
# *EvenStreamin Mobile Application*

Audrey Lincoln
Computer Science Department
Pacific University

5/8/2021

**Abstract**

Inefficient watering systems cause farmers to spend hours each day checking for water waste. Farmers get stuck choosing between wasting time - or potentially wasting water. The mobile application, *EvenStreamin*, gives farmers remote access to turn their irrigation systems on or off, making it so farmers never have to choose. *EvenStreamin* utilizes the *RainCat* mathematical model - which is a precision irrigation model developed for this project. With this, *EvenStreamin* can customize the water application based on variables such as soil type, wind, and humidity. *EvenStreamin* runs on iOS and Android devices, and was developed using Xamarin.Forms.

*Keywords:* Center Pivot Irrigation; Precision Agriculture; Xamarin; Azure; ArcGIS.

# Contents

# 1 Introduction

Farming began around 12,000 years ago [1], and it is estimated that the first irrigation system was introduced 8,000 years ago in the Middle East's Jordan Valley [2]. Since then humans have continued to bring technological advancements into the agriculture industry. A few advancements include introducing genetic engineering in crops, or programming autonomous tractors. But, as with any industry, there is always more room for improvement. The negative effects of farming are prevalent: aquaphor and river depletion, erosion of coastlines, habitat destruction, and excessive amounts of chemical fertilizers [3]. It is critical that agriculture advancements have an emphasis on sustainability.

## 1.1 *Problem Statement*

Growing up in a rural farming town, I saw the benefits of precision agriculture technology. However, this technology is not always the most efficient. By working on farms, I realized two issues:

1. Incorrectly set center pivots lead to flooding or drying out a field.

2. Time and resources are wasted on tasks that could be automated.

The issue of water is critical for farmers since agriculture accounts for "approximately 80 percent of the Nation's consumptive water use" [4]. With this in mind, a farmer must choose precise watering systems. Center pivot irrigation systems are the most popular sprinkler system in the world. A center pivot's uniform water application is the reason for their 75%-90% efficiency rate [5]. However, uniformity is easily affected by operating conditions and environmental factors. A non-uniform application can result in overwatering or underwatering areas of a field [6]. Overwatering wastes water and causes root damage, while underwatering stresses crops. Both options lead to poor crop yields [5].

As for the second issue, my work on the farm consisted of hand checking fields. Entire workdays could be spent driving to each field and documenting crop growth and moisture levels. Making sure the watering systems were working correctly was one of the biggest

reasons for checking each field by hand. However this wasted fuel for driving, as well as time.

## 1.2 *Solution Statement*

My senior capstone project attempts to address both issues through the *RainCat* center pivot model, and the *EvenStreamin* mobile application.

### 1.2.1 *RainCat Center Pivot Mathematical Model*

The *RainCat* center pivot is a mathematical model that targets the issue of water waste. The model optimizes each drop head sprinkler to release the least amount of water that is needed to evenly water the soil below it. This is accomplished by having the rate of water being applied equal the rate at which the soil can absorb water. Once the soil can no longer absorb anymore water, then the sprinkler head will shut off. This involves the following components:

- **Instantaneous Application Rate**: The peak intensity of water application at a specific point. Figure 1 shows how the instantaneous application rate increases the further from the central pivot point the sprinkler is [6].
- **Instantaneous Infiltration Rate**: The rate at which water can enter the soil. The infiltration rate is not the same for all parts of the field, and it changes over time as well [7].

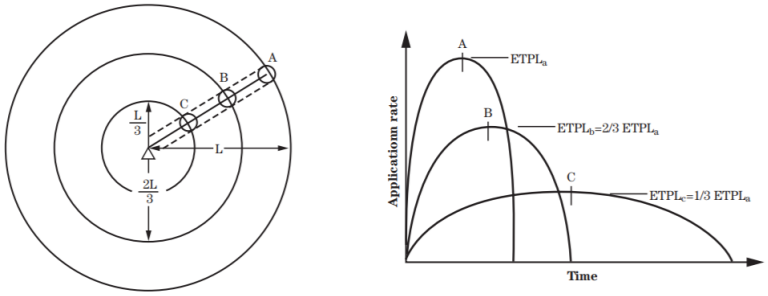


Figure 1: Diagram of water application rates at different points along the center pivot [6]

### 1.2.2 *EvenStreamin Mobile Application*

The *EvenStreamin* mobile application allows farmers to remotely monitor their fields. The app is cross-platform, allowing both Android and iOS users to download the app. Specific functionality that will reduce time and resources spent checking fields manually includes:

- Running and stopping a pivot remotely.
- Getting relevant statistics, such as the stop time of the pivot and how much water was used.

### 1.2.3 *The Project: EvenStremain & RainCat*

The complete project is then a combination of the *RainCat* mathematical model, and the *EvenStreamin* mobile application. To get real time watering statistics, the *RainCat* mathematical model will be implemented as an algorithm within the *EvenStreamin* app. This provides farmers with accurate information about how the field is being watered, and how much water the model predicts the sprinkler system will use.

## 1.3 *Background Information*

Agricultural terminology is prevalent within this project, refer to the Glossary for a table of all terminology. *EvenStreamin* is a precision agriculture mobile application. **Precision agriculture** is a "technology-based management system that collects and organizes data to optimize profits, sustainability, and protection of the environment" [8]. The optimizations of the project occur within the center pivot irrigation system. **Center pivots** irrigate fields in a circular pattern around a central pivot point [5]. The **central pivot point** is a stationary structure connected to a water supply - this is what the main water pipe rotates around. For reference use Figure 2.

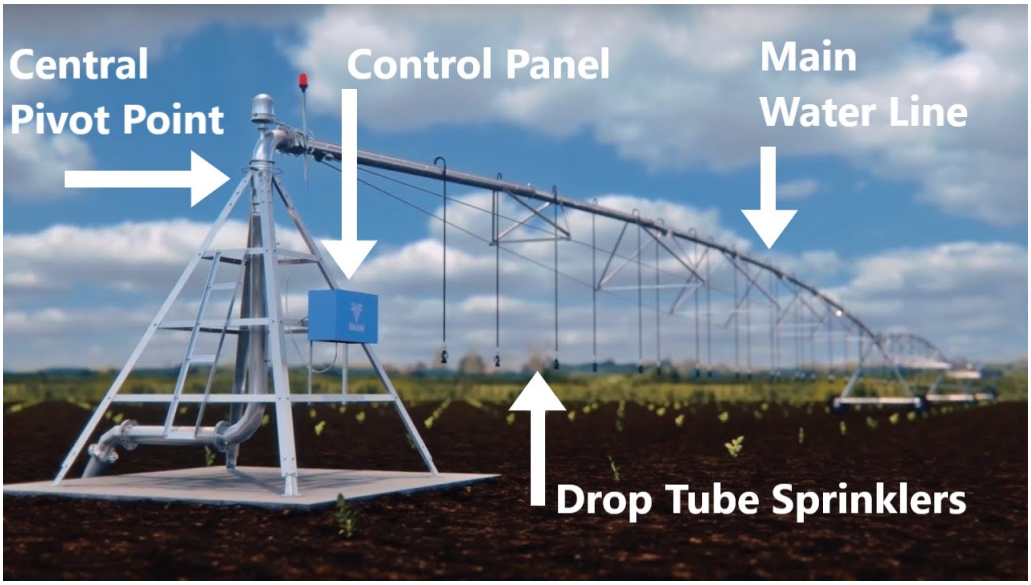


Figure 2: Diagram of a Center Pivot Irrigation System [9]

Spaced throughout the length of the main water pipe are drop tube sprinklers. **Drop tube sprinklers** hang below the pipe and are used to water individual sections of the field. The distance between each sprinkler and the type of sprinkler head will vary. A center pivot is controlled through a piece of hardware called the control panel. The **control panel** gives the center pivot commands on starting, stopping, and changing direction.

## 1.4 *Related Works*

Other farming mobile apps and precision center pivots already exist within the agriculture industry. I will discuss the following:

- Valley 365
- SmartIrrigation

Valley Irrigation is a big inspiration for my project. Their Valley 365 product is a remote crop management mobile app. Similar to *EvenStreamin*, the app acts as a command and control center for all Valley Irrigation center pivots [10]. This mobile app has an abundance of other functionality such as:

- Viewing data on soil moisture, crop type, crop development, and weather information.
- Irrigation recommendations.
- Computer vision to spot health concerns in the fields.

SmartIrrigation, is a company that has developed multiple, crop-specific mobile applications that give users irrigation tips [11]. Since it is crop-specific, the calculations and predictions are more accurate. However, it might be a hassle for farmers that grow multiple crops.

# 2 Design

## 2.1 *User Requirements*

The main type of users will be farmers that water their fields via center pivot irrigation. All users will interact with the application the same way:

- Making accounts.
- Adding fields to their account.
- Running and stopping pivots remotely.
- Reviewing watering statistics on how much water is being used.

## 2.2 *User Experience*

An assumption is made that users have a basic understanding of mobile apps. However, if the app is hard to use it will defeat the purpose of being made. Therefore, the app needs to be:

- *Efficient*, the users should be able to navigate the app with ease.
- *Consistent*, the users will want to access different information in a similar manner.
- *Useful*, the utility of the app is what will make users more likely to keep using it.

## 2.3 *User Interface*

As for the actual user interface, many styling choices were made to make the app user friendly. The example screenshots below show the iOS interface on an iPhone 12 Pro Max iOS 14.4 simulator. To see the Android and iOS implementations side by side refer to Appendix 1.

### 2.3.1 *Efficient*

The first step to making the app meet the efficiency usability goal was creating a minimal, easy to navigate, design. Pages that are frequently used require only a few taps to navigate to. The home page is the first example of this, as seen in Figure 3. All the main tasks a farmer wishes to complete are accessible in one area. Each button is large and labeled with the exact page that it will navigate the user to. Icons are used to be visually appealing to the user.

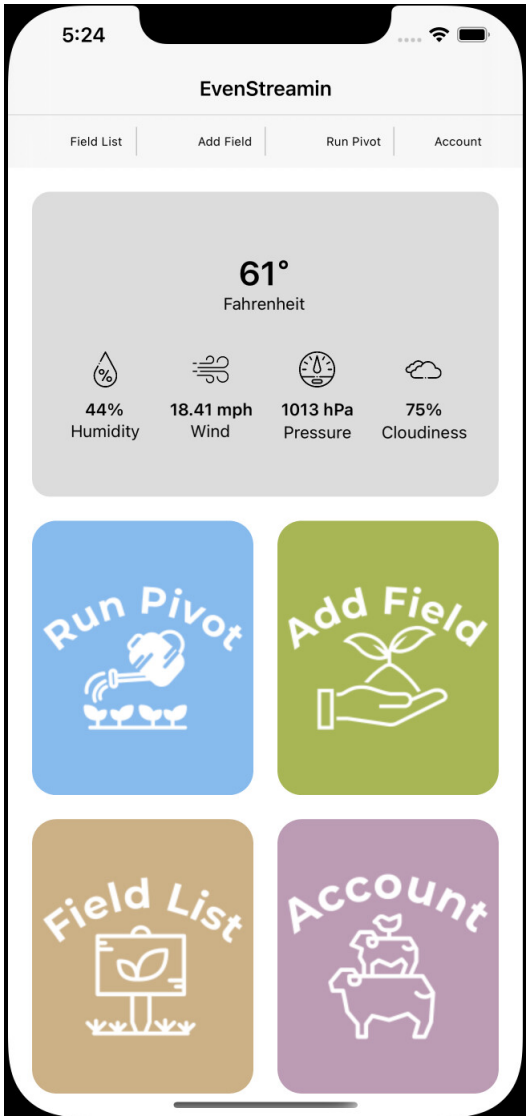


Figure 3: Home Page

Another key to efficiency is making navigation intuitive for all users. To accomplish this goal, the app uses standard navigation components that users are already familiar with. This includes the tab bar for iOS and the navigation drawer for Android. Figure 4 shows the tab bar of multiple different pages. The pages that appear in the tab bar are the most frequently used pages.


