Outreach Tracking Tool

Engineer: Graham Matthews, Computer Science

Project Sponsor: Jordyn Langley, Lochmueller Group

Project Advisor: Dr. Don Roberts, University of Evansville

University of Evansville

December 5, 2019

ABSTRACT

The Outreach Tracking Tool uses an iOS application, 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.

TABLE OF CONTENTS

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 API, phone cameras may be used to analyze images of candidate resumes and convert them to a digital format. The data is then automatically sorted and stored. 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 takes paper resumes. After attending these events, the resumes are allocated in 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.

Rather than waiting in a line to talk with any individual company, candidates may simply hand a resume to the recruiter for scanning, saving time and energy that would otherwise be spent filling out applications which ask for the same data.

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 requirements and specifications provide functionality required for recruiting professionals to efficiently organize candidate data.

1. An iOS application for recruiters to scan candidate documents

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. Manual entry of candidate data

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 should still 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 should also be supported on the web interface, in the event a candidate's data changes.

3. Utilize optical character recognition to automate the process of storing candidate information

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. Web interface for recruiters to query data

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>

Since users already use Microsoft accounts for company email, authentication should be accomplished simplest by integrating Microsoft authentication. As time permits, authentication should be accomplished by Microsoft authentication, but simpler email and password combinations of authentication may be used in its absence.

DESIGN APPROACH

Third-Party Subsystems

MySQL will be used for the database systems [3]. It was chosen for its cost effectiveness and simple structure. Okta will be used for authentication [4]. It was chosen for its cost effectiveness and simple implementation. Several Node.js frameworks will also be used in the web application. Express is a high performance routing framework that can be used to specify fetch requests [5]. The exceljs framework provides functionality to write to a spreadsheet file if, for example, the user chooses to export the results of a query from the web application [6]. The node-schedule framework is used to simulate Cron jobs on the Node.js server [7]. These automated jobs will be used to send reminders to recruiters to remember to follow up with applicants as needed. The Apple Vision framework will be used for optical character recognition when scanning candidate documents using the mobile device's camera [8].

Design Approach

The Outreach Tracking Tool seeks to solve problems for both interest groups, recruiters and candidates. For recruiters performing outreach, this tool should simplify the process of acquiring candidate data. Rather than seeking out candidate data, recruiters simply scan text documents for analyzing. The data from these documents is then stored in a single database that can be accessed by multiple users. For candidates, this speeds up the job search process.

Three components make up the overall design of the project, the iOS application, the web interface, and the MySQL database. The iOS application is responsible for providing a user friendly interface to allow for user authentication, document scanning, and manual entry. The web interface is responsible for providing another user interface that provides user authentication, data presentation and querying, and data manipulation. The MySQL database acts as a middleman between the iOS application and web interface by ensuring both applications reference the same data. Both the iOS and web application communicate communicate with the MySQL server through the application server. Thus, the MySQL database provides a central location for candidate information. Figure 1 diagrams the system overview for the application.

