The count of categories in formList or filteredForms if search term exists

tableView

Description:

Retrieve the table view cell of form selected

Parameters:

tableView - UITableView in scope
indexPath - the index of selected cell in tableView

Returns:

UITableViewCell of selected cell

tableView

Description:

Trigger segue on click of a table view cell with associated form

Parameters:

tableView - UITableView in scope
indexPath - the index of selected cell in tableView

Returns:

N/A

filteredContentForSearchText

Description:

Reload Table View based on the user-input of search text and filtered forms

Parameters:

searchText - string of search term to filter for
indexPath - the index of selected cell in tableView

Returns:

N/A

updateSearchResults

Description:

Trigger filtering of forms when input is typed in search bar

Parameters:

searchController - Search bar with entered search text on top of the table view

Returns:
N/A

populateForms

Description:
Trigger HTTP GET request to retrieve all forms under previously clicked category in CategoriesViewController

Parameters:
N/A

Returns:
N/A

PreviewPDFViewController

Extends Class: UIViewController

Description: Preview of the selected file from DocumentsViewController to display the PDF.

Member Variables:

Type	Name	Description
UIWebView	pdfWebView	Web View to display the document
UIButton	fillOutButton	Button clicked to proceed filling out the form
EditPDFViewController	destination	View Controller instantiated when clicked on "fillOutButton"
Form	currentForm	Current form clicked on DocumentsViewController to be displayed and filled out

Methods:

fillOutButtonPressed

Description:
Click button to proceed filling out the forms, triggering segue to EditPDFViewController

Parameters:
sender - UIButton clicked to trigger IBAction (fillOutButton)

Returns:
N/A

EditPDFViewController

Extends Class: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate

Description: View Controller with the form's parsing of fillable fields. All

Member Variables:

Type	Name	Description
Form	currentForm	Current form clicked on DocumentsViewController to be displayed and filled out
JSON	json	JSON of all parsable fields returned from the server
JSON	resultJSON	Updated JSON with all parsable fields and entered values
PDFTextField []	TFArray	Array of all "Tx" questions
YesNoButton []	YesNoArray	Array of all "yesNo" questions
CheckBoxGroup []	OptionArray	Array of all "Btn" questions
DropDownPickerView []	ChArray	Array of all "Ch" questions
Dictionary<Int,Array<String>>	ChData	Dictionary of all "Ch" questions with its associated options
Dictionary<Int,UITextField>	ChTF	Array of all "Tx" questions
Int	PVTAG	Incrementing UIPickerView tag index with each "Ch" question parsed
UIScrollView	scrollView	Main scroll view to hold all dynamically allocated fillable fields from the server

Methods:

donePressed
<p>Description: Send back the parsed fields and entered values back to the server through HTTP POST request</p> <p>Parameters: sender - UIButton clicked to trigger IBAction (Done Button)</p> <p>Returns: N/A</p>

parseJSON

Description:

Trigger HTTP GET request to receive parsed fillable fields in the form from the server, and dynamically allocate its appropriate UI objects to allow users to edit the fields.

Parameters:

N/A

Returns:

N/A

updateJSON

Description:

Updates JSON with new values entered by the user.

Parameters:

N/A

Returns:

N/A

pickerView

Description:

Returns the size of a row for a component.

Parameters:

pickerView - UIPickerView in scope

component - options in pickerView Data Source

Returns:

The count of options in pickerView

pickerView

Description:

Returns the view used by the picker view for a given row and component.

Parameters:

pickerView - UIPickerView in scope

row - row selected by the user

component - options in pickerView Data Source

Returns:

N/A

pickerView

Description:

Returns the value of the selected row in a given component.

Parameters:

pickerView - UIPickerView in scope

row - row selected by the user

component - options in pickerView Data Source

Returns:

The string value of option selected in pickerView

numberOfComponents**Description:**

Returns the number of components for the picker view.