Figure 4: Toolbar

### 2.3.2 *Consistent*

The user interface provides consistency by reusing styling concepts. Typefaces, buttons, and labels are all visually consistent throughout the app. The design reuses components and behaviors. The reusability allows users to recognize visual cues, rather than having to think about what to do. Figures 5 and 6 show that the pages for creating entities - whether that be an account or a field - are consistent. Each placeholder is a light grey and framed by a box. The buttons also use the same color scheme to signal canceling the form or submitting it.

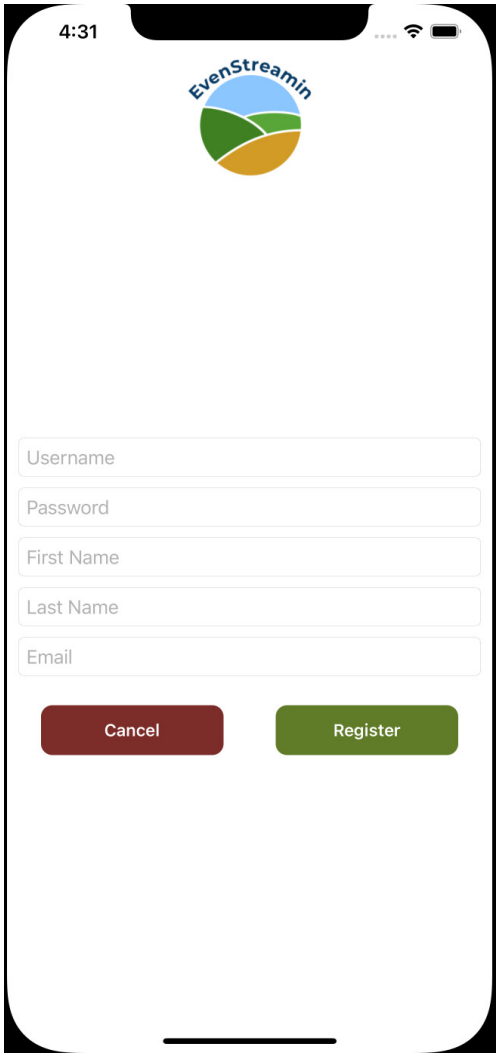


Figure 5: Registration Page

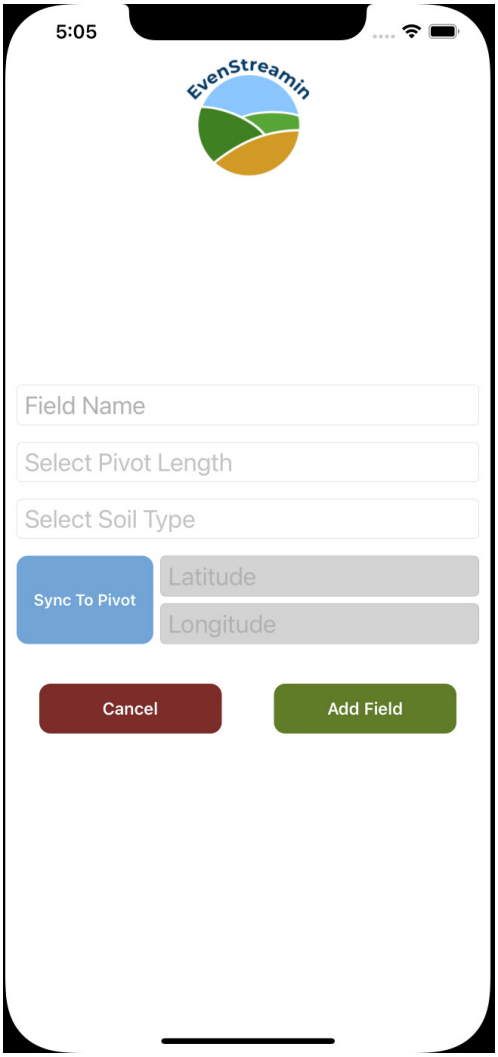


Figure 6: Add Field Page

Modifying these entities also has the same format. Figures 7 and 8 display a popup screen with placeholders of the current values. Writing over these placeholders allows users to edit their accounts or fields.

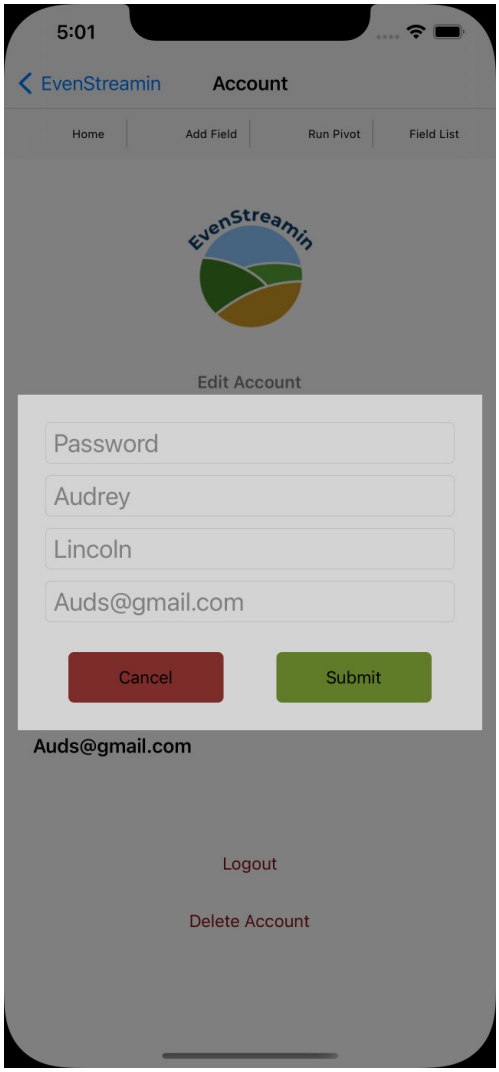


Figure 7: Account Edit

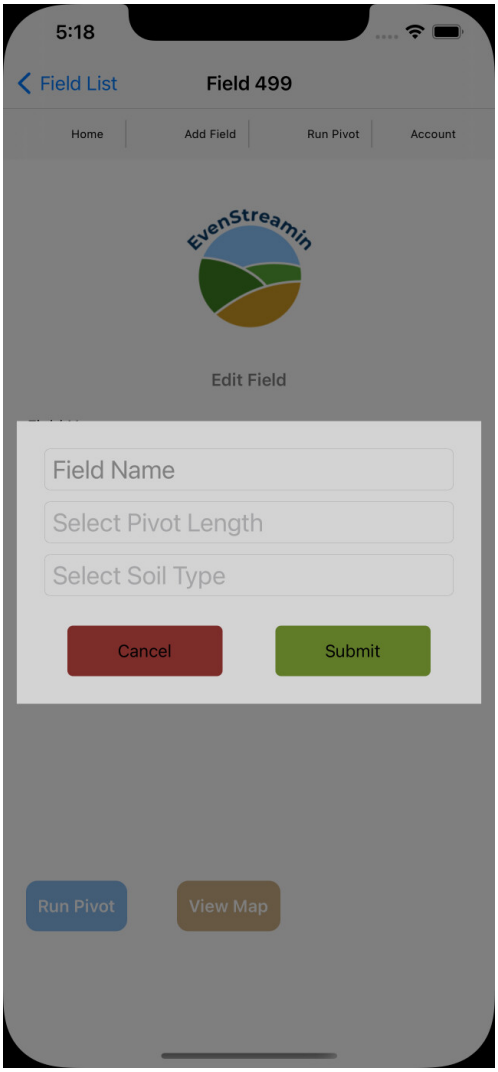


Figure 8: Field Edit

Information is displayed in a consistent manner as well. Figures 9 through 14 show that account and field pages use the same styling choices. All of the information is preceded with an italicized header, and the information itself is bolded.

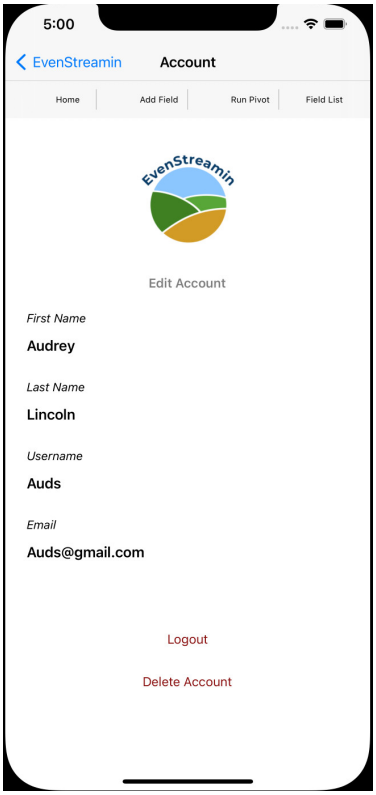


Figure 9: Account

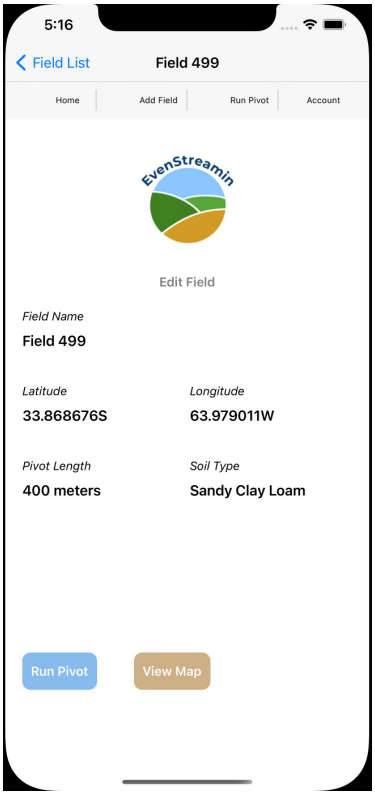


Figure 10: Stopped Field

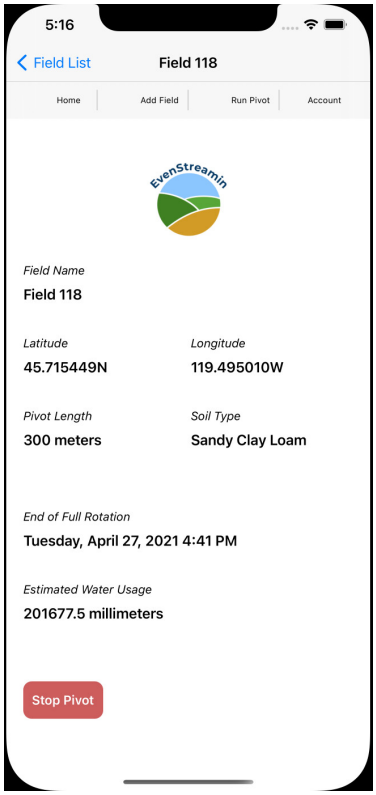


Figure 11: Running Field

### 2.3.3 *Useful*

There are two main components that make the application useful for farmers:

1. Being able to start and stop pivots remotely.
2. Accessing field-related watering statistics.

Farmers are more likely to start pivots. This means that there must be multiple places to start a field's pivot. The 'Run Pivot' page is easily accessible from the home page and tab bar. Figure 12 is the 'Run Pivot' page. The 'Run Pivot' page has a menu of all stopped pivots. This design is useful for farmers that already know the name of the field they want to run.