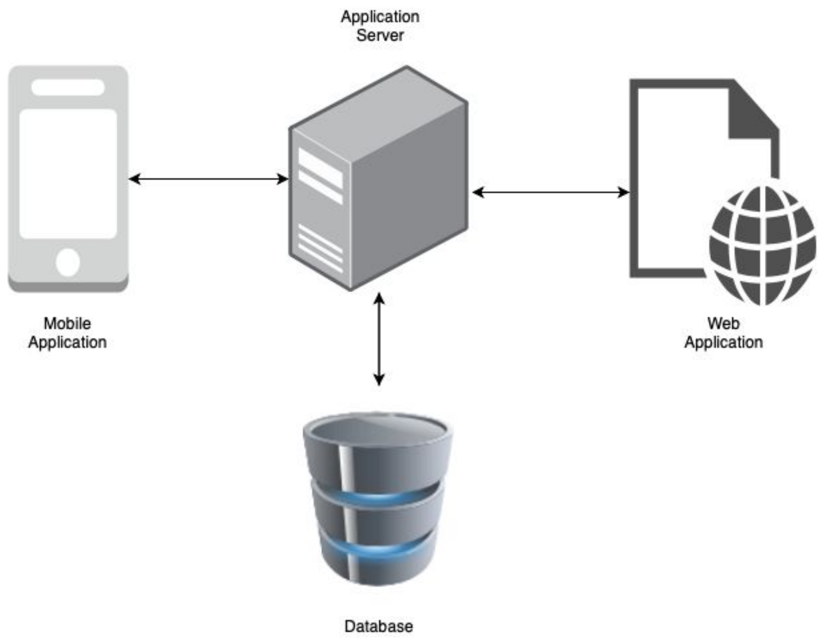

Figure 1 - System Overview

The iOS application will follow a model-view-controller design pattern. Candidate data is the model. The user interface is the view. The controller is the mediator of the relationship between the model and the view.

The iOS application should open initially to a sign in page. Figure 2 displays a mock-up sign-in view that would be presented to the user upon launching the mobile application.

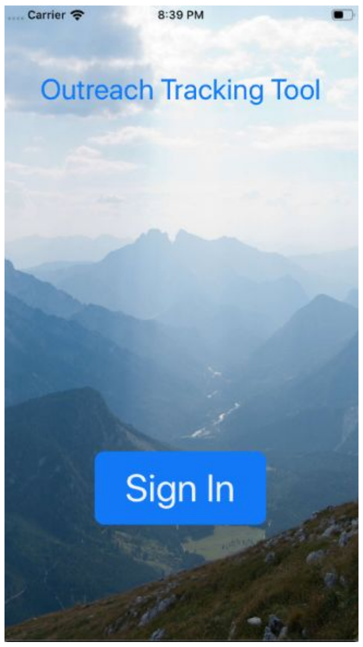

Figure 2 - iOS Application Sign In Screen Prototype

After the user has been authenticated, they should be presented with a new view with two options, either manual entry or scanned entry. The entry page mock up is displayed in Figure 3.

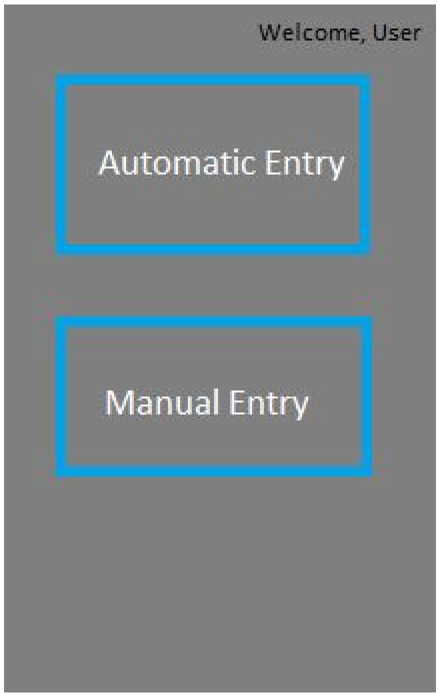

Figure 3 - iOS Application Entry Screen Prototype

If the user chooses manual entry, they should be directed to a blank form that allows for user input. If the user chooses automatic entry, the device's camera should activate and allow the user to scan a document with information. After a successful scan, the user should be redirected to the same form they would have been with manual entry, but with fields automatically filled using information gathered from the image. The user should then be allowed to confirm the placement is correct. If an element was misidentified, the user should be able to edit the element and correct the error. After confirming all scanned elements are correct, the user can save the entry. The entry will be saved to the database, with all entries listed and an image of the scanned document included. The form mock up is displayed in Figure 4.

