Outreach Tracking Tool

Project Engineer: Graham Matthews, Computer Science

Project Sponsor: Jordyn Langley, Lochmueller Group

Project Advisor: Dr. Don Roberts, University of Evansville

University of Evansville

May 7, 2020

ABSTRACT

The Outreach Tracking Tool uses an iOS application, a web interface, and a MySQL database to allow users to efficiently track recruiting efforts of potential candidates. Users log in to the iOS application and, using their device's camera, take a picture of candidate documents. Using a vision API, the words in the image are recognized and appropriately stored. Users can query this data later using the web interface.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Don Roberts for his support and expertise throughout this project. He offered valuable guidance as I progressed with development, and offered impactful advice used to steer the project closer to its goals.

I would also like to thank Jordyn Langley, Brittany Ludwig and Julie Kelly of Lochmueller Group for their cooperation and encouragement as project sponsors. They offered valuable insights into the various use cases users like themselves would be faced with.

LIST OF FIGURES

INTRODUCTION

As corporations grow, their methods used to attract talented job candidates widens. While a single dedicated corporate recruiter may suffice for some time, eventually a need arises for multiple recruiters to visit candidates at career fairs across the nation. After collecting resumes, either the recruiter is left to input the candidate's data into an entry system on their own, or the candidate is asked to fill out another application with the same information included on their resume. Either way, one individual's time is wasted duplicating information already available.

The Outreach Tracking Tool aims to solve these issues. Using a vision framework, phone cameras are used to analyze images of candidate resumes and convert them to a digital format. The data is then automatically sorted and stored using machine learning models. This process saves time for all involved parties, and allows candidates to spend more time talking with recruiters, and gives recruiters the ability to streamline their recruitment efforts. The process also allows for recruiters to efficiently search potential candidate information in seconds, compared to the several minutes that may be required to search a large spreadsheet document manually or with other inefficient searching means.

PROBLEM STATEMENT

Currently, corporate recruiters reach out to potential candidates in a variety of ways. This results in trips to career fairs and outreach to candidates in general. As they receive candidate resumes, they may choose to store information in an ordinary spreadsheet or database application. This requires time wasted duplicating information already available. In some scenarios, data may be stored in multiple locations due to multiple individuals being sent to career fairs. Instead of residing in a single, central location of data, resumes and candidate data may be spread through several documents in several locations. Alternatively, the recruiter may ask for the candidate to fill out a new application, though this application typically asks for information already included on the candidate's resume. This approach saves time for the recruiter, but may leave the candidate upset or feeling as though their resume was sent in vain.

According to Glassdoor's HR and Recruiting statistics for 2019, based on data from 2018, 32% of job seekers prefer to look for information about a company they would like to work for through professional networking. Furthermore, 58% of job seekers find clear and regular communication the most important aspect of a positive job seeking experience. 62% of job seekers prefer a short application process, specifically less than two weeks from initial application to job offer [1]. Keeping in mind these aspects, the proposed solution should ensure that recruiters have the time to make adequate face-to-face communication at professional networking events and stay organized with automatic data storage.

Currently, the project sponsor attends recruiting events such as career fairs and collects paper resumes. After attending these events, the information on the resumes are transferred to her own spreadsheet document with color coding or columns to specify attributes of each candidate. Each resume must be manually entered into the spreadsheet document.

There are existing tools that solve this problem. For example, Ascend Software's SmartTouch AIR utilizes optical character recognition to solve the same issues outlined here, namely reduction in cost of labor for printing, organizing and searching documents. SmartTouch AIR expands to include more administrative documents such as contracts, invoices and expense reports to assist in accounting and legal departments [2]. The Outreach Tracking Tool will differ from current technologies by ensuring the tool is cost effective while maintaining the core functionality of similar tools.

PROJECT REQUIREMENTS AND SPECIFICATIONS

These were the requirements and specifications provided in the proposal phase. These requirements and specifications provided functionality required for recruiting professionals to efficiently organize candidate data.

1. <u>An iOS application for recruiters to scan candidate documents</u>

The iOS application provides the functionality for recruiters to scan candidate documents. Candidate documents may range in size, such as a one page résumé, a multipage résumé or business cards. The application should confirm the data is accurate with the user before saving to the database.

2. <u>Manual entry of candidate data</u>

In the event that a candidate's documents are not able to be recognized by the application, or if a candidate's information was read inaccurately, there still should be an option to manually input the candidate's data using the iOS application, and the ability to edit the user's information before confirming. Manual entry and editing of data also should be supported on the web interface, in the event a candidate's data changes.

3. <u>Utilize optical character recognition to automate the process of storing candidate information</u>

The iOS application should be able to recognize text in images. The text extracted from these images should then be classified and stored appropriately into a database. The optical character recognition should be reliable enough to remain the preferred option over manual entry.

4. <u>Web interface for recruiters to query data</u>

A web interface should be built to allow for review of data at a later date. The web interface should allow for users to filter and update data.

