2017

Migrant Farm Worker App – AgHelp!

Huang Xiaomei
Grand Valley State University

Migrant Farmworker App – AgHelp!

By

Xiaomei Huang

April, 2017

Migrant Farmworker App – AgHelp!

By
Xiaomei Huang

A project submitted in partial fulfillment of the requirements for the degree of

Master of Science

in Computer Information Systems

at

Grand Valley State University

April, 2017

Jonathan R. Engelsma **April, 2017**

Table of Contents

Abstract

Migrant and Seasonal Farmworkers (MSFWs) travel each year to help cultivate and harvest crops in various regions of the United States. They often struggle to find resources such as education, health care or legal services when they get to a new place. On the other side, agencies with limited outreach budget also struggle to connect with them. Meanwhile, growers pay thousands of dollars for recruiting farmworkers every harvest season but often fail in finding enough laborers, while lots of unemployed farmworkers are waiting.

To estimate the potential of using a mobile app to address this problem within the farm worker community, we interviewed more than 70 farmworkers, agencies, and growers through Grand Valley State University's Customer Discovery Program in September 2016. We found a strong demand for a convenient and portable way to connect farmworkers, agencies, and farmers to real-time information.

This project is to build a native mobile application named "AgHelp!" for iOS phone users, written in Swift 3.x. It provides a platform for connecting migrant seasonal farmworkers, agencies, growers, and potentially Mexican stores. Bringing these groups together has never been done. "AgHelp!" will assist migrant farmworkers to locate resources more conveniently and help growers find and retain laborers. It also will help agencies locate seasonal workers who are not living in labor camps, and increase the number of MSFWs served. The notification feature of this app will alert users about the latest events, jobs, and news, so that it will help secure farmworkers' jobs and lives when they are traveling.

Introduction

Michigan's agricultural industry relies on more than 94,000 migrant farmworkers to hand-harvest more than 44 different crops every year. In 2014, more than $9 million in revenue was lost due to a shortage in labor in the fruit industry alone. Currently, farmworkers rely on word of mouth, fliers, or unscrupulous contractors to locate work and shelter. For these mobile workers, the insecurity of not knowing the local area works against them. They don't always know where to find basic health care and educational resources even if service providers are nearby. On the other side, agencies with limited outreach also have difficulties in locating workers due to the workers' mobility.

In September 2016, we participated in GVSU's Customer Discovery Program, which was focused on mapping out the market and stakeholders, identifying and understanding the "pain points" of potential customers, and determining the best product-to-market fit. Through this program, we interviewed more than 70 customers including farm workers, growers, and agencies. The feedback we received from these interviews has been taken as the main reference to design our app.

The goal of this app is to build a mobile platform that will connect farmworkers, growers, and agencies. It will bring the mobile workforce to the growers who struggle to fill the jobs, so that the loss in the agricultural industry can be minimized. Meanwhile, migrant farmworkers will be able to locate resources more quickly and efficiently using a mobile app, which will help them feel more secure traveling to Michigan. They can learn what resources will be available to them, and be less susceptible to labor trafficking and exploitation. Workers will also be able to maximize their hard-earned pay and obtain much-needed support services throughout the season. The app will help growers streamline the recruiting process by easily posting positions or anything associated with the jobs. Last but not least, AgHelp! will assist service providers to locate seasonal workers not living in labor camps, and notify farmworkers when they enter an area. The service providers will be able to receive more funding by serving more workers.

Program Requirements

The first goal of this app is to provide the best user experience and performance, so a native approach was adopted. IOS and Android version will be developed sequentially. In this program, the scope is limited to the iOS version using Swift 3.x.