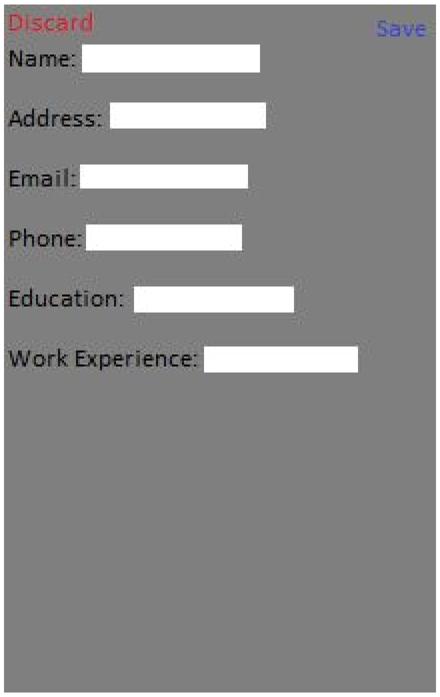

Figure 4 - iOS Application Entry Details View Prototype

Error messages should be implemented for appropriate error handling, such as communication errors over the network or for invalid data entry. If the form is filled with acceptable data upon submission, a success message should be displayed to the user. The complete application workflow is demonstrated in Figure 5.

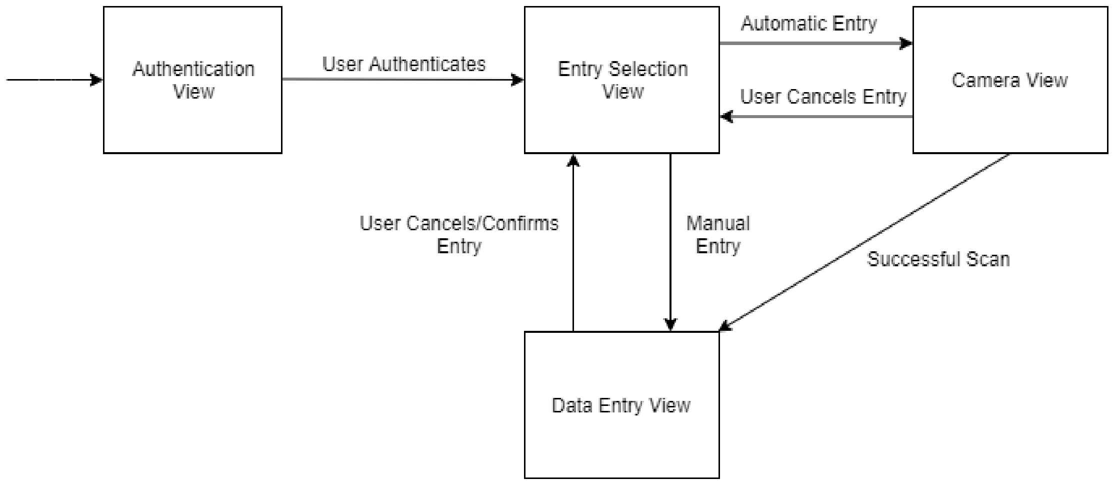

Figure 5 - Mobile Application Workflow

The web interface follows a model-view-controller design pattern. Candidate data is the model. The user interface is the view. The controller is the backend logic that controls what view to display when and with what data from the model. The system overview is displayed in Figure 6.