5. <u>MySQL database implementation</u>

The data should be stored on a MySQL database instance that can communicate with both the iOS application and web interface.

6. <u>Multiple user support</u>

The iOS and web applications should allow for multiple users of different security levels. A super-admin role should allow users to create new users and give administrator privileges to users. Administrators should be allowed to create new users, and users should be able to use the basic functions of the application.

7. <u>User authentication</u>

The data users interact within the application is sensitive and should be protected. Since the project sponsor is in the process of adopting Okta's single sign on functionality, authentication should be accomplished by integrating Okta authentication.

As the implementation of the project began, it became clear that another requirement needed to be added.

8. <u>Text Classification</u>

After text has been recognized and extracted from the document scanned using optical character recognition, the text must be classified by some means in order to sort the data appropriately. This specification requires that the application be able to classify the recognized text as contact information, education information, work experience, or additional information, then further classify the text into a category according to what kind of information the text was classified as. Contact information should be appropriately classified as a name, email, phone, and so on.

DESIGN APPROACH

The system contains two user-facing pieces: the mobile application and the web application. Both of these applications communicate with an API server that handles updates to the database. This relationship is modeled in Figure 1.

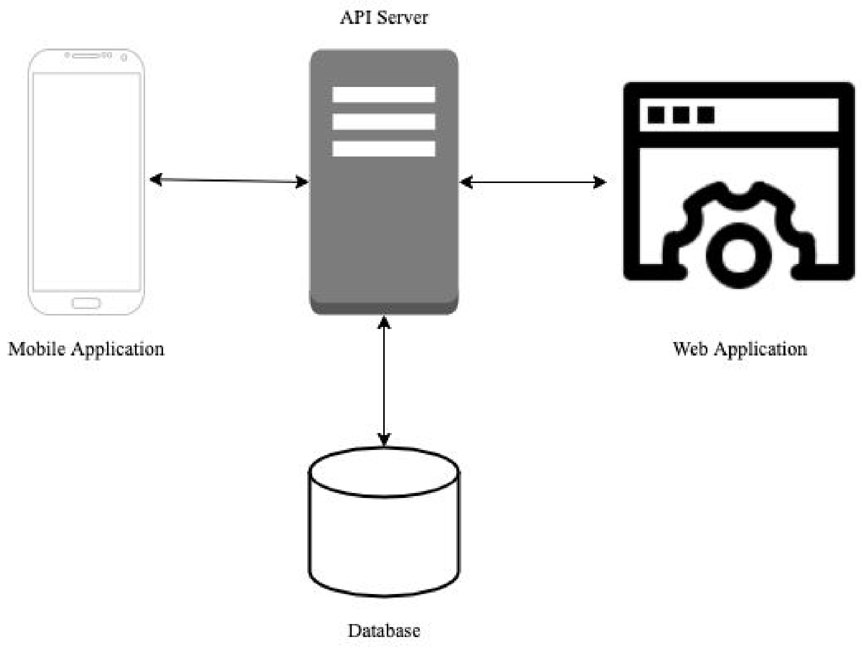


Figure 1 - System Overview

Mobile Application Third Party Subsystems

The mobile application utilizes a number of subsystems in order to utilize optical character recognition, text classification, location information and user authentication. Apple's Vision framework allows for optical character recognition, rectangle detection and face detection in images [3]. For the purposes of this project, we are only concerned with the optical character recognition, though it is important to keep in mind the framework's capabilities in order to understand its request process. Any image from the user can be converted from a UIImage, Apple's generic image class used to store image data, and converted to a Core Graphics image format that converts the UIImage data to a bitmap image. The images are passed in an array to a Vision image request handler, and are converted to an abstract Vision request superclass. These requests are converted into the abstract request class first in case one may be performing multiple requests on the same image, for instance wanting to detect any faces and rectangles on the same image. These requests are converted to text observations, where the request handler offers multiple candidates for what the text may be. The candidate with the highest confidence value is chosen as the value of the text string.

Apple's Create ML and Natural Language frameworks are used for text classification [4][5]. The Create ML framework allows the engineer to create simple machine learning models. After these models have been trained, they are able to identify simple patterns in language. Training data consists of JSON objects with two properties: the text being classified, and the classification of the text. If one were to train a model for classification of movie reviews, for example, one may give a set of training data that includes an object with text, "Five stars, great performance by John Doe," and a classification attribute of "positive." Using the model's pattern recognition, it is possible to classify text on candidate documents.