The second way to turn on a field's pivot is on the 'Stopped Pivot' page. Figure 13 is the 'Stopped Pivot' page. This is useful for farmers that have already navigated through the 'Field List' page in search of a specific field.

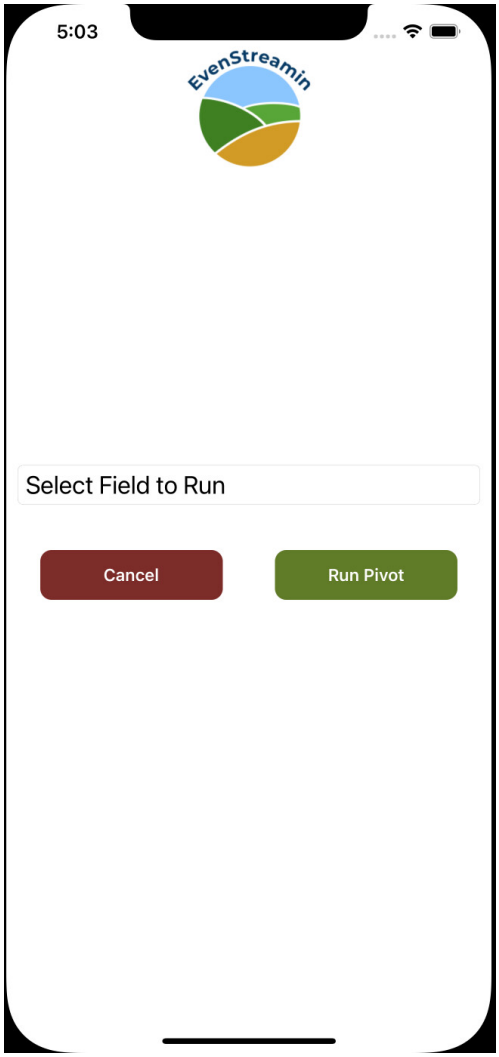


Figure 12: Run Pivot Page

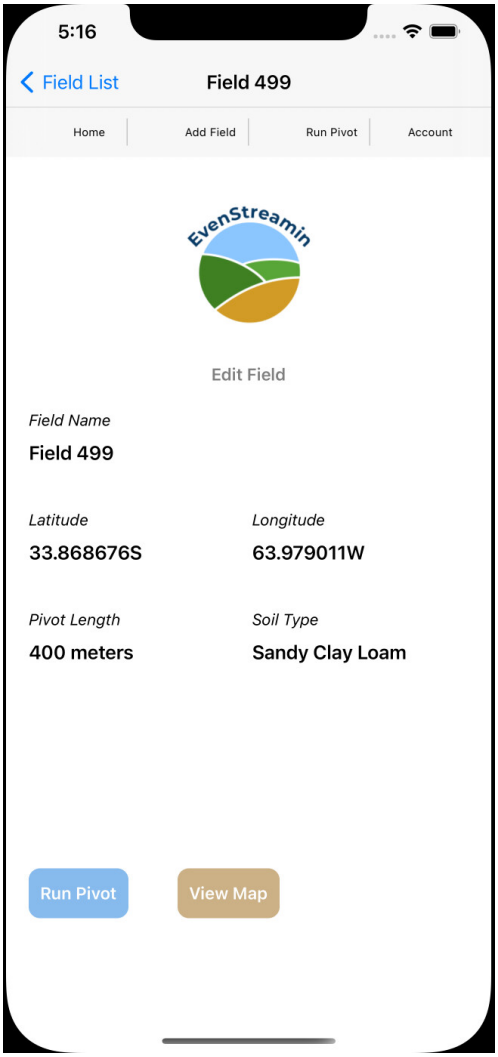


Figure 13: Stopped Field Page

A farmer is less likely to shut off a pivot, and would not want to shut it off accidentally. Therefore, the functionality to turn off the pivot exists in one place. The user must navigate to the 'Running Pivot' page of a specific field in order to locate the 'Stop Pivot' button. Figure 14 shows the 'Running Pivot' page.

*EvenStreamin* is also useful for its watering statistics. The field pages with running pivots display the *RainCat* water usage statistics. Figure 14 also shows the water usage report.

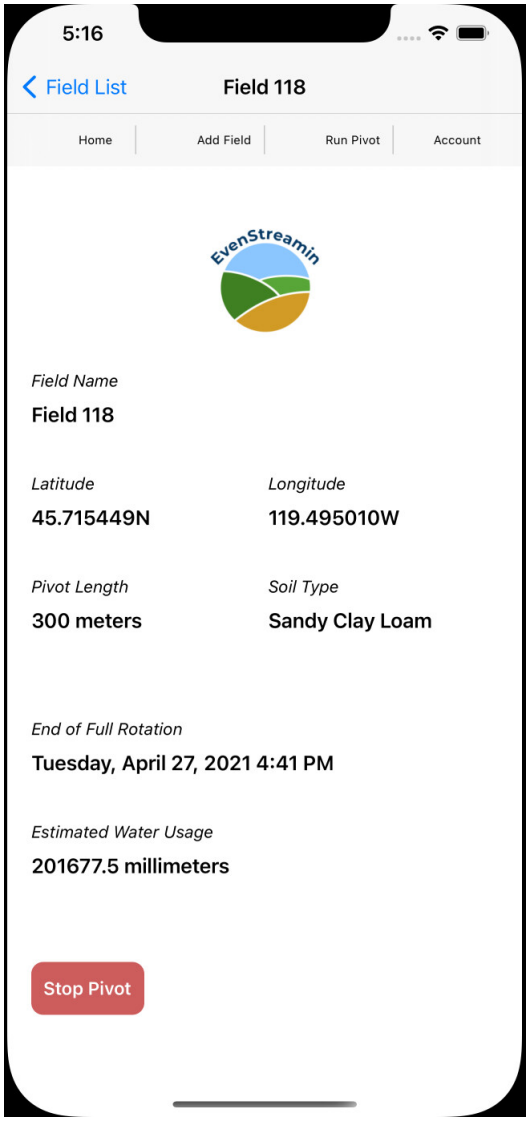


Figure 14: Running Field Page

# 3 Project Components

The five main components of the *EvenStreamin* mobile application include:

1. Cross-Platform Mobile Framework
2. MVVM Design Pattern
3. Azure
4. ArcGIS
5. *RainCat*

All five components are explained in their respective sections below. Along with these components is the development environment. All of the development has been done using the Visual Studio 2019 Integrated Development Environment (IDE).

The majority of the work has been on a PC, which uses a Windows operating system. Since the mobile application is cross-platform, Apple build tools are required to do development on an iOS device. Therefore, a Mac device must be paired with the Visual Studio IDE running on a PC. This allows me to use the Mac as a build server, without having to switch between machines. All of the revision history has been saved on GitHub.

## 3.1 *Xamarin*

**Cross-platform mobile development** is the creation of software that is compatible with multiple mobile operating systems. There are currently two major smartphone operating systems: Android developed by Google, and iOS developed by Apple [12]. In order to make *EvenStreamin* more accessible, it needed to exist for both platforms.

To do cross-platform development on Visual Studio 2019, I installed the 'Mobile development with .NET' workload. **Workloads** are used to modify Visual Studio and contain tools needed for the specific programming languages or platforms being used [13].

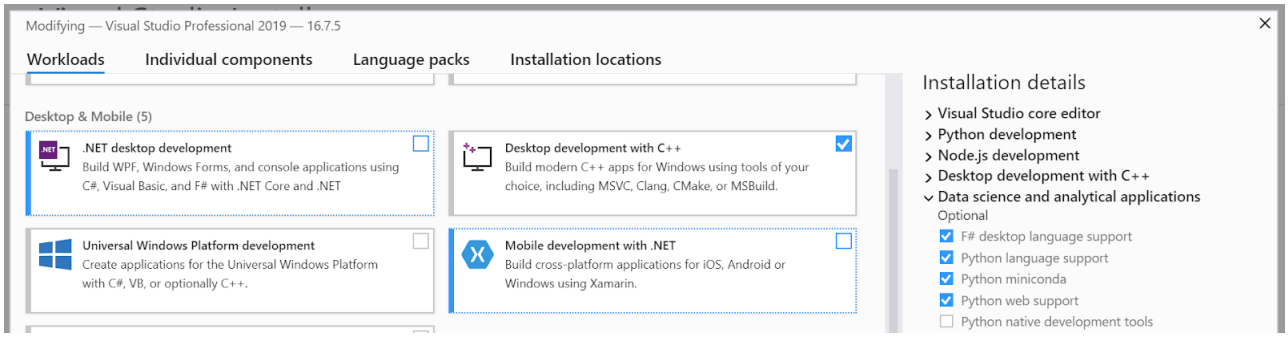


Figure 15: Mobile Development with .NET selection [13]

This workload specifically utilizes **Xamarin**, which is an open-source framework for cross-platform mobile development [13]. Xamarin is built on top of a .NET framework - which handles memory allocation and garbage collection [13]. **Xamarin.Forms** is an open-source User Interface (UI) framework on top of Xamarin. This allows developers to build Android and iOS applications from a single codebase [13]. The shared codebase is written such that:

- Xamarin gives developers the ability to write business logic in C#. C# is a modern object-oriented programming language [14].
- Xamarin.Forms allows for UI elements to be written in both Extensible Application Markup Language (XAML) and C#. XAML is a declarative language mainly used for designing graphical user interfaces (GUI) [15].

Typically 80-90 percent of code can be shared across platforms [13]. However, developers are not limited to the shared codebase. Xamarin.Forms also includes native development if certain functionalities cannot be completed through shared code.

How does a shared codebase work on different operating systems? Refer to Figure 16 for reference. Xamarin applications "compile into native application packages" of the device being deployed [13]. More specifically, Xamarin.Forms uses platform renderers to convert the UI elements into Android or iOS native controls. This is how native performance and looks are achieved for each platform.

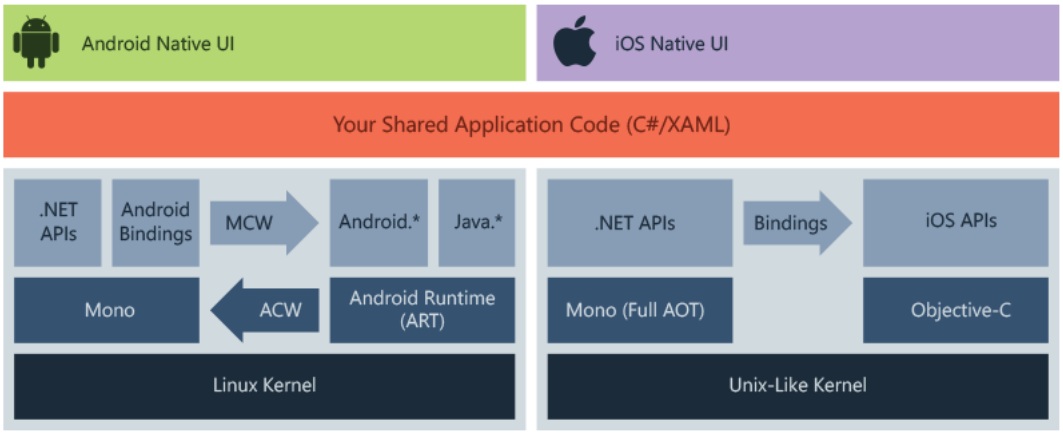


Figure 16: The architecture of a cross-platform Xamarin.Forms application [13]

I chose Xamarin over other cross-platform frameworks because of the mass amounts of documentation and sample code that Microsoft provides users. There is also a large community of developers that use Xamarin; this means there are many answers to issues that I have encountered throughout the year.