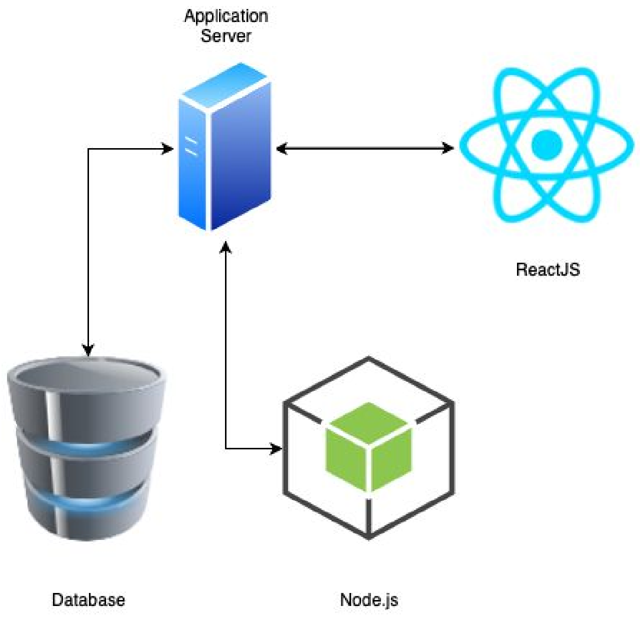

Figure 6 - Web Application System Overview

The web interface should allow the user to sign in before presenting data. After the user authenticates, a dashboard should be presented showing pertinent information, such as recent new candidates or reminders to follow up with candidates. The mock up for this landing page is demonstrated in Figure 7.

From here, the user should have the option to navigate to the candidates page and query data from the database. Upon initially loading the candidates page, the user should be presented with a table view of candidate data for candidates being recruited. Candidates that have been marked as hired or uninterested should not be shown in this initial view, but may be queried. The user should be able to query the database using a combination of dropdown menus and text fields. Users should have the option to edit and remove data as necessary. Users should have the option to export the results of a query to a document of their choice. The mock up view for the candidates page is modeled in Figure 8.