The key features provided by this app are:

- “Contact” Tab:
  - Provide more than 5,000 service providers’ information.
  - Search function allows users to easily locate providers by category, current location, or keywords.
  - Save function lets users find important contacts even when they don’t have Internet access.
  - Providers’ business pages and post lists
  - Map, navigation, and in-app calling functions
- “Job” Tab:
  - Users can easily find jobs by state.
  - Users can receive new job notifications based on their interested locations.
  - Growers can post jobs with one click.
  - Simplified job application procedure
  - Efficient applicant management
- “Hub” Tab:
  - Official account can post fliers, news, events, jobs, and notify farmworkers.
  - Share, save posts.
  - Follow providers’ updates.
  - Register for upcoming events.
  - Apply for new jobs.
- “Notification” Tab:
  - In-app notifications on new jobs, new applicants, job status changes, updates of following events, emergency alerts, et al.

  - Alert On / Off options

- “Home” Tab:
  - User roles: Basic user and official user
  - Business card
  - Management of applied jobs, planned events, and saved posts
  - Advance management panel for official accounts.

Implementation

This project is implemented using Xcode, Swift 3.0, following Apple's standard guidelines. The package manager Cocopods was used for facilitation.

Pods used are:

Pods	Purpose
Google/Analytics	Track application events you care about and gain insights from discovery and installation to conversion and engagement.
Eureka	Elegant iOS forms
Parse	Access to Parse cloud platform

1. User interface design and user experience

Once the system requirements were determined, a prototype was created on a large portion of the system. It helped to define the system scope visually and generate some new ideas when evaluating them. Initially, the contents of the app were separated into five sections: Contact, Job, Event, Notifications, Home. A tab bar provides a good way to flatten the information hierarchy and provide access to several peer information categories at once.

Sketch was used for the graphic design. It can show in every fiber of the app:

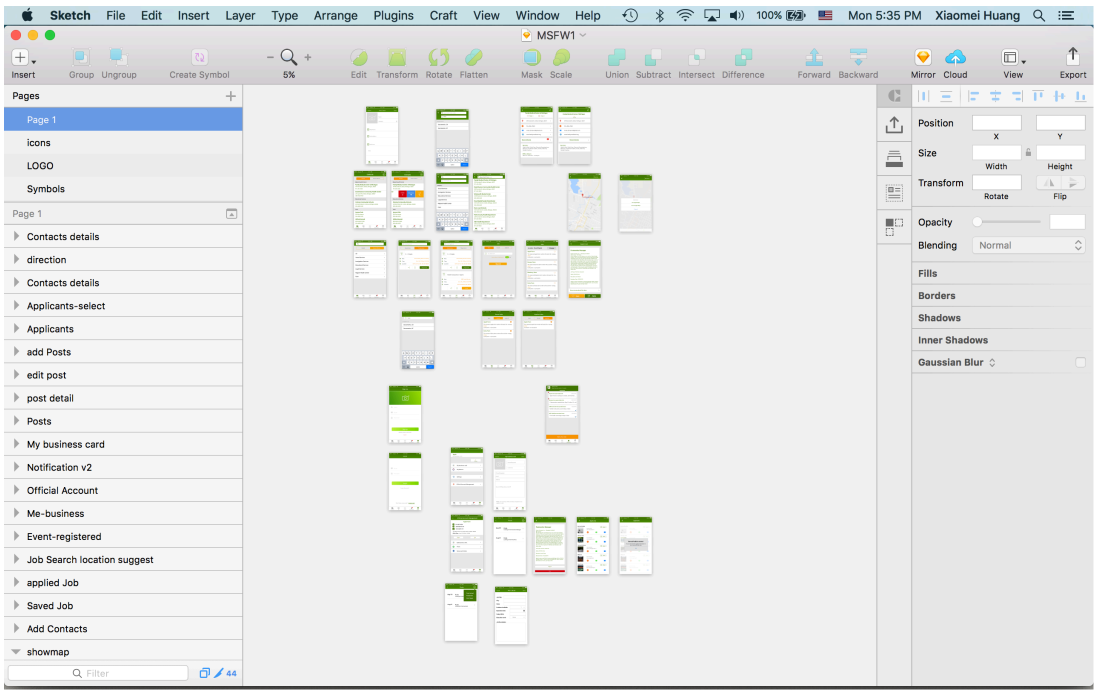
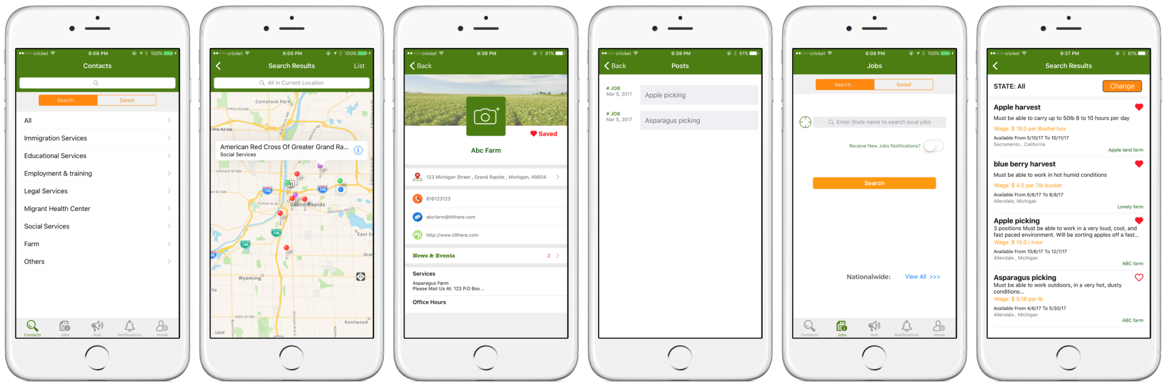