## 3.2 *MVVM Design Pattern*

Design patterns help structure applications to make them more maintainable and reusable [16]. Model-View-ViewModel (MVVM) is a design pattern that separates the business logic from the graphical user interface (GUI) logic. Refer to Figure 17 for the component structure. MVVM is broken up into three groups [16]:

- **Models**: The Model encapsulates the application's data. It is often referred to as the domain object [17]. The Model holds information but does not define behaviors or services that manipulate that information.
- **Views**: The View represents the visual elements, and controls what the user sees on the screen.
- **View Models**: The ViewModel is the connection between the View and the Model. The ViewModel exposes Models, properties, and commands that the View binds to. This binding allows the View and Model to interact, but still be separate.

The MVVM design pattern is popular for mobile applications because of the strict separation between all of the elements. Designers can work on the UI without having to write the code that manipulates the Model. At the same time, developers can work on the ViewModel and Model components.

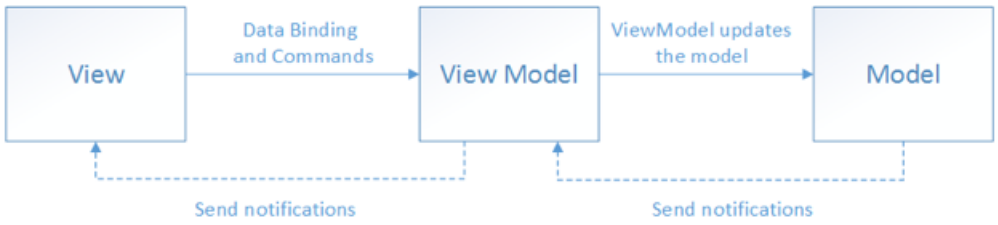


Figure 17: Model-View-ViewModel components [17]

## 3.3 *Azure*

The *EvenStreamin* mobile application has account and field information that persists even after the user has closed or logged out of the app. This information needs to be easily accessible. Therefore, a database is a necessary component of the mobile application. A database is an "organized collection of structured information" which is controlled by a database management system (DBMS) [18].

In the fall semester I utilized a SQLite database engine. SQLite is serverless, meaning the database files are stored on the disk of the application [19]. This was useful because I did not have to write any requests to a server. However, since all files are local to the one device, that means the user cannot log into their account anywhere else. If they were to get a new phone then their account information would be lost. Therefore, I decided to move to the industry standard of cloud computing with Azure.

**Cloud computing** is the delivery of computing services, where a network of remote servers are used to analyze, manage, and route data [20]. This is all done over the internet, which makes cloud computing faster and more scalable than on-premise computing [21]. Rather than managing files and services on a local device, you will be doing the same over the internet. [20].

Cloud computing consists of three connected components [20]. Refer to Figure 18 for

the core components of cloud computing.

1. The cloud based resources such as storage, servers, and virtual desktops.
2. Network connecting devices such as routers and switches for data to pass through.
3. The end user, who can access the data from any type of device.

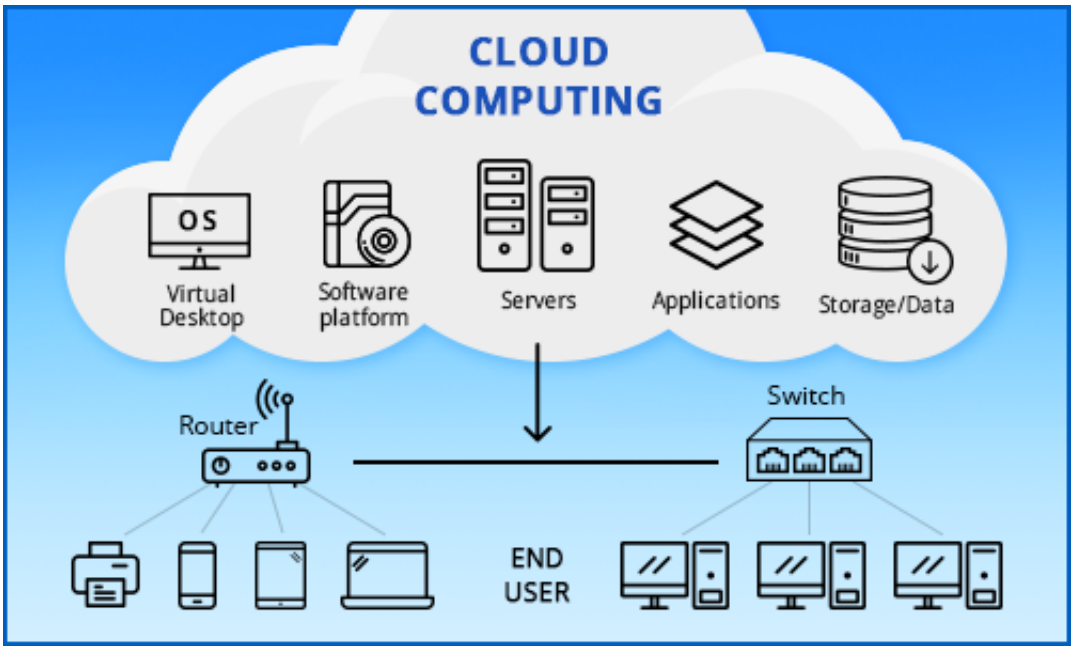


Figure 18: Cloud Computing diagram [20]

With a working explanation of cloud computing, Azure will now make more sense. **Azure** is a cloud computing platform owned by Microsoft. To refer back to Figure 18, Azure is a platform that provides the cloud based resources and services. Azure has over 200 services [22].

Although there are many products, *EvenStreamin* only needs Azure as a cloud-based storage service to replace the SQLite database. This is accomplished by creating a backend service and web API using Azure Functions bound to Azure Table Storage.

**Azure Functions** is an event-driven serverless compute platform [23]. By event-driven, it means there is some kind of trigger that will run the function. Serverless does not mean that there is no server, rather it means there is *no server the developer needs to worry*

*about* [24]. Azure has many different types of functions - for this project I utilize the **HTTP Trigger Function**. This function will run whenever it receives an HTTP request [23]. In this function I specify what should be done for GET, POST, PUT, and DELETE requests. This means that the API has full CRUD capabilities. CRUD refers to the four basic operations that can be performed on database applications:

- Create
- Read
- Update
- Delete

Azure Functions are usually bound to some kind of Azure storage device. The storage devices allow data to persist within the cloud. The specific storage device I utilize is the **Azure Table Storage**. Azure Table Storage is a non-relational, key/attribute storage design. Refer to Figure 19 for an example of how the components are connected. The components consist of [25]:

- Storage account: The Azure storage account that allows access to different storage types.
- Table: A collection of different entities.
- Entity: Set of properties, similar to a database row.
- Properties: The name/value pair.

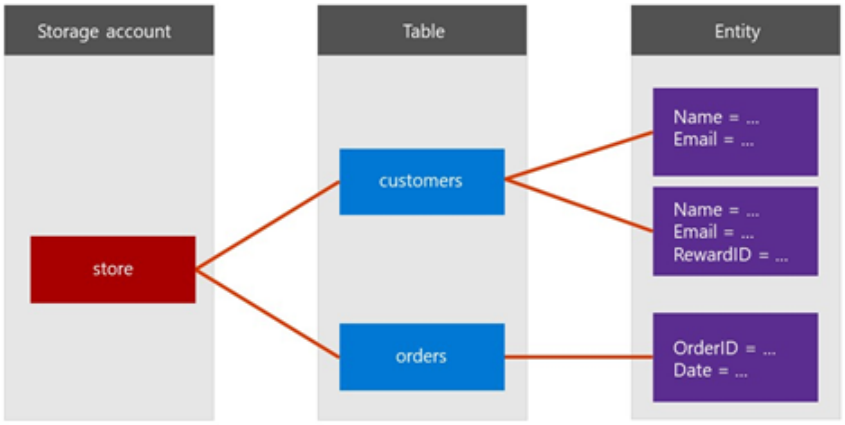


Figure 19: Components of Table Storage [25]

Once the Azure HTTP Functions, and Azure Tables have been bound together, then the backend service is complete. The app can now communicate with the newly written API through HTTP requests.

## 3.4 *ArcGIS*

**ArcGIS** is a geographical information system (GIS) software created by the Environmental Systems Research Institute (Esri). This software is used to visualize, edit, manage, and analyze geographic data [26]. ArcGIS works by creating maps with specified layers. These layers are made out of organized information. Multiple layers can be added to one map. For example, Figure 20 displays the 'Landsat 8 (Agriculture)' map. This map color-codes vegetation to analyze agricultural health in the area. The base map is a topographic layer, with the Landsat 8 layer on top.

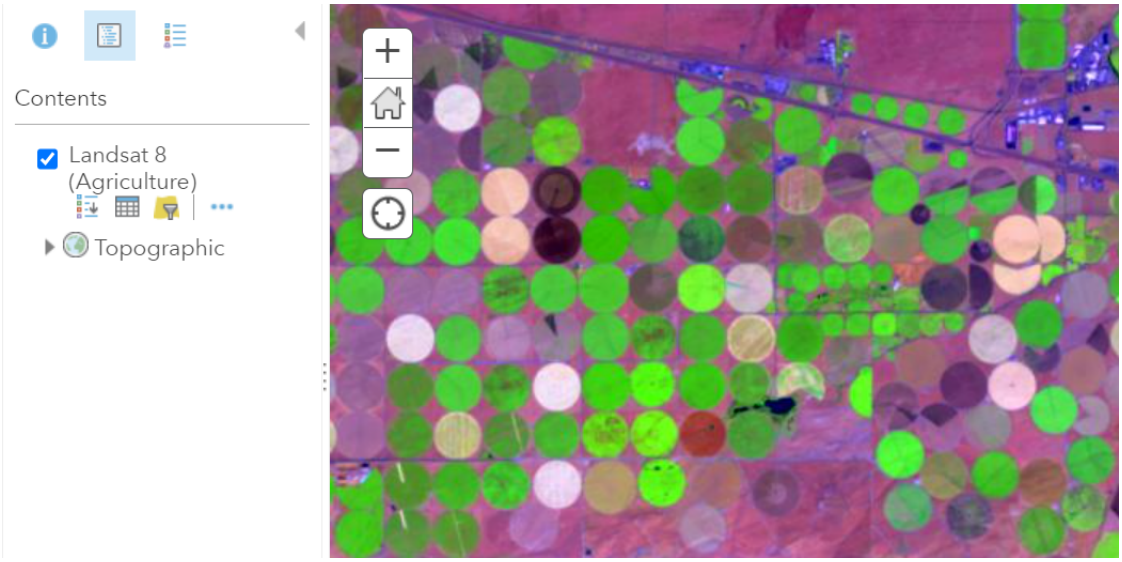


Figure 20: Landsat 8 (Agriculture) ArcGIS map [27]

The map in Figure 20 was made available through Esri's web map viewer; just one of the many software applications that make up ArcGIS. Having multiple products allows ArcGIS to be applicable for single person projects, or whole enterprises.