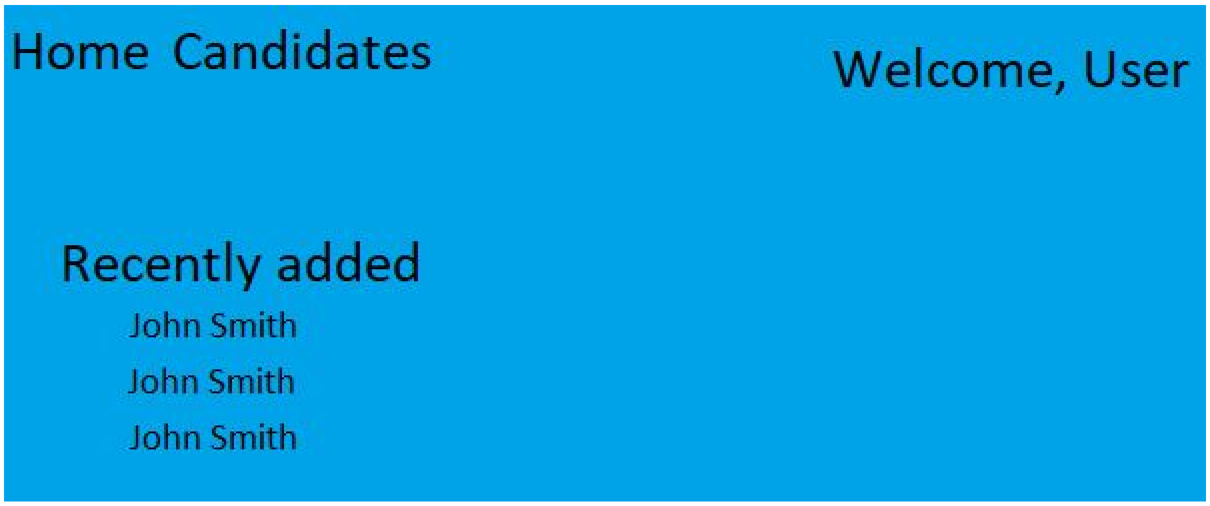

Figure 7 - Mock Homepage with Navigation Bar

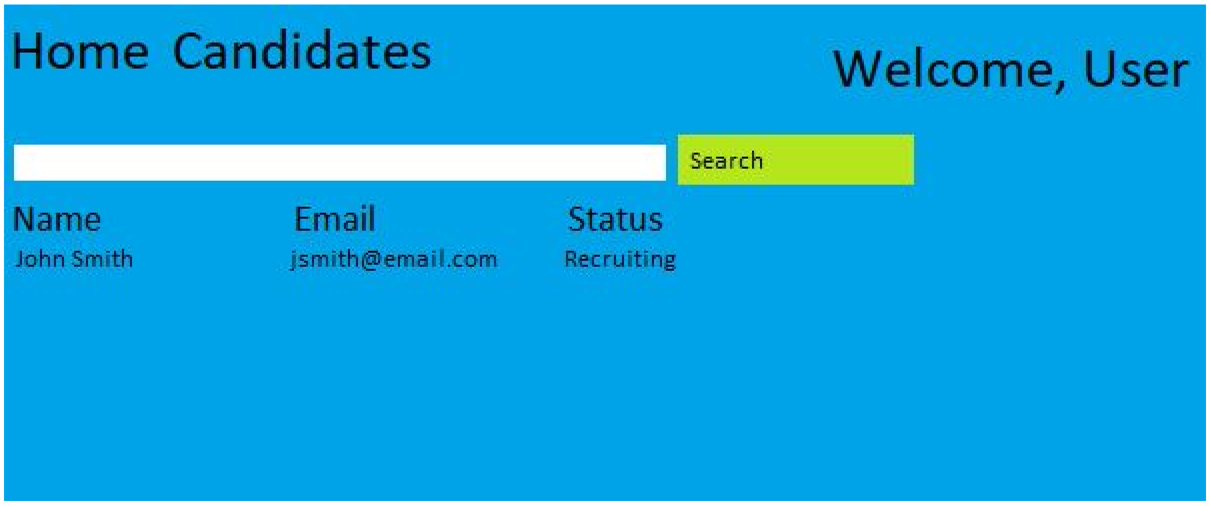

Figure 8 - Mock Candidates page with Search Bar

Clicking on the name of a candidate should link to a new page where more details on the candidate can be viewed. This view is simulated in Figure 9.

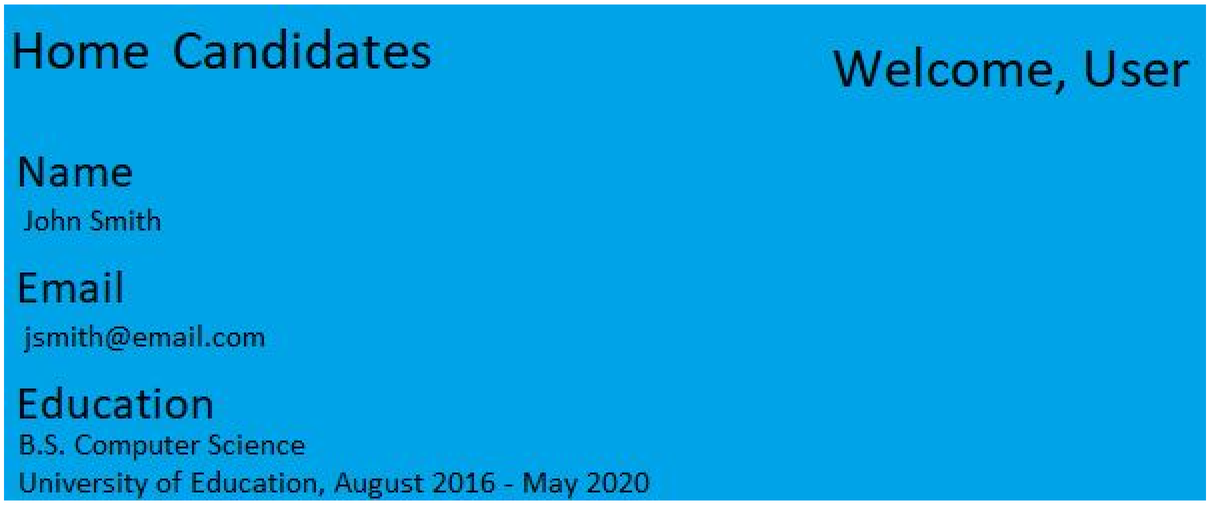

Figure 9 - Mock Candidate Detail Page

The web application workflow is displayed in Figure 10.