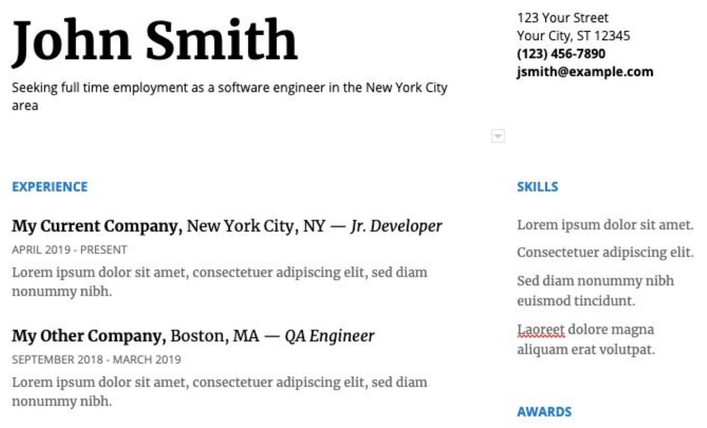
The Okta OpenID Connect package for iOS secures the mobile application by requiring user authentication before allowing the user to proceed to secure areas of the application [6]. The Okta JWT package allows the application to include a JWT token in the header of API requests [7].

Mobile Application Design

When the user opens the application, they are met with a login screen. Pressing the Sign In button presents the user with a web view to sign in through Okta. Upon authentication, the user is presented with the new data entry view.

The user is presented with three options: manual entry, automatic entry or sign out. Signing out will remove the user's authentication token and require the user to authenticate again. Selecting automatic entry automatically presents the user with a camera view. The user is prompted to position a document in the view, where the camera automatically takes a picture when the document is visible.

From here, the Vision framework identifies sections of text within the document. It is important to note that the Vision framework identifies text in the direction one would read a book, that is to say left to right and top to bottom. For the purposes of this application, this causes a discrepancy in expected behavior of the application if it were left to read in this manner. Most candidate documents do not read like a book. A human is more likely to read these documents as the candidate intends, whether that be a combination of left to right only to the middle of the page, then top to bottom for half of the page, and then left to right again beginning at the middle of the top of the page, and so on. Consider the document in Figure 2.



John Smith

Seeking full time employment as a software engineer in the New York City area

123 Your Street
Your City, ST 12345
(123) 456-7890
jsmith@example.com

EXPERIENCE

My Current Company, New York City, NY — *Jr. Developer*
APRIL 2019 - PRESENT
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh.

My Other Company, Boston, MA — *QA Engineer*
SEPTEMBER 2018 - MARCH 2019
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh.

SKILLS

Lorem ipsum dolor sit amet.

Consectetuer adipiscing elit.

Sed diam nonummy nibh euismod tincidunt.

Laoreet dolore magna aliquam erat volutpat.

AWARDS

Figure 2 - Example candidate resume

While a human would read the experience section in its entirety before continuing to the skills section, the Vision framework may read the "EXPERIENCE" heading, then the "SKILLS" heading, then the line beginning with, "My Current Company" and then the first skill. Suddenly, the text related to experience has become interleaved with other information. The solution implemented requires using a coordinate system on the image of the document to determine what lines of text are associated with what blocks of text.

When text is identified using the Vision framework, it is contained within a bounding box identified by the framework. The coordinates of this bounding box are accessible to the engineer. Using the coordinates of the bounding box, it is possible to organize the text into a TextBlock class, composed of TextLine objects. Figure 3 demonstrates the model for these two objects.

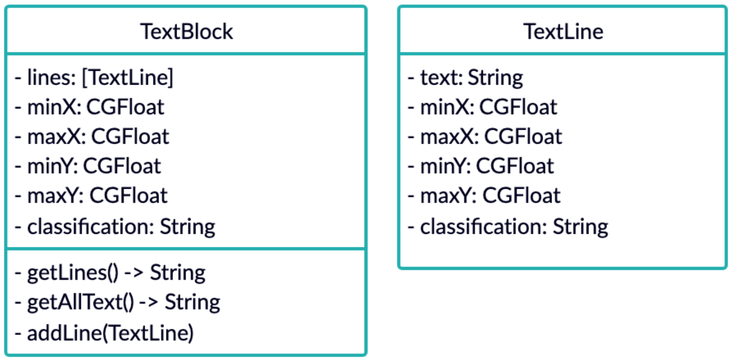


Figure 3 - TextBlock and TextLine models

Each Vision observation creates a new TextLine, which stores the appropriate coordinate values of its bounding box in the TextLine object along with its recognized text. If no TextBlock objects exist, a new TextBlock is created with its first string as the recognized TextLine. All subsequent TextLine objects are compared to existing text blocks. If the difference of minX values (the coordinate for the left-most end of the observation's bounding box) of a TextLine are less than some constant, the line is aligned with the existing TextBlock's left most value. In other words, the TextLine and the TextBlock it is being compared to share roughly the same minX value and are aligned similar to the first word of each line on this page. Now, if the difference between the minY (bottom-most coordinate) of the TextBlock and the maxY (upper-most coordinate) of the TextLine are less than some constant, the application determines the TextLine must belong to the TextBlock being compared to. If the TextLine is compared to all existing TextBlock objects and is not within the coordinate difference constants, the TextLine begins a new TextBlock. As each TextLine is added to a TextBlock, the minimum and maximum coordinate values are updated accordingly.