Parameters:

pickerView - UIPickerView in scope

Returns:

The count of data source in pickerView

DonePDFViewController

Extends Class: UIViewController

Description:

Member Variables:

Type	Name	Description
UIWebView	pdfWebView	Web View to display the form
Form	currentForm	Current form to be displayed and filled out
String	fileURL	HTTP address to the new file

Methods:

savePressed**Description:**

Trigger UIAlertController to prompt user to either save or cancel save for the filled out form

Parameters:

sender - UIButton clicked to trigger IBAction (Save Bar Button)

Returns:

N/A

handleSave

Description:

Save the document to the server and display UIAlertController to notify that document has been saved

Parameters:

sender - UIButton clicked to trigger IBAction (Save Button)

Returns:

N/A

cancelSave**Description:**

Cancel saving of the filled form

Parameters:

sender - UIButton clicked to trigger IBAction (Cancel Button)

Returns:

N/A

SavedFilesViewController

Extends Class: UITableViewController, UITabBarControllerDelegate

Description: View controller displaying user's saved files in a Table View

Member Variables:

Type	Name	Description
Form []	filteredSavedFiles	List of user's saved files per search text entered
IndexPath	deleteFileIndex	Index of the file to be deleted
UISearchController	searchController	Search bar for filtering user's saved files

Methods:**tableView****Description:**

Returns the count of categories, filtered or full list

Parameters:

tableView - UITableView in scope
section - number of rows in section

Returns:

The count of categories in formList or filteredForms if search term exists

tableView

Description:

Retrieve the table view cell of form selected

Parameters:

tableView - UITableView in scope

indexPath - the index of selected cell in tableView

Returns:

UITableViewCell of selected cell

tableView

Description:

Trigger segue on click of a table view cell with associated form

Parameters:

tableView - UITableView in scope

indexPath - the index of selected cell in tableView

Returns:

N/A

filteredContentForSearchText

Description:

Reload Table View based on the user-input of search text and filtered forms

Parameters:

searchText - string of search term to filter for

indexPath - the index of selected cell in tableView

Returns:

N/A

updateSearchResults

Description:

Trigger filtering of forms when input is typed in search bar

Parameters:

searchController - Search bar with entered search text on top of the table view

Returns:

N/A

confirmDelete

Description:

Instantiate UIAlertController to confirm deletion of user's saved file

Parameters:

fileToDelete - name of the file to be deleted from user's saved files

Returns:

N/A

handleDelete**Description:**

Delete the specified saved file of the user from the server

Parameters:

alertAction - UIAlertAction from confirmDelete to handle deletion of user's saved file

Returns:

N/A

cancelDelete**Description:**

Cancel deletion of the specified saved file of the user

Parameters:

alertAction - UIAlertAction from confirmDelete to handle deletion of user's saved file

Returns:

N/A

tabBarController**Description:**

Refresh user's saved files each instance "Saved Files" is clicked on the interface

Parameters:

tabBarController - parent view controller that holds different tabs

viewController

Returns:

N/A

ViewPDFViewController

Extends Class: UIViewController, MFMailComposeViewControllerDelegate

Description: View Controller to display user's previously saved files from the selected in SavedFilesViewControllers

Member Variables:

Type	Name	Description
------	------	-------------

UIWebView	webView	Web View to display the filled out form
Form	currentForm	Current form selected from SavedFilesViewController
Data	formData	Current form saved in Data variable for email and share functionality

Methods:

optionsPressed

Description:

Trigger UIAlertController to prompt user for either download or email options

Parameters:

sender - UIButton clicked to trigger IBAction (Options Bar Button)

Returns:

N/A

downloadPressed

Description:

Perform download of the form when "Download" UIAlertAction is clicked

Parameters:

alertAction - UIAlertAction clicked to trigger IBAction (Download Button)

Returns:

N/A

emailPressed

Description:

Instantiate MailComposeViewController when "Email" UIAlertAction is clicked

Parameters:

alertAction - UIAlertAction clicked to trigger IBAction (Download Button)

Returns:

N/A

configuredMailComposeViewController

Description:

Instantiate MFMailComposeViewController with specified subject, recipient and file attached

Parameters:

N/A

Returns:

N/A

showSendEmailErrorAlert
Description: Display UIAlertController if fails to properly instantiate MFMailComposeViewController
Parameters: N/A
Returns: N/A

mailComposeController
Description: Dismiss MFMailComposeViewController
Parameters: controller - MFMailComposeViewController in view result - result of MFMailCompose error - error that triggered failure of MFMailCompose
Returns: N/A

Design Pattern

MVC (Model - View - Controller)

The Model-View-Controller (MVC) design pattern assigns objects in an application one of three roles: model, view, or controller. The pattern defines not only the roles objects play in the application, it defines the way objects communicate with each other. Each of the three types of objects is separated from the others by abstract boundaries and communicates with objects of the other types across those boundaries. Model objects encapsulate the data specific to an application and define the logic and computation that manipulate and process that data. A view object is an object in an application that users can see. A controller object acts as an intermediary between one or more of an application's view objects and one or more of its model objects.