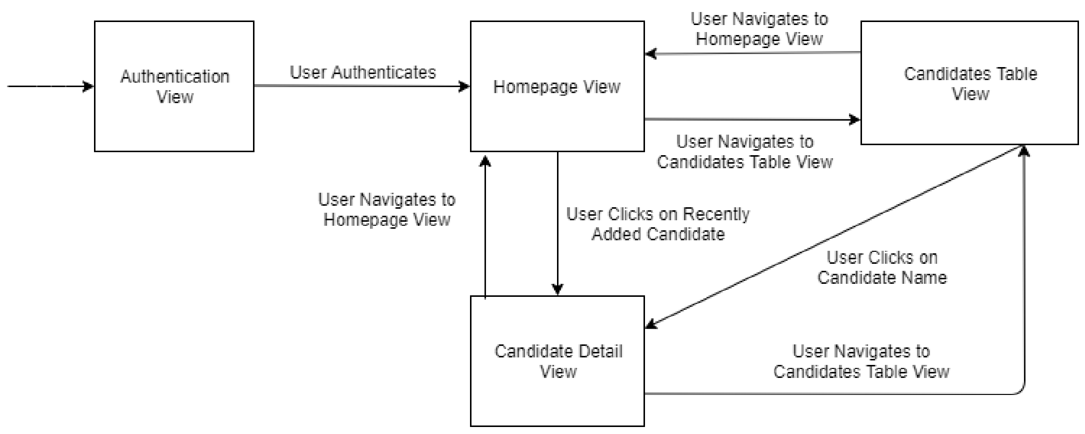
Authentication View
User Authenticates
Homepage View
User Navigates to Homepage View
Candidates Table View
User Navigates to Candidates Table View
User Navigates to Homepage View
User Clicks on Recently Added Candidate
User Clicks on Candidate Name
Candidate Detail View
User Navigates to Candidates Table View

Figure 10 - Web Application Workflow

The database should use a relational model. Tables for the database should at least include a table for users, who are those who have accounts within the application, a table for candidates, those whose information is being gathered at career events, a table for education information for applicable candidates, a table for work experience information for candidates, a table for degrees, and a table for additional information. The additional information table encompasses any information provided on the resume that may not fall under the education information or work experience tables. These tables are modeled in Tables 1 through 6.

Attribute	**Description**
id	An auto-incrementing primary key for the row
fName	The first name of the user
lName	The last name of the user
email	The email address of the user
isAdmin	A boolean for if the user is an administrator or not
active	A boolean for if the user account is active or not

Table 1 - Users

Attribute	**Description**
id	An auto-incrementing primary key for the row
fName	The first name of the candidate
lName	The last name of the candidate
email	The email of the candidate
phone	The phone number of the candidate
status	The status of the candidate. May be set to a finite number of states, such as recruiting, hired, or not interested
educationInfo	An integer to the ID of the corresponding education information row, can be null
workExperience	An integer to the ID of the corresponding work experience row, can be null
additionalInfo	An integer to the ID of the corresponding additional information row, can be null

Table 2 - Candidates

Attribute	**Description**
id	An auto-incrementing primary key for the row
institution	The name of the educational institution
beginDate	The date the candidate began at the educational institution
endDate	The date the candidate completed the educational institution, or the date the candidate expects to complete the educational institution
summary	A short summary of the candidate's accomplishments at the institution if included
degree	An ID of a degree entry from the degrees table

Table 3 - Education Information

Attribute	**Description**
id	An auto-incrementing primary key for the row
type	The type of degree (high school, B.S., B.A., M.S. etc.) represented as an enum
field	The field of study of the degree (Computer Science, Computer Engineering, etc.)