*EvenStreamin* utilizes **ArcGIS Online**, which is a web and cloud based platform. This allows ArcGIS users to share their maps and data without having to install an ArcGIS desktop application. ArcGIS Online is accessible via web, mobile, and desktop [26]. This means that developers can add ArcGIS Online's web maps and functionality into their mobile applications.

Development with ArcGIS Online is made possible through ArcGIS software development kits (SDK). An SDK contains a range of tools such as libraries, application programming interfaces (API), programming tools, and documentation. In order for *EvenStreamin* to utilize ArcGIS functionality, I incorporated the ArcGIS Runtime SDK for .NET into the application.

## 3.5 *RainCat*

The *RainCat* mathematics capstone goal was focused on creating an optimized watering pattern. Consider Figures 21 through 24. As the center pivot crosses over each area, the

individual sprinklers behavior changes. So an area that is boggy or fully ponded over should receive less or no water from the sprinklers above. However, sandy areas require a normal amount of water.

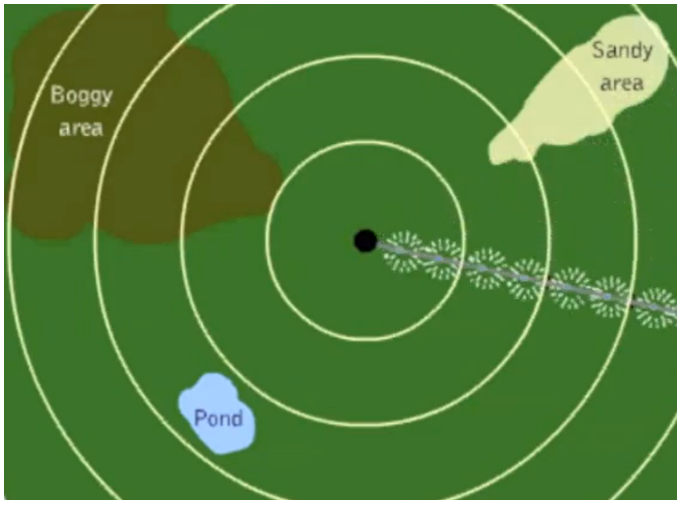


Figure 21: Ideal Center Pivot Set Up [28]


Figure 22: Boggy


Figure 23: Ponded


Figure 24: Sandy

Looking at the images it becomes clear: in order to create an ideal watering system there must be information on each sprinkler head, as well as information on the soil below it. Thus, the two biggest components to the *RainCat* Model are:

1. Soil Instantaneous Infiltration Rate
2. Sprinkler Instantaneous Application Rate

Refer to Figure 25. By graphing each equation it becomes evident that after the first intersection the instantaneous application rate begins to exceed the instantaneous infiltration rate. This is where over watering occurs.

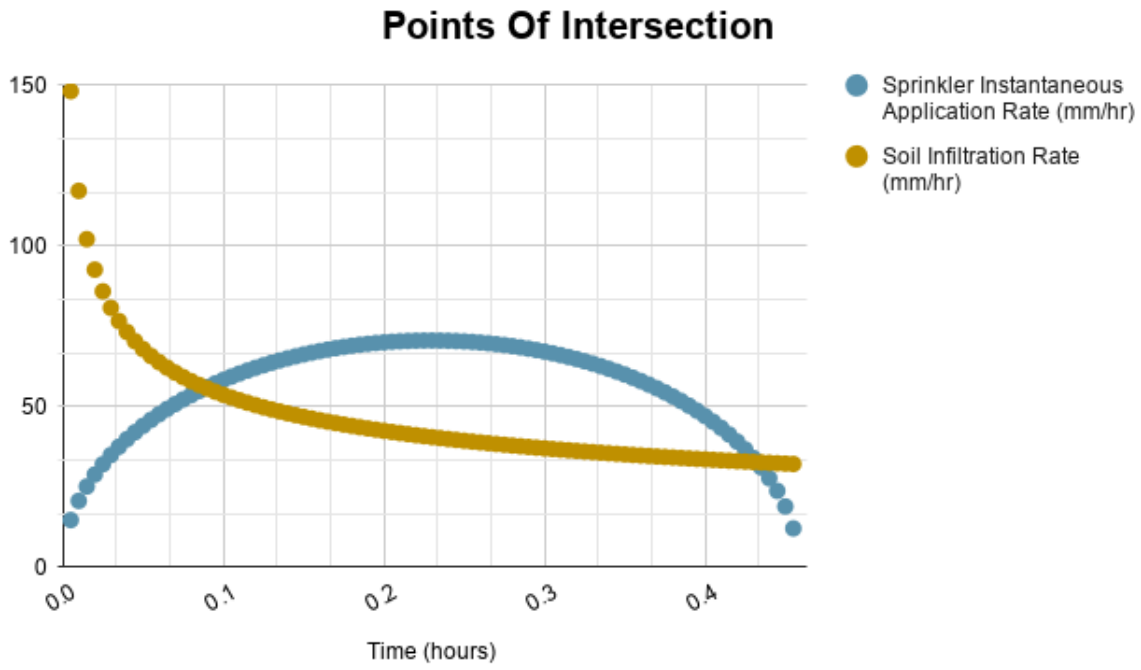


Figure 25: Algorithm Idea

Since the goal of the algorithm is to limit water waste, it is apparent that the application rate of each sprinkler need to be modified so that it does not over water the soil below. The solution is to turn the sprinkler off at the first intersection, and then back on at the second intersection. This process is repeated until the sprinkler has finished a complete revolution.

# 4 Implementation

The main implementation components of the entire year are the following:

- Mobile Development
- Weather API
- Azure
- ArcGIS Geolocation
- ArcGIS Visualization
- RainCat

Each are described in their respective subsections.

## 4.1 *Mobile Development*

*EvenStreamin* has been developed using the cross-platform framework Xamarin, with the MVVM design pattern. The framework allows the app to work on both iOS and Android operating systems.

The focus for fall semester was getting used to the new platform and languages, as well as creating the UI skeleton of the entire application. A UI skeleton meant that most of the pages for the application were designed and implemented - but possibly missing some key feature. This was done so that those features could be integrated into an already existing page.

The spring semester was spent adding in these features. Since the pages were already made, testing the new components was much quicker than I expected. An example of this would be the running field pages; these pages existed since the fall but the *RainCat* algorithm did not exist until the end of spring.

## 4.2 *Weather API*

One implementation component includes the *OpenWeatherMap* weather API. *OpenWeatherMap* provides access to many different API's, such as:

- Current Weather
- Hourly Weather
- 5 Day Forecast
- 16 Day Forecast
- Historical Weather Data

For the scope of this senior capstone, the 'Current Weather' API was the most applicable. The point of incorporating this API is to allow more information to be in one spot. If a farmer has to travel between multiple apps for information then they may stop using *EvenStreamin*.

With the addition of the API, the *EvenStreamin* home page now displays the current weather data for the exact location of the farmer. Figure 26 and Figure 27 display the weather information, and how the color of the weather box changes depending on the forecast.

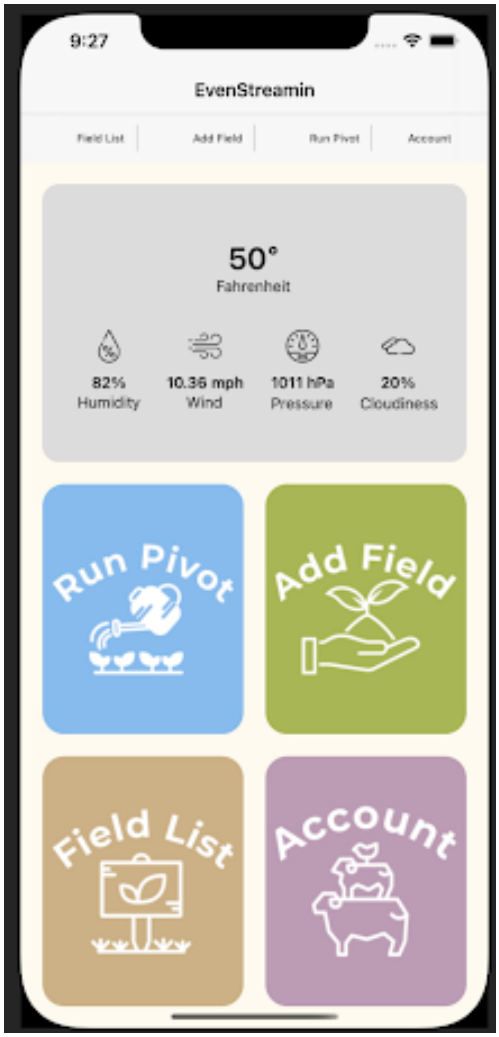


Figure 26: iOS Cloudy Weather

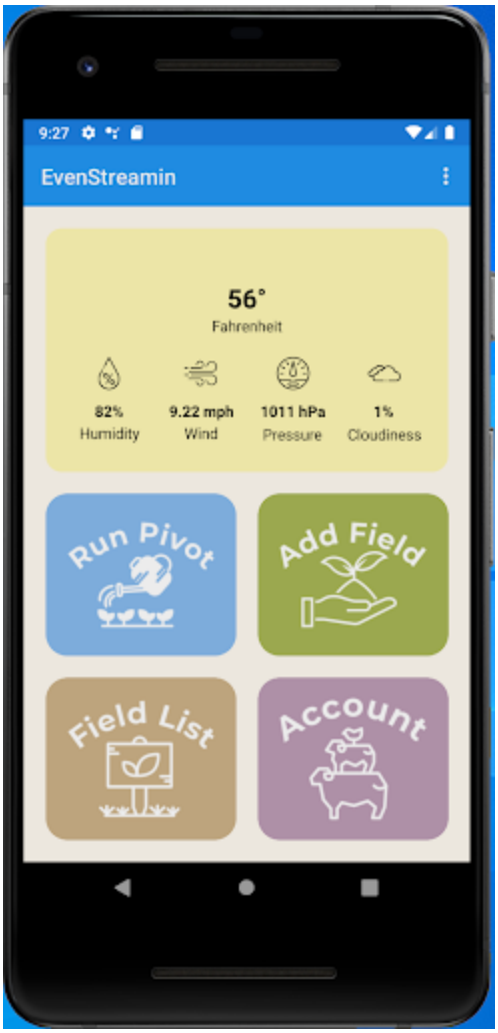


Figure 27: Android Sunny Weather

## 4.3 *Azure*

Another implementation component is the addition of a database. Refer to the entity-relationship (ER) diagram in Figure 28.