Figure 1 – User interface design by Sketch

“InVision” is another useful tool to make the design more efficient. There is a plugin in Sketch called “Craft” that allows syncing all designs to InVision. Once the files are uploaded to InVision, animations, gestures and transitions can be added to transform their static screens into clickable, interactive prototypes. It is a nice tool to collect feedback from the stakeholders and improve the user experience design.



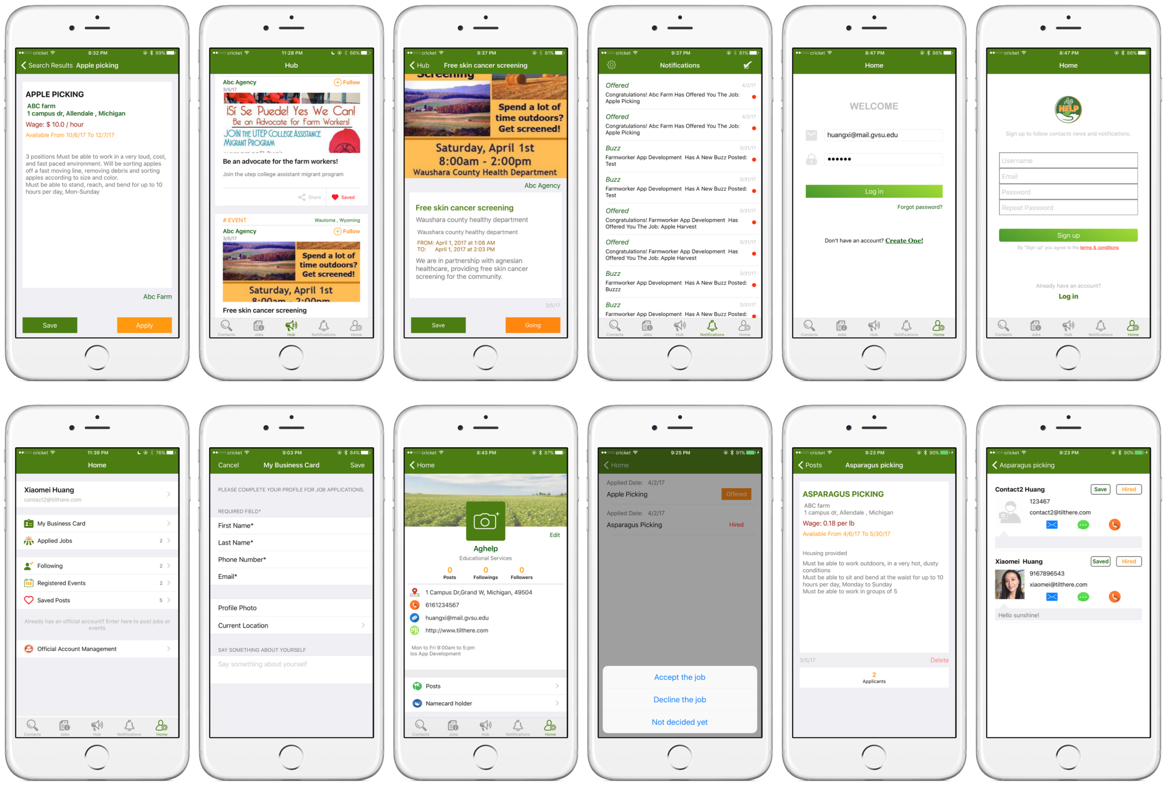


Figure 2 – AgHelp! Screens

Adobe Illustrator and Photoshop were used for icon design.

Figure 3 – AgHelp! App icon

2. User engagement and user login

Unregistered users are allowed to view service providers' information, job news, etc. But if they want to apply for jobs, subscribe to notifications or post news, users will be asked to register and log in. There are two levels of user authorities: basic account and official account. In order to prevent mendacious news spreading, only official accounts will be allowed to post jobs and news.

Pushing notifications is one of the key features of this app. There are five types of notifications offered in this app: 1, Notifications on job updates from any states that the user is interested in; 2, Notifications on news or jobs from any official account