Table 4 - Degrees

Attribute	Description
id	An auto-incrementing primary key for the row
name	The name of the workplace
beginDate	The date the candidate began working at the workplace
endDate	The date the candidate ended working at the workplace
summary	A short summary of the candidate’s accomplishments at the workplace if included

Table 5 - Work Experience Information

Attribute	Description
id	An auto-incrementing primary key for the row
title	The title of the information
description	A description of the additional information

Table 6 - Additional Information

DESIGN EVALUATION

Using the iOS application as outlined in the design approach, it is clear to see how the application meets specifications one, two and three. By developing an iOS application with options to manually enter candidate data, or automatically scan candidate data, the requirements to manually enter candidate data and create an application to scan candidate documents into the database is accomplished. By utilizing Apple's Vision API, requirement three is also satisfied by utilizing optical character recognition to automate the process of storing candidate data.

By implementing a web application with ReactJS and Node.js, the solution meets specification four. The web application will communicate with a MySQL database using the Node.js backend to establish a web interface used to query a database. Using MySQL also satisfies requirement five. Finally, with the use of Okta for user authentication, requirements six and seven are satisfied. Okta provides the functionality required for multiple user support and user authentication with ease.

CONCLUSION

The project engineer's qualifications for the project include the University of Evansville's Mobile Application Development and Databases courses, and a number of previous personal projects completed using ReactJS and Node.js.

STATEMENT OF WORK

The project is estimated to take a total of 170 hours. At minimum, the final project should deliver an iOS application that allows for manual data entry after user authentication. The final project should also deliver a web application that can be used to query the data as the user needs. The final project should also include the use of optical character recognition in the mobile application for automatic document entry. As time permits, the remaining hours will be devoted to the user interface for the remaining time of the project. The approximate schedule and time estimates are as follows:

Week 1 (12/16)	10 Hours	Setting up version control, database and primitive user interfaces
Week 2 (12/23)	5 Hours	Implementing manual creation of candidate data
Week 3 (12/30)	5 Hours	Implementing manual updating of candidate data
Week 4 (1/6)	10 Hours	Efficiently querying candidate data
Week 5 (1/13)	10 Hours	Export queries to a spreadsheet document, implement user authentication on web application
Week 6 (1/20)	10 Hours	Adding different views to mobile application, implement user authentication on mobile application
Week 7 (1/27)	10 Hours	Implement primitive optical character recognition
Week 8 (2/3)	10 Hours	Finish implementing optical character recognition

Week 9 (2/10)	10 Hours	Begin implementing text classification of documents
Week 10 (2/17)	10 Hours	Continue implementing text classification of documents
Week 11 (2/24)	10 Hours	Complete implementation of text classification of documents
Week 12 (3/2)	10 Hours	Add confirmation screen after scanning candidate document, automated reminders for recruiters to reach out to candidates
Week 13 (3/9)	10 Hours	Add admin and super admin functionality to web application
Week 14 (3/16)	10 Hours	Code review, refactoring and documentation as needed
Week 15 (3/23)	10 Hours	User testing, implement updates from user testing
Week 16 (3/30)	10 Hours	Implement updates from user testing as needed
Week 17 (4/6)	10 Hours	Focus on user interface
Week 18 (4/13)	10 Hours	Focus on user interface

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] MySQL. 2019. MySQL. Retrieved from https://www.mysql.com/.

[4] Okta. 2019. Okta. Retrieved from https://www.okta.com/.

[5] npm. 2019. express. Retrieved from https://www.npmjs.com/package/express.

[6] npm. 2019. exceljs. Retrieved from https://www.npmjs.com/package/exceljs.

[7] npm. 2019. node-schedule. Retrieved from https://www.npmjs.com/package/node-schedule.

[8] Apple. 2019. Vision | Apple Developer Documentation. Retrieved from https://developer.apple.com/documentation/vision.