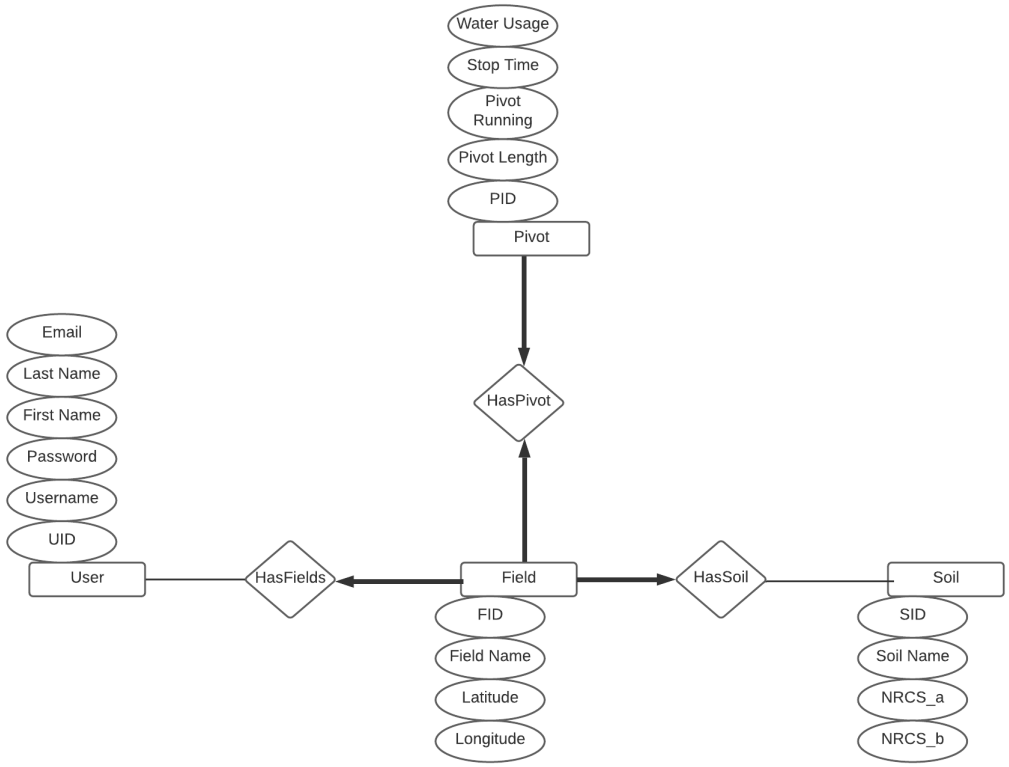


Figure 28: Entity-Relationship (ER) diagram for *EvenStreamin*

## 4.4 *ArcGIS Geolocation*

The end of fall semester was spent incorporating the ArcGIS SDK into the mobile application. This was done in order to use the 'LocationDataSource' class of ArcGIS. The main goal of this class is to track the location of the user's phone. This is necessary because the 'Add Field' functionality requires the exact latitude and longitude of the control panel. The ArcGIS 'LocationDataSource' class provides more exact results than user input would. Figures 29 and 30 display the process the user goes through to use the ArcGIS 'LocationDataSource' capabilities.

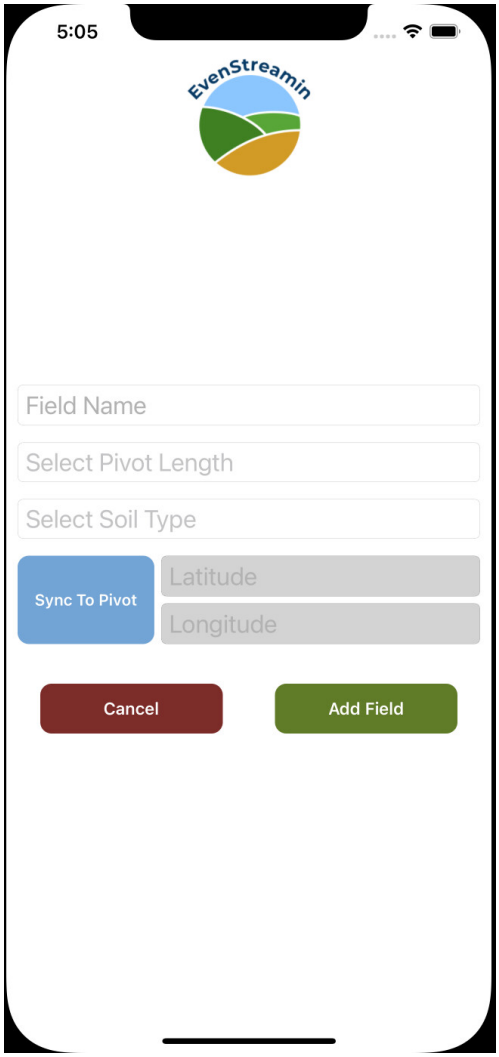


Figure 29: Add Field Page

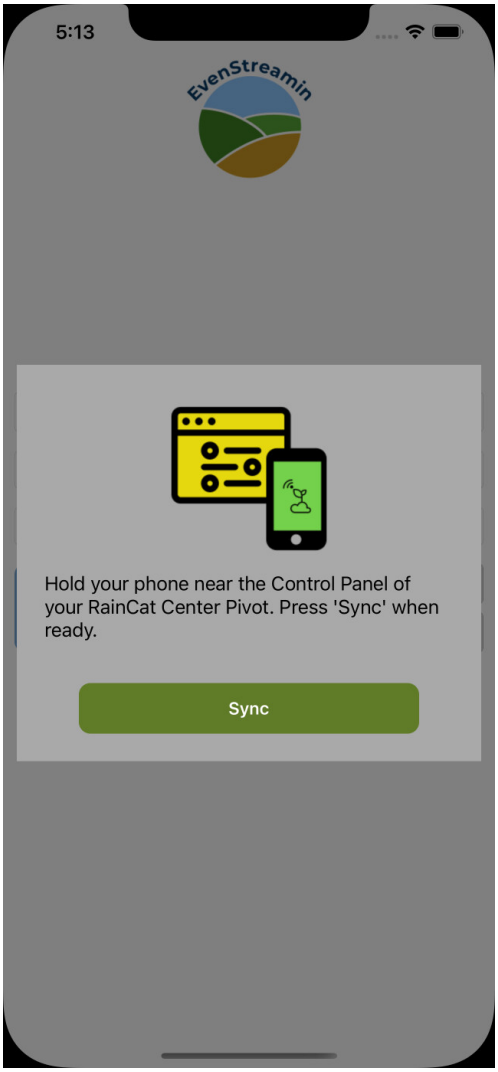


Figure 30: Sync To Pivot Popup

In order to get the location of an iOS or Android device, the app must first request permissions. Xamarin.Forms already includes a permissions API in order to request this information. If the user denies being able to track their location then they cannot add a field until they change that setting in their device.

## 4.5 *ArcGIS Visualization*

Since ArcGIS provides mapping capabilities, it seemed important to include that functionality within *EvenStreamin*. So during the spring semester I utilized the 'Map' ArcGIS

SDK class. Now the fields can be displayed with an ArcGIS 'MapView' page. Figures 31 and 32 display the process the user goes through to locate the ArcGIS map of their field.

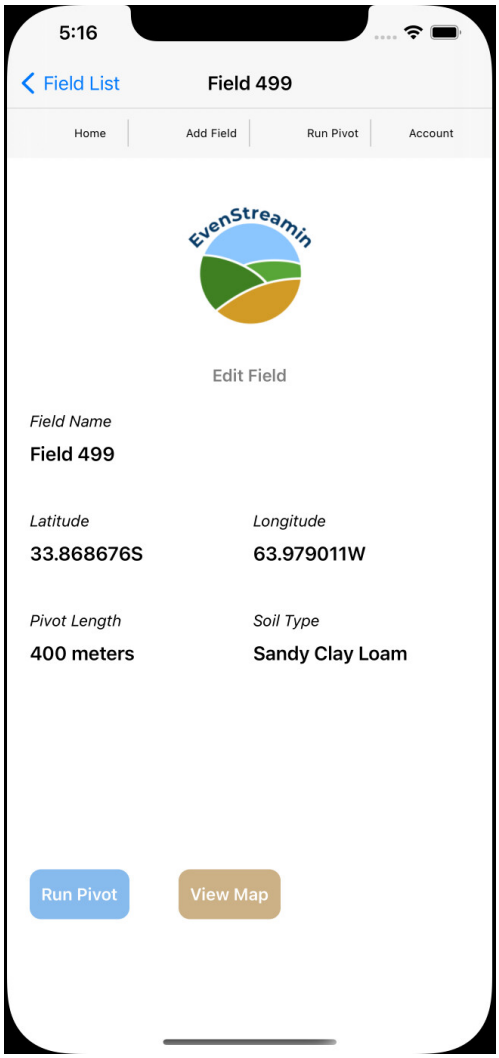


Figure 31: Stopped Field Page

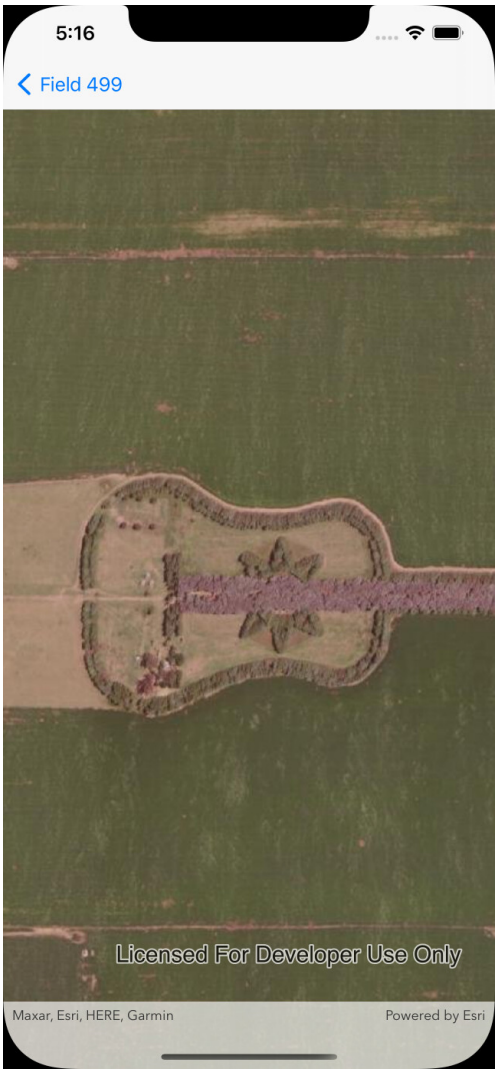


Figure 32: Map View

## 4.6 *RainCat*

Implementing the *RainCat* algorithm consisted of two parts:

- What is the end time of the running pivot?
- What is the total water usage of the pivot?

Based on assumptions made in the math capstone, it takes exactly 24 hours for a pivot to finish one rotation. So the implementation utilizes the DateTime functionality of Xamarin to add 24 hours to the current time.

Getting the total water usage was the main focus of the math capstone. As seen in the *RainCat* components section, the algorithm needs to find two intersections. However, neither equation is linear, so the intersections - if they happen at all - must be approximated. C# does not have any equation approximating libraries unlike other languages. So I had to iterate through both equations by time, checking for an intersection.

If the intersections exist then the time intervals for when the sprinkler is on, and when it is off are known. Figure 33 demonstrates the watering pattern that we have of the field so far.

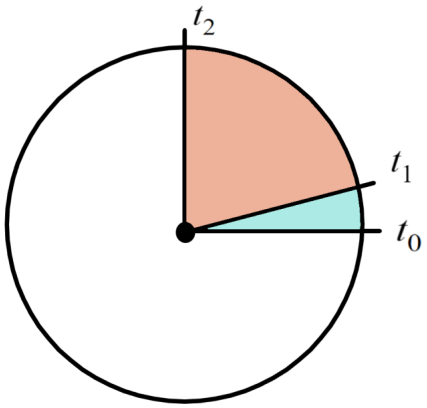


Figure 33: Sprinkler Watering Pattern for One Slice of the Field

Since it is known how long the pivot is on, the integral over that time interval is then how much water is used for that one section of the field. However, that is only one slice of the field. By using the circumference it is calculated how many of those slices exist in the full revolution. So the actual total water used by that one sprinkler is the integral times how often the slice is repeated. This is demonstrated in Figure 34.