Benefits of MVC design pattern:

- Many objects in these applications tend to be more reusable, and their interfaces tend to be better defined.

- Applications having an MVC design are also more easily extensible than other applications.
- Moreover, many Cocoa technologies and architectures are based on MVC and require that your custom objects play one of the MVC roles.

Model: Form, CheckBox, User, PDFTextField

View: UITextField, UIView, UIScrollView, UITableView, UIButton, UISwitch, UIPickerView

Controller: UIViewController, UITableViewController, UITabBarController, UINavigationController

Singleton

Singleton is where only one instance of the class only exists for the current process - the instance is shared across the entire application. Singleton eases the way to share data and common methods across the entire application, returning the same instance no matter how many times an application request it. In scope of this application, singleton was utilized to extend the instance of current user across the entire scope of the application without duplicating user's data multiple times.

Implementation of Singleton: User

Open-Source Framework

Alamofire

Alamofire is a Swift-based HTTP networking library for iOS and Mac OS X. It provides an elegant interface on top of Apple's Foundation networking stack that simplifies a number of common networking tasks. Alamofire provides chainable response/request methods, JSON parameter and response serialization, authentication, and many other features. Within the scope of this project, Alamofire was used to allow simple, condensed GET/POST request to the server. More information on Alamofire can be found here: <https://github.com/Alamofire/Alamofire>

Implementing Classes: ContinueAccountViewController.swift, CategoriesViewController.swift, DocumentsViewController.swift, EditPDFViewController.swift, LoginViewController.swift, ViewPDFViewController.swift, SavedFilesViewController.swift

SwiftyJSON

SwiftJSON is a Swift-based JSON handling library for iOS and Mac OS X. It allows easier JSON data handling in Swift. Swift is very strict about types. But although explicit typing is good for saving us from mistakes, it becomes painful when dealing with JSON and other areas that are, by nature, implicit about types. More information can be found here:

<https://github.com/SwiftyJSON/SwiftyJSON>

Implementing Classes: ContinueAccountViewController.swift, CategoriesViewController.swift, DocumentsViewController.swift, EditPDFViewController.swift, LoginViewController.swift, ViewPDFViewController.swift, SavedFilesViewController.swift

CryptoSwift

CryptoSwift is a growing collection of standard and secure cryptographic algorithms implemented in Swift. CryptoSwift currently supports hash, CRC, cipher, message authenticators, cipher block mode, password-based key derivation function, and data padding. In scope of this application, only SHA1 Hash method was utilized. More information can be found here: <https://github.com/krzyzanowskim/CryptoSwift>

Implementing Classes: LoginViewController.swift, CreateAccountViewController.swift

PDFBox

The Apache PDFBox® library is an open source Java tool for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Apache PDFBox also includes several command line utilities. Apache PDFBox is published under the Apache License v2.0: <https://pdfbox.apache.org/>

Server/Database

The Server had the LAMP stack setup, however the backend server application is in Java, not PHP. There is a tomcat apache server setup.

Database setup

The database is a Mysql database. The database is called “capstonedb”. This database has 6 tables.

1. documents: This table has 4 columns.

- a. Id - Auto-increments
 - b. FileName - String
 - c. FileURL - String with the folder and file the file is located at in the server.
 - d. Category - String corresponding the the folder name for that category and file.
- 2. fields_filled
 - a. Id- Auto-increments
 - b. Key - String (64)
 - c. Value - String (128)
- 3. filled_pdfs
 - a. UniqueID - Auto-increments
 - b. PDFTitle - String (128)
 - c. UniqueIDOfUser - Int
 - d. Filepath
 - e. Timestamp
- 4. metatag
 - a. Id
 - b. MetaTagName
 - c. UserID
- 5. pdf_structure
 - a. id
 - b. type
 - c. field_name
 - d. field_option
 - e. pdf_id
- 6. User
 - a. Id
 - b. FirstName
 - c. LastName
 - d. MiddleName
 - e. DOB
 - f. Address1
 - g. Address2
 - h. City
 - i. ZipCode
 - j. PhoneNumber
 - k. State
 - l. DLNumber
 - m. Email
 - n. Password

Server-Side Application

PDF Processing

pdfToText

Description:

Strips a PDF and makes the text accessible.