the user is following; 3, Notifications on new applicants for jobs the growers posted (official accounts only); 4, Notifications on applied job status changes to applicants; 5, System notifications. These features are implemented by third party API Integration.

3. Use of location data

Migrant farmworkers travel across the country. It is essential for them to get location-based information. A current-location based search function offered in this app can help them locate the nearby resources much easier. Geopoints from parse server plays an important role for this purpose. By adding service providers' PFGeoPoint to a PFObject allows queries to take into account the proximity of an object to a reference point. This allows users to easily find resources nearby.

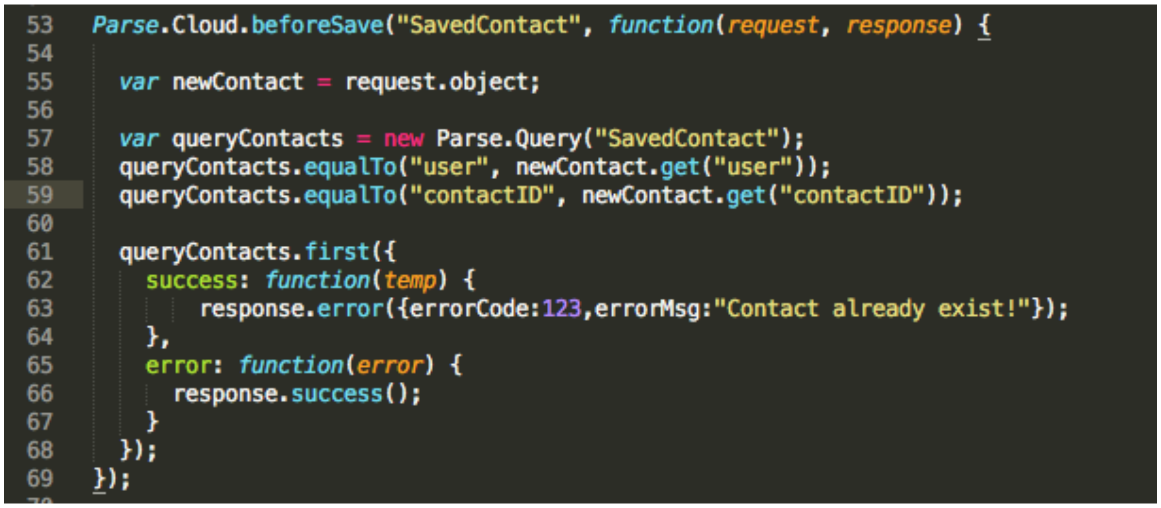
4. Parse server and data storage

Parse Server is the open source version of Parse that is supported by a strong community of open-source developers. It can be self hosted by users. It uses MongoDB to store data and stores files in JSON format as a backup. It supports cloud code and push notifications, has no file-storage restrictions, and provides control over backup, restore, and database indexes. All of these features make it one of the best platforms to develop new iOS or Android apps or API's.