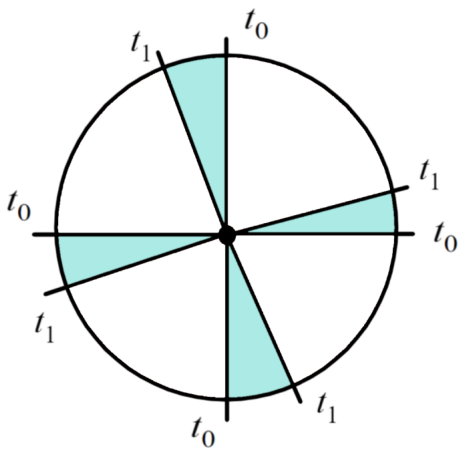


Figure 34: Sprinkler Watering Pattern for Entire Field

# 5 Future Work

There are three main projects that could extend my senior capstone:

1. Hardware Control Panel
2. Machine Learning Growth or Problem Detection
3. Generalized *RainCat* Model

The first project listed is to build the actual control panel of a center pivot. At the moment ArcGIS geolocation relies on the user too much. It is assumed that they will correctly hold their phone right next to the center pivot. But in software development it isn't safe to assume that the user will follow directions. Thus, building the actual control panel of the center pivot could result in more exact coordinates. Rather than using ArcGIS, the software would now interact with the control panel which would have a GPS within it.

The second project would be to utilize machine learning to monitor different aspects of a field. Right now *EvenStreamin* helps to remotely monitor water usage. However, water usage is not the only thing that is important when growing crops. Being able to monitor crop growth, or catch early stages of pests and diseases would be beneficial to farmers. Since the application is already connected to ArcGIS, future developers could easily utilize the maps and machine learning capabilities of ArcGIS. Valley Irrigation's mobile application was the inspiration for this future work, which can be seen in Figure 35

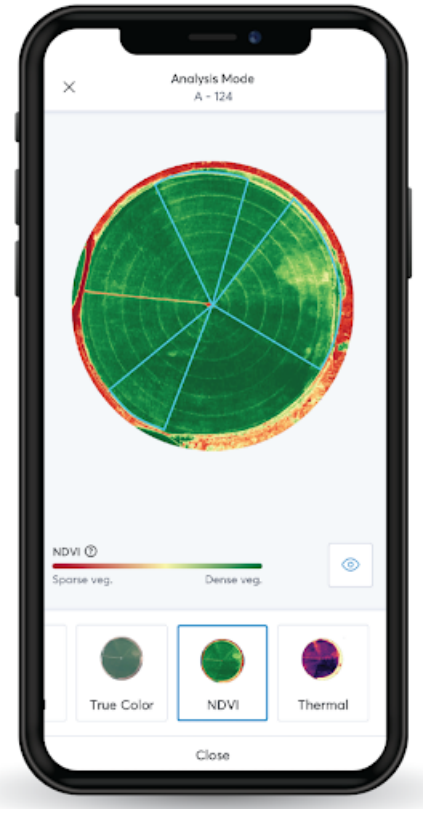


Figure 35: Valley Growth Analysis [10]

Finally is the *RainCat* model. Right now the model is too strict - the center pivot, sprinklers, and soil type can only have a handful of variations. To make the app work in more than just the perfect case, there needs to be a lot of work done on generalizing the model. This means adding different center pivot and sprinkler specs, and allowing for more specific soil types.

# 6 Summary

After nine months I have accomplished my original plan: the creation of a precision agriculture mobile application, and an algorithm, that reduce the water waste in center pivot irrigation. This has involved:

- Development with Xamarin and Xamarin.Forms.
- Incorporation of tools such as ArcGIS, Azure, and OpenWeatherMap.
- A joint project with the mathematics department.

In the end, I am proud of the mobile application that I was able to develop. Being passionate about the project helped me design and follow through on all aspects of the project. I did not realize how much I would enjoy learning new tools on my own. Although the learning curve for Xamarin, Azure, and ArcGIS were all steep, I feel as though I am a better computer science major and a better job applicant because of it. Working in the Agile environment has also been helpful for preparing what a real job will be structured like.

After finishing the entire senior capstone, I do have some advice for future students. First I think students should look at using tools and software outside of their comfort zone - especially if they are commonly seeing these tools on job applications. That was a big reason for the switch from SQLite to Azure. I wish I would have started the year with Azure to have a better understanding of it.

I highly encourage using tools that are not taught in any of the courses at Pacific. All seniors are aware of the limitations of being at a smaller school. It can be intimidating going up against students that attend schools with large computer science departments. This is a project that you can use to leverage the gap - and show that you don't need a class setting to learn new tools.

Along with the previous advice, if you are looking to break into a specific field - gaming, mobile, web development, etc - then try to focus your capstone in that area. I have received a handful of messages on LinkedIn from recruiters specifically looking for Xamarin

developers. So it might pay off to show companies that you truly are passionate about the industry or the tools.

My final piece of advice is that students should use LaTeX to write their papers. I don't enjoy LaTeX for creating presentations, but it was helpful for the final report. I have utilized the website Overleaf to complete mine - and realized that multiple people can edit a document at the same time! The only downside is that tools such as Google Docs or Microsoft Word have much better grammar and sentence structure checking.

# Glossary

| TERM | DEFINITION |
|---|---|
| ArcGIS | Geospatial software to view, edit, and analyze geographic data. The ArcGIS software is used to get the coordinates of each field. |
| ArcGIS Online | Web and cloud based platform that allows ArcGIS users to share their maps and data without having to install an ArcGIS desktop application. |
| Average Application Rate | The rate of water application over a wetted area [29]. |
| Azure | Cloud computing platform owned by Microsoft. Azure has over 200 services [22]. |
| Center Pivot Irrigation | Irrigates fields in a circular pattern around a central pivot point, creating a circular layout [5]. |
| Central Pivot Point | Stationary structure connected to a water supply, that the main water pipe rotates around. |
| Cloud Computing | The delivery of computing services, where a network of remote servers are used to analyze, manage, and route data [20]. |
| Control Panel | A piece of hardware attached to the pivot point that gives commands to the center pivot machine. They control starting, stopping, changing directions, running wet versus dry, and much more. |
| Cross-platform mobile development | Creation of software that is compatible with multiple mobile operating systems. |
| Database | "Organized collection of structured information" which is controlled by a database management system (DBMS) [18]. |

| | |
|---|---|
| Design patterns | Structure applications to make them more maintainable and reusable [16]. |
| Drive Units | Part of the center pivot that touches the ground, and have specific hardware that allows for movement. It consists of a basebeam, drive train, wheels, and various structural supports [30]. |
| Drop Tube Sprinkler | Sprinklers that hang below the main water pipe. |
| EvenStreamin | A cross-platform mobile application for farmers to make accounts and manage the watering of their fields remotely. |
| Integrated Development Environment (IDE) | Software application that provides an environment for software development. Has a source code editor, build tools, and a debugger. |
| Infiltration Rate | The rate at which water can enter the soil. The infiltration rate is not the same for all parts of the field nor is it even the same at one point all of the time [7]. |
| Instantaneous Application Rate | The peak intensity of water application at a specific point. Due to the circular characteristics of a center pivot, the instantaneous application rate increases the further from the central pivot point the sprinkler is [6]. |
| LRDU | Last Regular Drive Unit is the last drive unit on a pivot. |
| Model-View-ViewModel (MVVM) | Design pattern that separates the business logic from the graphical user interface (GUI) logic. |
| Precision Agriculture | Technology-based management system that collects and organizes data to optimize profits, sustainability, and protection of the environment [8]. |
| RainCat | The mathematical model of the center pivot sprinkler system. |

| | |
|---|---|
| Relational Database | Database that is organized as sets of tables with rows and columns [18]. |
| Software development kits (SDK) | An SDK contains a range of tools such as libraries, application programming interfaces (API), programming tools, and documentation. |
| Span | Consist of the main water pipeline, sprinklers, and a supporting structure of trussing that holds the weight between towers [30]. Spans can come in different sizes so that a center pivot can fit any field size. |
| SQLite | Library that provides a relational database management system. |
| Workloads | Used to modify Visual Studio and contain tools needed for the specific programming languages or platforms being used [13]. |
| Xamarin | Open-source framework for cross-platform mobile development [13]. Xamarin is built on top of a .NET framework - which handles memory allocation and garbage collection [31]. |
| Xamarin.Forms | Open-source User Interface (UI) framework on top of Xamarin. |

Table 1: Table of Terms and Definitions

# References


[1] National Geographic Society, "The development of agriculture," Aug 2019. [Online] Available: https://www.nationalgeographic.org/article/development-agriculture/#:~:text=Agricultural%20communities%20developed%20approximately%2010%2C000,foraging%20and%20hunting%20for%20survival.

[2] R.E. Sojka, D.L. Bjorneberg, and J.A. Entry, "Irrigation: An historical perspective," Accessed March 2, 2021. [Online] Available: https://eprints.nwisrl.ars.usda.gov/id/eprint/815/1/1070.pdf.

[3] National Geographic Society, "Environmental impacts of agricultural modifications," May 2020. [Online] Available: https://www.nationalgeographic.org/article/environmental-impacts-agricultural-modifications/.

[4] USDA, "Irrigation and water use." Accessed Nov 9, 2020. [Online] Available: https://www.ers.usda.gov/topics/farm-practices-management/irrigation-water-use/#:~:text=USDA's%20Farm%20and%20Ranch%20Irrigation,is%20equivalent%20to%20325%2C851%20gallons.

[5] P. Waller and M. Yitayew, *Irrigation and Drainage Engineering*. Springer-Cham, 1st ed., 2016. [Online] doi: https://doi.org/10.1007/978-3-319-05699-9.

[6] USDA, "National engineering handbook," 2016. [Online] Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=39754.wba.

[7] Traxo, "Maximum length and area of a center pivot." Accessed Nov 17, 2020. [Online] Available: http://www.traxco.com/maximum-length-area-of-a-center-pivot.

[8] T. Hammerich, "What is 'precision agriculture'?," September 2020. [Online] Available: https://aggrad.com/what-is-precision-agriculture/.

[9] Valley Irrigation, "What is a center pivot? pivot 101 - valley irrigation," November 2018. [Online] Available: https://www.youtube.com/watch?v=2bILpvH3EuQ.

[10] Valley Irrigation, "Valley 365®." Accessed Sep 19, 2020. [Online] Available: http://www.valleyirrigation.com/valley-365.

[11] SmartIrrigation, "Smartirrigation about." Accessed Nov 9, 2020. [Online] Available: https://smartirrigationapps.org/about/.

[12] J. Mendes, T. M. Pinho, F. N. Santos, J. J. Sousa, E. Peres, J. Boaventura-Cunha, M. Cunha, R. Morais, "Smartphone applications targeting precision agriculture practices—a systematic review," *Agronomy*, vol. 10, May 2020. [Online] doi: 10.3390/agronomy10060855.