Now that the recognized text is organized into TextBlock and TextLine objects, the application begins classifying the text. First, the total text of the TextBlock is evaluated using a machine learning model. The text block classifier model can classify a TextBlock as a heading, contact information, education information, or additional information. After the TextBlock object is classified, each TextLine is evaluated using the appropriate model for its TextBlock classification. TextBlock objects classified as contact information are evaluated using a different model than education information TextBlock objects and so on. Contact information TextBlock objects are evaluated using a combination of text classification models and regular expressions, since matching a pattern for an email or phone number was more reliable than relying on the classification model in testing.

As text is classified, new objects for the data models are created. The data models used in the mobile application are modeled in Figure 4. Once all identified text has been classified, users are presented with the data entry. This view allows the user to edit data that was classified by the automated entry, in the event that some text was incorrectly classified or missing. The data entry view seen after scanning a candidate document is the same view as the manual entry view. Upon completion of entering and editing candidate data, a review view is presented with a synopsis of data entered. A submit button allows the user to submit the data for storage in the database, and is followed with a success view upon completion.

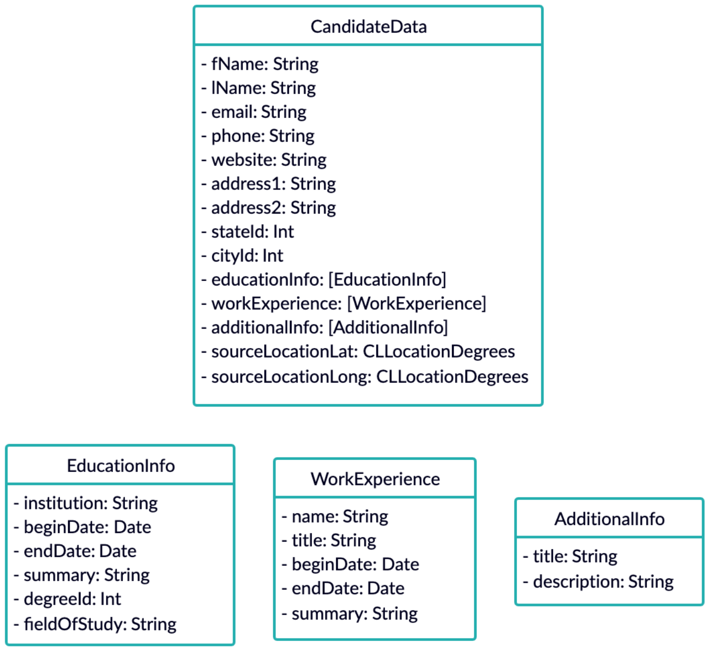


Figure 4 - Mobile application data models

API Server Third Party Subsystems

The API server utilizes a number of Node.js packages [8]. Express is a web framework for Node.js that provides routing for the API. The various API endpoints are defined using Express. The MySQL package allows for implementation of the MySQL protocol. Endpoints are secured using Okta's JWT verifier package. Using this, any requests to the API without a valid JWT token included in the header of the HTTP packet are denied. The ExcelJS package allows for management of Excel worksheets and provides functionality to export search results to a spreadsheet from the web application.

API Design

The API utilizes various routes for handling CRUD (create, review, update and destroy) operations of the data models. All endpoints are secured using a middleware function that determines the authenticity of the included JWT token. Failure to provide an authentication token or providing an invalid token when making an API request results in an error response from the API.

The application appropriately uses GET and POST HTTP protocols for the retrieval and updating of data.

Web Application Third Party Subsystems

React is a JavaScript library maintained by Facebook used for the development of single-page applications [9]. React allows for a responsive user interface without the need for the browser to load new pages. Okta again provides authentication for the application, secures page routes that should only be seen once authenticated, and provides the authentication token needed to form API requests [10]. Google Maps is used to present location data associated with candidates [11]. Font Awesome is a font and icon toolkit, ranked second for third-party font script providers behind Google Fonts [13]. Font Awesome icons are used in the web application to provide an interface that matches the user's perception of how the interface should respond to user input. Font Awesome was chosen over Google Fonts due to its larger selection of icons specifically, as Google Fonts typically focuses on font styles for text.

Web Application Design

The web application utilizes a model-view-controller design, where the model is defined in the database, the view is provided using React, and the controller is the API. The flow of data in the web application is modeled in Figure 5.