Considering that farmworkers might not have Internet connection when they're traveling or working in the field, this app needs to allow users to find providers' information offline. A Parse local datastore feature lets the data be pinned to the local and be accessed anytime without Internet requirement. UserDefaults is also used for keeping the app data persistence.

5. Cloud code and app security

Unlike client code running on users' devices that may have been tampered with, Cloud code is guaranteed to be the code that you've written, so it can be trusted with more responsibility. ‘BeforeSave’ triggers in cloud code and is used to validate user inputs. It can also be used to prevent record duplication.



```javascript
Parse.Cloud.beforeSave("SavedContact", function(request, response) {

  var newContact = request.object;

  var queryContacts = new Parse.Query("SavedContact");
  queryContacts.equalTo("user", newContact.get("user"));
  queryContacts.equalTo("contactID", newContact.get("contactID"));

  queryContacts.first({
    success: function(temp) {
        response.error({errorCode:123,errorMsg:"Contact already exist!"});
    },
    error: function(error) {
      response.success();
    }
  });
});
```

Figure 4 – Example of ‘beforeSave’ Cloud code

Cloud code is also used for sending push notifications. In general, clients can't be trusted to send push notifications directly; because they could modify the alert text, or push to people they should not be able to. So cloud code function is necessary to validate the data before sending a push.