Parameters:

String - Filename

Returns:

String - all of the text from the PDF

listFields

Description:

Lists all of the fields in the PDF.

Parameters:

PDDocument - the document/PDF

Returns:

N/A

In the main method there are a few main sections to the code:

1. Loading the file
2. Connecting to the server and SQL database
3. Getting the Fields
4. Adding them to the Database

Backend APIs

Endpoints Doc

BASEURL = <http://159.203.67.188:8080/Dev>

1. BASEURL/SignUp
 - a. GET requests
 - i. Sent to Server:
 1. FirstName

2. LastName
 3. MiddleName
 4. DOB
 5. Address1
 6. Address2
 7. City
 8. PostalCode
 9. Phonenumber
 10. State
 11. LicenseNumber
 12. EmailAddress
 13. Password
 - ii. Recieved from Server:
 1. Message
 2. Success
2. BASEURL/SignIn
- a. GET requests
 - i. Sent to Server
 1. Username
 2. Password
 - ii. Recieved From Server
 1. Message
 2. Password
 3. EmailAddress
 4. LicenseNumber
 5. State
 6. PhoneNumber
 7. PostalCode
 8. City
 9. Address2
 10. Address1
 11. DOB
 12. MiddleName
 13. LastName
 14. FirstName
 15. Success
3. BASEURL/ListPDF
- a. GET Requests
 - i. Sent to Server
 1. Get categories
 - a. Type = (1)
 2. Get documents
 - a. Type = (2)

- b. Category
 - 3. Get filled_pdfs
 - a. Type = (4)
 - b. UserUniqueID
 - ii. Returned from Server
 - 1. Categories
 - a. Message
 - b. Success
 - c. Categories - Array of strings
 - 2. Documents
 - a. Message
 - b. PDFS - Array of (Array of(String))
 - c. Success
 - 3. Filled PDFs
 - a. Message
 - b. UserDocs - Aof(Aof(String))
 - c. Success
- 4. BASEURL/FinPDF
 - a. POST data to create a PDF.
 - i. Sent to Server
 - 1. USERID - for DB after pdf is filled.
 - 2. PDFID - id of pdf for getting pdf name and location.
 - 3. pdfJsonResults - JSON object of fields and values.
 - ii. Returned from Server
 - 1. FileURL - the location the filled PDF is stored on the server.
- 5. BASEURL/FillPDF
 - a. GET fields data for a PDF
 - i. Sent to Server
 - 1. PdfID - Id for the pdf to get fields for.
 - ii. Returned from Server
 - 1. Message
 - 2. Success
 - 3. Fields - Ordered array of Fields. DB is guaranteed to have these in correct order top->bottom as they were in PDF, so DB query orders by field ID and fills this array in that order.
- 6. BASEURL/PDFLoad
 - a. POST data to load a fillable pdf into database.
 - i. Sent to Server
 - 1. PDF - a pdf file uploaded with enctype multipart/form-data
 - 2. FILENAME - Optional name of file
 - 3. CATEGORY - The file category
 - ii. Returned from Server
 - 1. Message

2. Success

War Deployment

Source: <https://github.com/N1gorena/Dev>

Extra Libs: <https://github.com/N1gorena/Dev/tree/master/WebContent/WEB-INF/lib>

In order to redeploy the tomcat web application, follow these steps. Current .war file is name Dev.

1. Create your java .war server application.
2. Stop tomcat on the server using command: `systemctl stop tomcat`.
3. Remove the .war file and any folder that takes its name. E.g a "Dev" folder.
4. File transfer a new Java .war file to `/opt/tomcat/webapps`.
5. Change ownership and group of freshly transferred .war file to "tomcat" user
6. `Systemctl start tomcat`.

Front-End Website

The HTML file for the front end website is located at `/var/www/html/test.html`.

To make changes to the front-end, access `test.html` and modify the HTML code.

There are 4 main aspects to this document:

1. File name
`<input name="PDF" type="file">`
2. Upload File
`<input name = "FILENAME" type="text">`
3. Choose a category
`<select name="CATEGORY" >`
4. Submit
`<input type="submit" value="GO!">`

The backend API expects these 4 values.