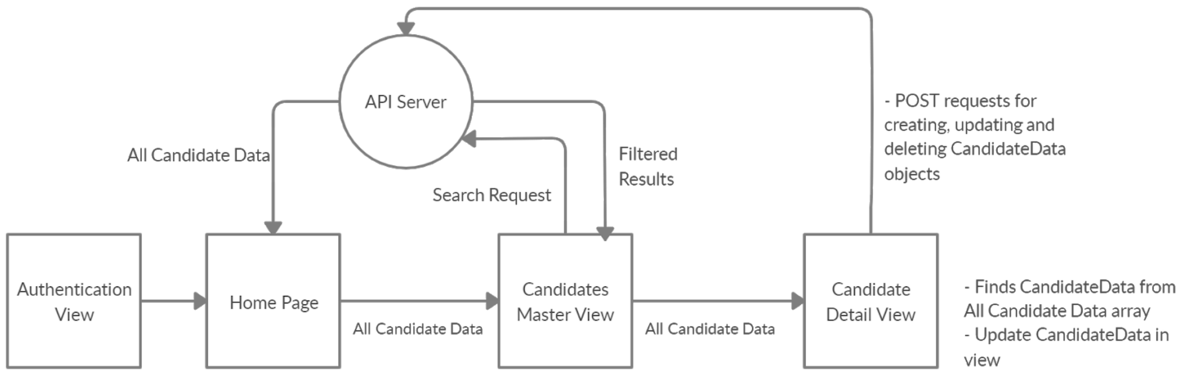


Figure 5 - Data flow in web application

After authenticating, the user is presented with a home screen. This view offers potential for displaying pertinent information to the user if desired in the future. The candidates list view presents all candidates in the database, and includes a search function to filter the list of candidates. All searches can be exported to a spreadsheet for the user to manipulate further. Clicking on a candidate's name presents the candidate detail view. This view includes all data associated with the candidate, including contact information, education information, work experience, additional information, and a Google Map view including a pin for where the candidate was added. All points of data, with the exception of the connected location, are able to be edited by application users. Icons next to each data point toggle edit mode for the data entry. The user has the option to save or cancel changes made to the data point. All updates to data are able to be accomplished without a page refresh, and this view also includes the ability for application users to leave notes related to the candidate.

Database Third Party Subsystems

The database for this application uses MySQL [13]. MySQL was chosen primarily for its low costs, but also for its integration with Node.js using the mysql package.

Database Design

The database uses a relational model. The entity relationship diagram is shown in Figure 6.

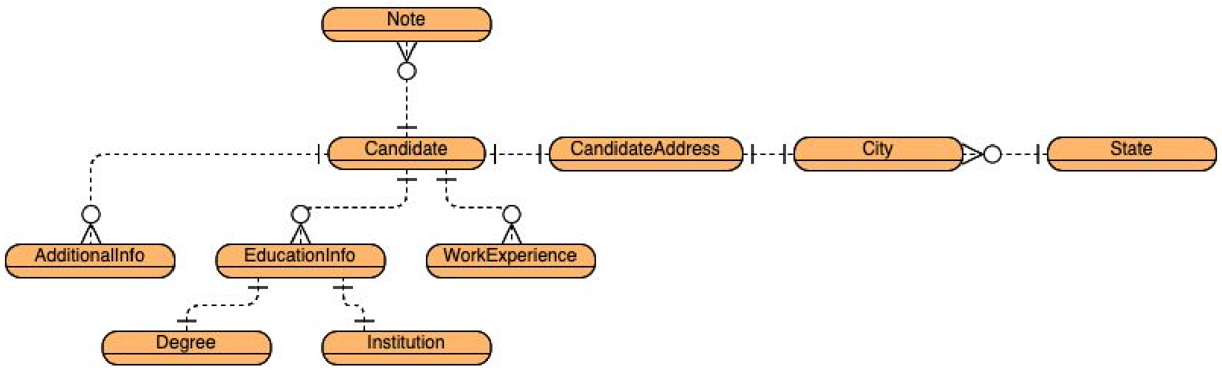
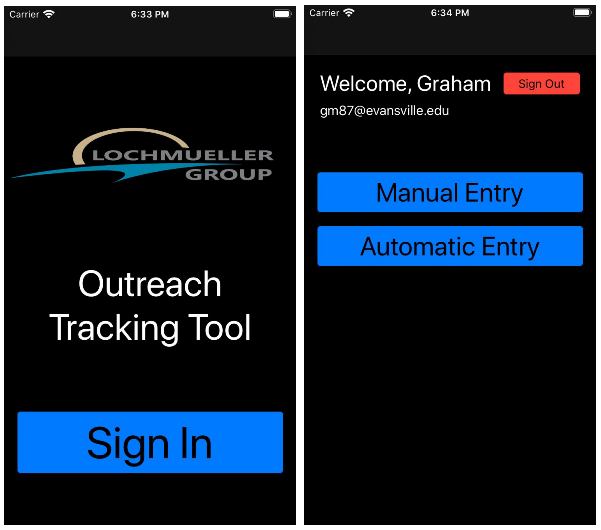


Figure 6 - Database entity relationship diagram

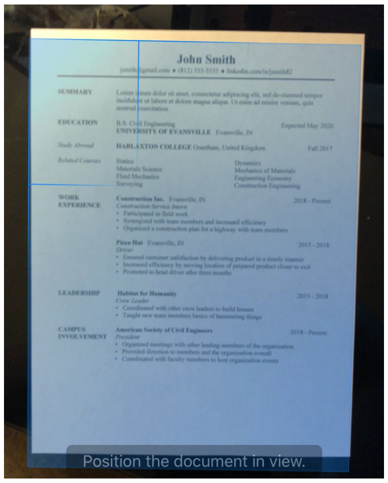
RESULTS

The application meets all requirements, to varying degrees of success. With the use of Okta, each application successfully authenticates the user before allowing the user to access functional components of the application. The authentication view for the mobile application is demonstrated in Figure 7. After the user authenticates, they are presented with the new data entry view as demonstrated in Figure 8.