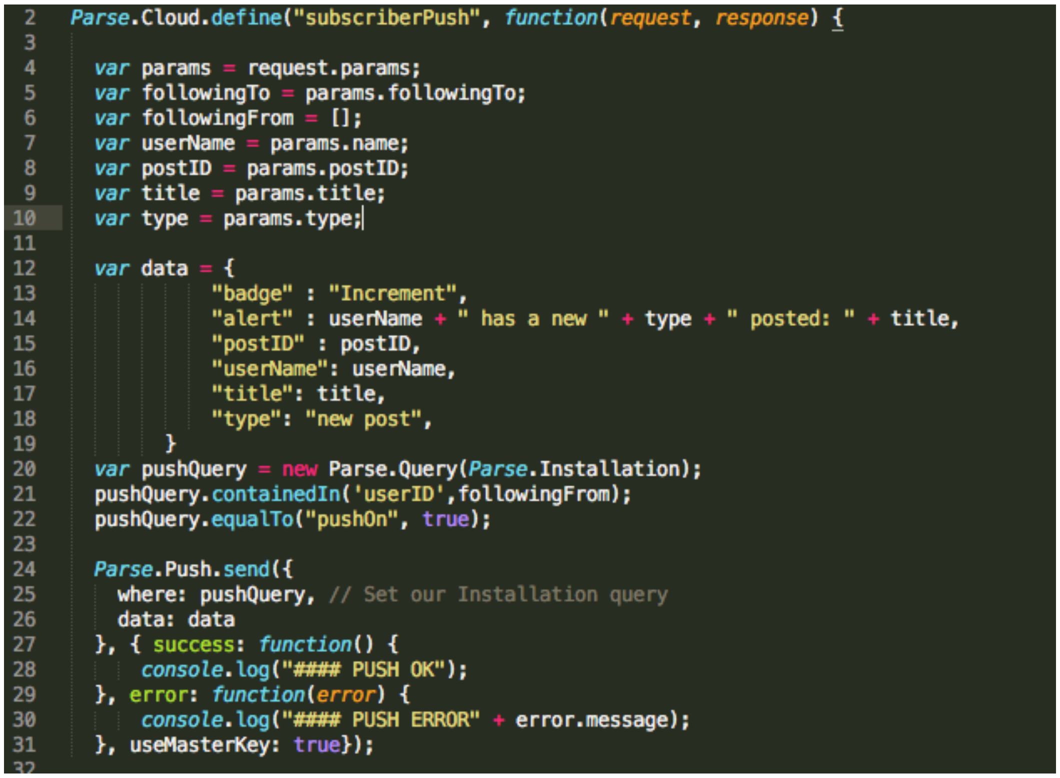
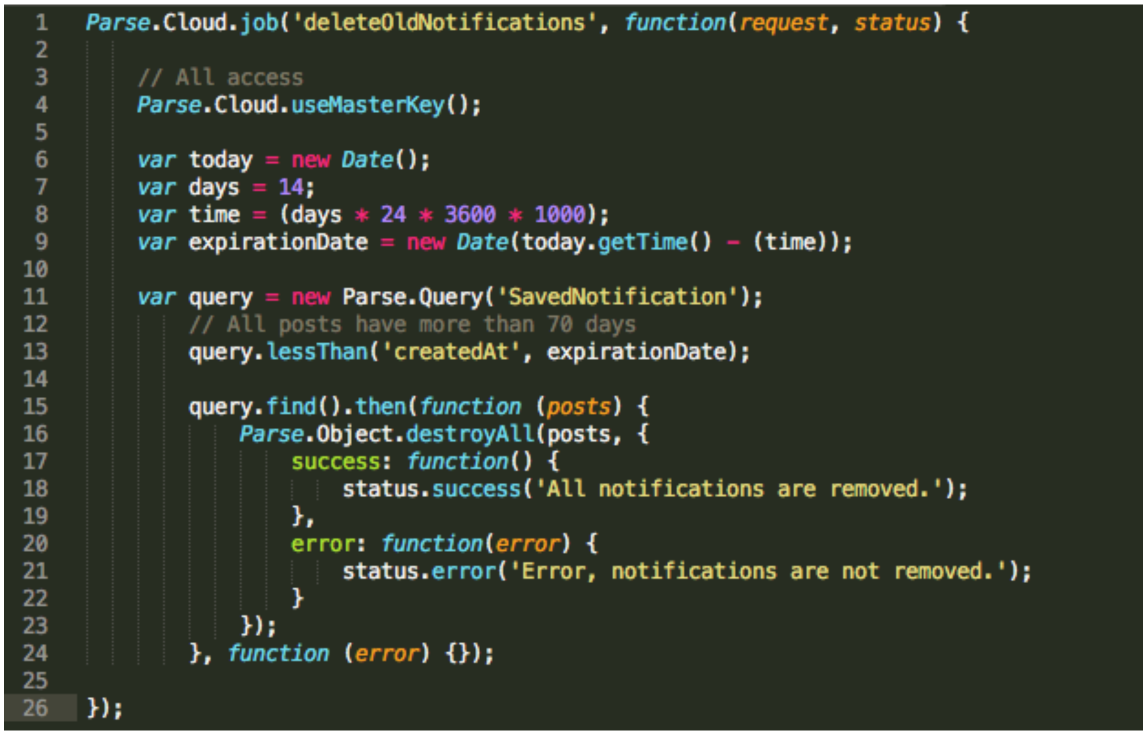

```
Parse.Cloud.define("subscriberPush", function(request, response) {

  var params = request.params;
  var followingTo = params.followingTo;
  var followingFrom = [];
  var userName = params.name;
  var postID = params.postID;
  var title = params.title;
  var type = params.type;

  var data = {
            "badge" : "Increment",
            "alert" : userName + " has a new " + type + " posted: " + title,
            "postID" : postID,
            "userName": userName,
            "title": title,
            "type": "new post",
        }
  var pushQuery = new Parse.Query(Parse.Installation);
  pushQuery.containedIn('userID',followingFrom);
  pushQuery.equalTo("pushOn", true);

  Parse.Push.send({
    where: pushQuery, // Set our Installation query
    data: data
  }, { success: function() {
      console.log("#### PUSH OK");
  }, error: function(error) {
      console.log("#### PUSH ERROR" + error.message);
  }, useMasterKey: true});
```