Figures 7 and 8 - Authentication view and new data entry view

The mobile application utilizes a Vision framework to recognize text from images as demonstrated in Figure 9.



John Smith

jsmith@gmail.com • (812) 555-5555 • linkedin.com/in/jsmith82

SUMMARY — Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation

EDUCATION — B.S. Civil Engineering — Expected May 2020
UNIVERSITY OF EVANSVILLE Evansville, IN

Study Abroad — **HARLAXTON COLLEGE** Grantham, United Kingdom — Fall 2017

Related Courses — Statics, Materials Science, Fluid Mechanics, Surveying, Dynamics, Mechanics of Materials, Engineering Economy, Construction Engineering

WORK EXPERIENCE — **Construction Inc.** Evansville, IN — 2018 - Present
Construction Service Intern
- Participated in field work
- Synergized with team members and increased efficiency
- Organized a construction plan for a highway with team members

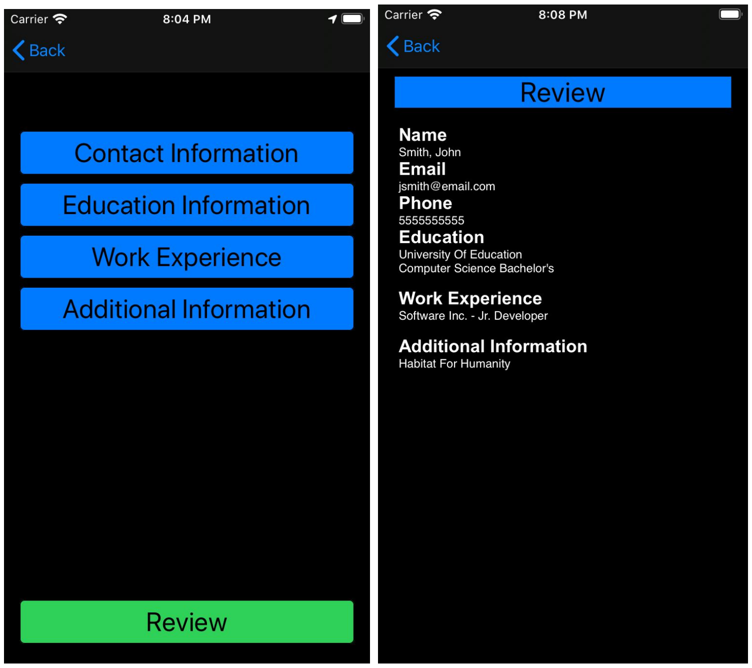
Pizza Hut Evansville, IN — 2015 - 2018
Driver
- Ensured customer satisfaction by delivering product in a timely manner
- Increased efficiency by moving location of prepared product closer to exit
- Promoted to head driver after three months

LEADERSHIP — **Habitat for Humanity** — 2015 - 2018
Crew Leader
- Coordinated with other crew leaders to build houses
- Taught new team members basics of hammering things

CAMPUS INVOLVEMENT — **American Society of Civil Engineers** — 2018 - Present
President
- Organized meetings with other leading members of the organization
- Provided direction to members and the organization overall
- Coordinated with faculty members to host organization events

Figure 9 - Camera view automatically capturing candidate document

The mobile application also allows for manual entry of candidate data and a review of the candidate data entered as demonstrated in Figure 10.



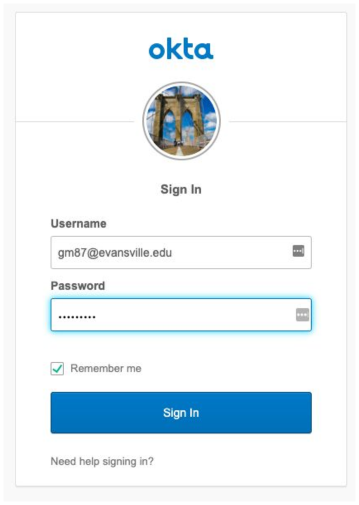
Name
Smith, John
Email
jsmith@email.com
Phone
5555555555
Education
University Of Education
Computer Science Bachelor's

Work Experience
Software Inc. - Jr. Developer

Additional Information
Habitat For Humanity

Figure 10 - Data entry and review views

MySQL provides a database to store candidate information. The web application first authenticates users as demonstrated in Figure 11.

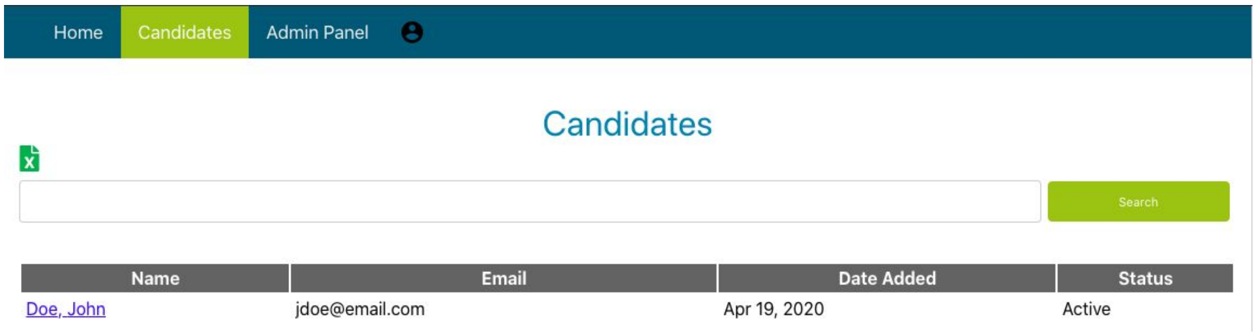


Username: gm87@evansville.edu

Password: ·········

Figure 11 - Okta authentication view

After users are authenticated, they are able to view a master list of all candidates as shown in Figure 12.

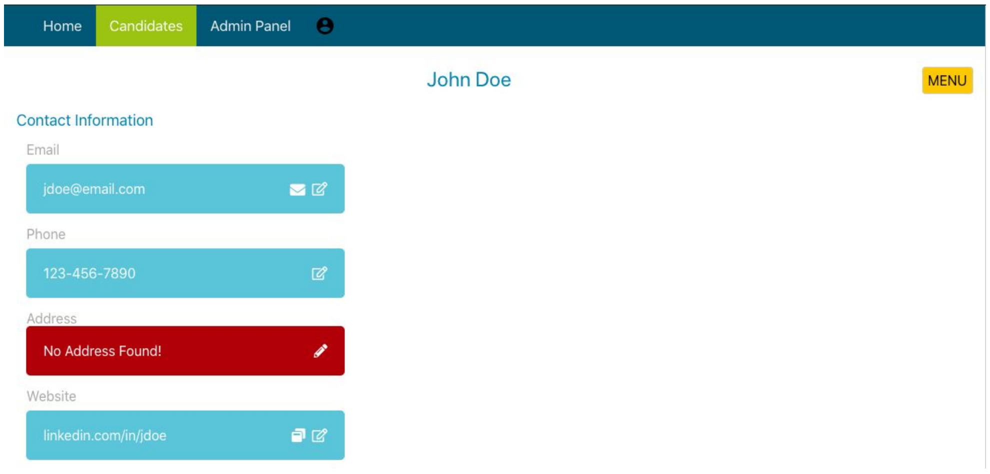


Candidates

Name	Email	Date Added	Status
Doe, John	jdoe@email.com	Apr 19, 2020	Active

Figure 12 - Candidates list view

When the user clicks on the name of the candidate, they are presented with a candidate details view that allows for review, updating and deletion of candidate data. The candidate detail view is shown in Figures 13 and 14.



John Doe

Contact Information

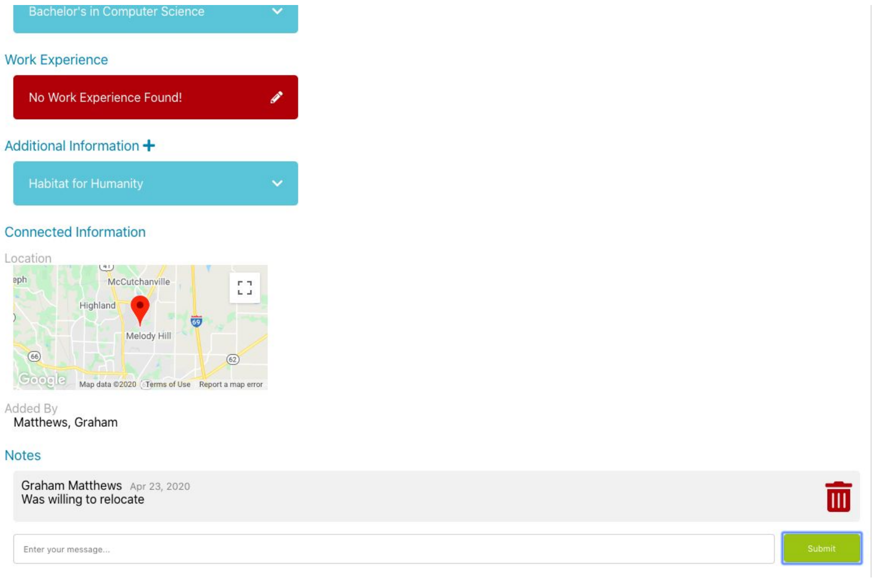
Email
jdoe@email.com

Phone
123-456-7890

Address
No Address Found!

Website
linkedin.com/in/jdoe

Figure 13 - Candidate contact information



Bachelor's in Computer Science

Work Experience
No Work Experience Found!