Figure 5 – Example of using Cloud code to push notification

Background jobs in Parse server allow users to schedule Parse server cloud code functions to be executed at periodic intervals. Background jobs can run seamlessly in the background without user input, so it helps to reduce manual effort. For example, we keep user notifications on the server for two weeks. Since these messages are time-based, it's not necessary to keep them forever. Background jobs can be used to clear these out-of-date records from the server.



```
Parse.Cloud.job('deleteOldNotifications', function(request, status) {

    // All access
    Parse.Cloud.useMasterKey();

    var today = new Date();
    var days = 14;
    var time = (days * 24 * 3600 * 1000);
    var expirationDate = new Date(today.getTime() - (time));

    var query = new Parse.Query('SavedNotification');
        // All posts have more than 70 days
        query.lessThan('createdAt', expirationDate);

        query.find().then(function (posts) {
            Parse.Object.destroyAll(posts, {
                success: function() {
                    status.success('All notifications are removed.');
                },
                error: function(error) {
                    status.error('Error, notifications are not removed.');
                }
            });
        }, function (error) {});

});
```

Figure 5 – Example of background job

Results, Evaluation, and Reflection

This app was fully implemented by the end of March, 2017 and has submitted for TestFlight Beta Testing. By April 18, 2017, 30 agencies were enrolled to test the app and more will be coming in. The beta testing will carry on for around 50 days. Then this app is expected to launch at Apple store by the end of June, 2017.
To assess the app performance and user behaviors, Google Analytics is utilized.

Conclusions and Future Work

This app has caught the attention of many people who work in agriculture and technology commercialization. Vegetable Growers News, the No. 1 choice among the nation's vegetable growers for industry news and information, stated that the app "provides the opportunity to find resources they need right at their fingertips. .... This is a very high-value population that comes into our region, and we're happy to have them here. They're providing an invaluable service to be sure to conduct our local food harvest properly. It becomes a democratization of that industry – allowing the grower to be able to access people, and people to access growers."

During the Customer Discovery Program, we noticed that the farmworkers are mainly using Android phones. So the Android version is the most priority to develop after the iOS version.

Another plan for future work is providing a Web portal for official users. This will help them to manage posts and job applicants more conveniently.

There is still a lot of work to be done, such as allowing farmworkers to rate the service providers, and report occupational injury or abuse. Or even have a button for emergency help. I hope this app could really help people in need in the near future!

Acknowledgment:

First of all, I would like to sincerely acknowledge my advisor, Professor Engelsma, for his encouragement and guidance, without which this app would not have been developed.
I thank Mr. Feliciano Paredes, who brought this brilliant idea for building this app. Thanks for all the efforts he has made to help farmworkers and others.
I also thank Dr. Linda Chamberlain for offering the opportunity of the Customer Discovery Program and helping us building the business model.
Finally, I am eternally grateful to my husband, my two lovely children, my parents, my friends, and my teachers at GVSU. Your love and support keep me moving forward!
Thank you indeed!

Tools:

1. Xcode
2. Sublime
3. CocoPod
4. Sketch
5. Adobe illustrator
6. Adobe Photoshop
7. InVision
8. iTunesConnect / TestFlight

Programming languages:

1. Swift
2. JavaScript

Bibliography

1. Parse server. IOS guide. http://docs.parseplatform.org/ios/guide/
2. Simon Ng, Intermediate iOS 10 Programming with Swift, https://www.appcoda.com/
3. https://www.raywenderlich.com
4. Apple developer guidelines, https://developer.apple.com/app-store/guidelines/
5. The Swift Programming Language, Swift 3.1 edition, https://itunes.apple.com/us/book/the-swift-programming-language-swift-3-1/id881256329?mt=11
6. Vegetable Growers News, April 2017, volume 51, issue 4