Additional Information +
Habitat for Humanity

Connected Information
Location

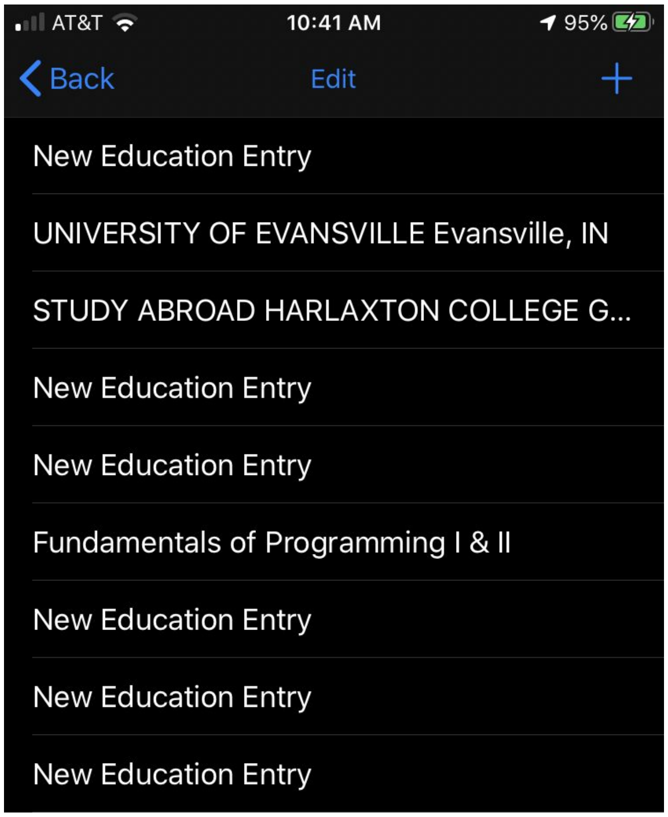
Added By
Matthews, Graham

Notes
Graham Matthews Apr 23, 2020
Was willing to relocate

Enter your message...

Figure 14 - Candidate education, work experience, additional, connected and notes information

While the mobile application does recognize and classify text from images, the accuracy of the classifier could be greatly improved. Due to a small set of training data, the machine learning models are unable to confidently classify some text blocks, resulting in false positives as shown in Figure 15.



New Education Entry
UNIVERSITY OF EVANSVILLE Evansville, IN
STUDY ABROAD HARLAXTON COLLEGE G...
New Education Entry
New Education Entry
Fundamentals of Programming I & II
New Education Entry
New Education Entry
New Education Entry

Figure 15 - False positives after an automatic entry

Lines of text still appear separate from other lines they should be associated with. This is believed to be a result of only comparing the difference between coordinates using a constant value, and not other means.

CONCLUSION

The Outreach Tracking Tool provides a solid foundation for an application with the potential to be used in all industries. Future work for the project could include expanding the training data used in each of the text classification models, improving the algorithm for associating lines of text together and improving the user interface of the mobile application.

REFERENCES

[1] Glassdoor. 50 HR and Recruiting Stats for 2019. Retrieved October 14, 2019 from https://library.glassdoor.com/c/50-hr-and-recruiting-stats-2019.

[2] Ascend Software. SmartTouch AIR for Workday. Retrieved September 28, 2019 from https://www.ascendsoftware.com/imaging-ocr-for-workday.

[3] Apple. Vision | Apple Developer Documentation. Retrieved May 2, 2020 from https://developer.apple.com/documentation/vision.

[4] Apple. Create ML | Apple Developer Documentation. Retrieved May 2, 2020 from https://developer.apple.com/documentation/createml.

[5] Apple. Natural Language | Apple Developer Documentation. Retrieved May 2, 2020 from https://developer.apple.com/documentation/naturallanguage.

[6] Okta. Add User Authentication to Your iOS App. Retrieved May 2, 2020 from https://developer.okta.com/code/ios/.

[7] Github. okta/okta-ios-jwt. Retrieved May 2, 2020 from https://github.com/okta/okta-ios-jwt.

[8] Node.js. About | Node.js. Retrieved May 2, 2020 from https://nodejs.org/en/about/.

[9] React. React - A JavaScript library for building user interfaces. Retrieved May 2, 2020 from https://reactjs.org/.

[10] Okta. Add User Authentication to Your React App. Retrieved May 2, 2020 from https://developer.okta.com/code/react/.

[11] Google. Overview | Maps JavaScript API. Retrieved May 2, 2020 from https://developers.google.com/maps/documentation/javascript/tutorial.

[12] Font Awesome. Font Awesome. Retrieved May 2, 2020 from https://fontawesome.com/.

[13] MySQL. MySQL. Retrieved May 2, 2020 from https://www.mysql.com/.

BIOGRAPHY

Graham Matthews graduated from the University of Evansville in May 2020 with a degree in Computer Science. He is an Evansville native and will continue to work in the Evansville area for the foreseeable future as a Systems Analyst at Lochmueller Group. He develops tools that automate data entry and data querying tools for internal use. This and other projects of his can be found at github.com/gm87.