[13] Visual Studio Docs, "Installing xamarin in visual studio 2019." Accessed Nov 17, 2020. [Online] Available: https://docs.microsoft.com/en-us/xamarin/get-started/installation/windows.

[14] GeeksforGeeks, "Introduction to c," December 2019. [Online] Available: https://www.geeksforgeeks.org/introduction-to-c-sharp/.

[15] TutorialsPoint, "Xaml - overview." Accessed Dec 1, 2020. [Online] Available: https://www.tutorialspoint.com/xaml/xaml_overview.htm.

[16] B. D. McLaughlin, G. Pollice, D. West, *Head First Object-Oriented Analysis and Design*. November 2006.

[17] Visual Studio Docs, "The model-view-viewmodel pattern." Accessed Nov 24, 2020. [Online] Available: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

[18] Oracle, "What is a database?." Accessed Nov 24, 2020. [Online] Available: https://www.oracle.com/database/what-is-database/.

[19] SQLite Tutorial, "What is sqlite." Accessed Nov 24, 2020. [Online] Available: https://www.sqlitetutorial.net/what-is-sqlite/.

[20] R. Borah, "Cloud computing architecture: What is front end and back end?." Accessed April 10, 2021. [Online] Available: https://www.clariontech.com/blog/cloud-computing-architecture-what-is-front-end-and-back-end.

[21] Microsoft, "What is cloud computing?." Accessed April 10, 2021. [Online] Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/.

[22] Microsoft, "What is azure?." Accessed April 10, 2021. [Online] Available: https://azure.microsoft.com/en-us/overview/what-is-azure/.

[23] Microsoft, "Azure functions." Accessed April 10, 2021. [Online] Available: https://azure.microsoft.com/en-us/services/functions/.

[24] S. Hanselman, "What is serverless computing? exploring azure functions," Aug 2016. [Online] Available: https://www.hanselman.com/blog/what-is-serverless-computing-exploring-azure-functions.

[25] S. Mohammad, "Azure storage - tables," Feb 2019. [Online] Available: https://www.c-sharpcorner.com/article/azure-storage-tables/.

[26] GIS Geography, "What is arcgis?," November 2020. [Online] Available: https://gisgeography.com/what-is-arcgis/.

[27] L. Valdellon, "What is an sdk? everything you need to know," July 2020. [Online] Available: https://clevertap.com/blog/what-is-an-sdk/.

[28] Crop Metrics, "Vri zone control animation," Accessed March 3, 2021. [Online] Available: https://www.youtube.com/watch?v=UrfqeHKa9kA.

[29] Nelson Irrigation, "We'd never criticize mother nature, but sometimes "rain-like" irrigation is not the best for soil integrity." Accessed Nov 17, 2020. [Online] Available: http://www.nelsonirrigation.com/products/family/pivot-sprinklers/precision-irrigation/.

[30] Irrigation Education, "Pivotbasics," Accessed March 3, 2021. [Online] Available: https://blog.irrigation.education/hubfs/Valley_Files/Valley%20-%20eBooks/Pivot_Basics_-_What_You_Need_To_Know_About_Center_Pivot_Irrigation.pdf.

[31] Microsoft, "What is xamarin?." Accessed Nov 17, 2020. [Online] Available: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin.

[32] D. Rogers, J. Aguilar, I. Kisekka, F. R. Lamm, "Center pivot irrigation system losses and efficiency," [Online] Available: https://www.ksre.k-state.edu/irrigate/reports/r17/Rogers17.pdf.

[33] B. Leib, T. Grant, "Understanding center pivot application rate," October 2019. [Online] Available: https://extension.tennessee.edu/publications/Documents/W809-F.pdf.

[34] AltexSoft, "The good and the bad of xamarin mobile development," November 2020. [Online] Available: https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/.

[35] Wikipedia, "Model–view–viewmodel," November 2020. Accessed Nov 24, 2020. [Online] Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel.

[36] SQLite, "Appropriate uses for sqlite." Accessed Nov 24, 2020. [Online] Available: https://www.sqlite.org/whentouse.html.

[37] eGIS Associates, "What is arcgis?," April 2020. [Online] Available: https://www.youtube.com/watch?v=s6cotWPLfN4.

[38] ArcGIS, "Landsat 8 (agriculture)." Accessed Nov 24, 2020. [Online] Available: https://www.arcgis.com/home/item.html?id=4f9fb4c44401443b9b416ae7a2917032.

[39] Howell, T.A. , "Sprinkler package water loss comparisons," Accessed March 3, 2021. [Online] Available: https://www.ksre.k-state.edu/irrigate/oow/p04/Howell.pdf.

[40] Seminis, "Innovations in center pivot irrigation." Accessed March 3, 2021. [Online] Available: https://www.seminis-us.com/innovations-center-pivot-irrigation/.

[41] T. Peters, H. Neibling, R. Stroh, B. Molaei, H. Mehanna, "Low energy precision application (lepa) and low elevation spray application (lesa) trials in the pacific northwest," Accessed March 3, 2021. [Online] Available: https://extension.oregonstate.edu/sites/default/files/documents/33601/lepa-lesa-pnw-stroh-revisions.pdf.

[42] Application of Kostiakov's Infiltration Model on the Soils of Umudike, Abia State - Nigeria, "R. adindu, k. igbokwe, t. chigbu, c.a. ike-amadi," 2014. [Online] doi: 10.5923/j.ajee.20140401.01.

[43] Simplilearn, "What is azure and how does it work?," Feb 2021. [Online] Available: https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure.

[44] Microsoft, "Azure products." Accessed April 10, 2021. [Online] Available: https://azure.microsoft.com/en-us/services/.

[45] Microsoft, "Table storage." Accessed April 10, 2021. [Online] Available: https://azure.microsoft.com/en-us/services/storage/tables/.

# 7 Appendices

## 7.1 *Appendix 1 - Android and iOS User Interface*

The iOS interface is on an iPhone 12 Pro Max iOS 14.4 simulator, while the Android interface is on a Pixel 2 Pie 9.0 API 28 simulator.

Account Deletion Alert

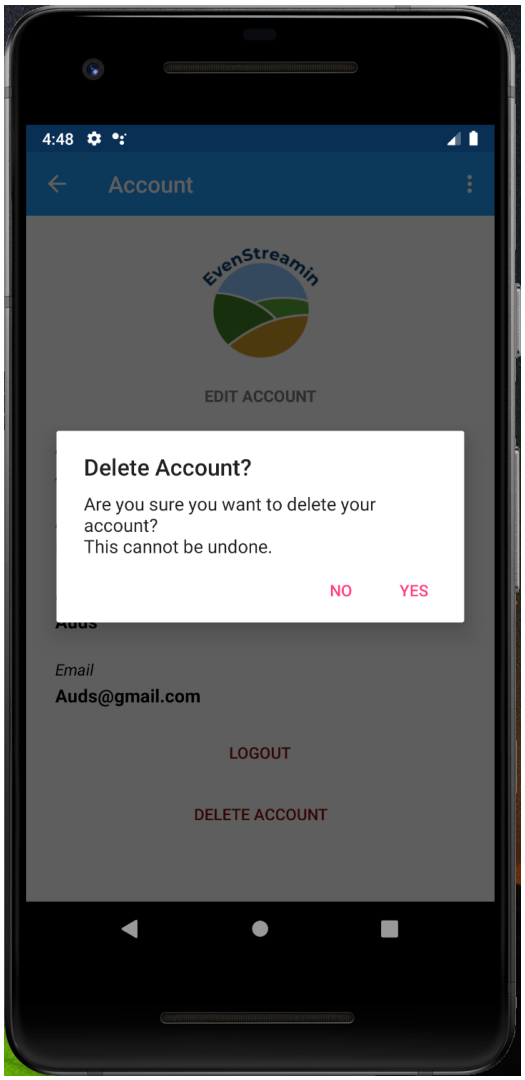


Figure 36: Android

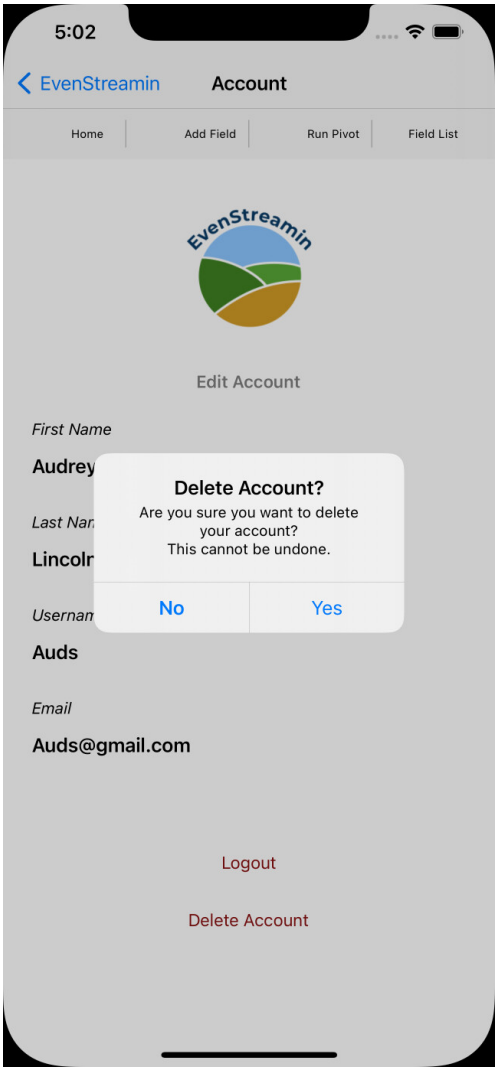


Figure 37: iOS

Account Edit Popup

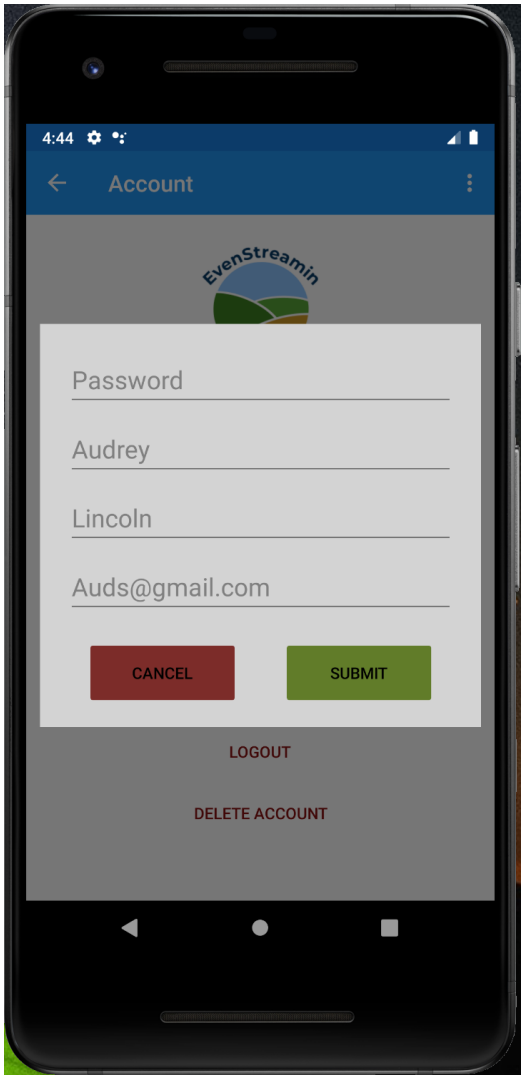


Figure 38: Android

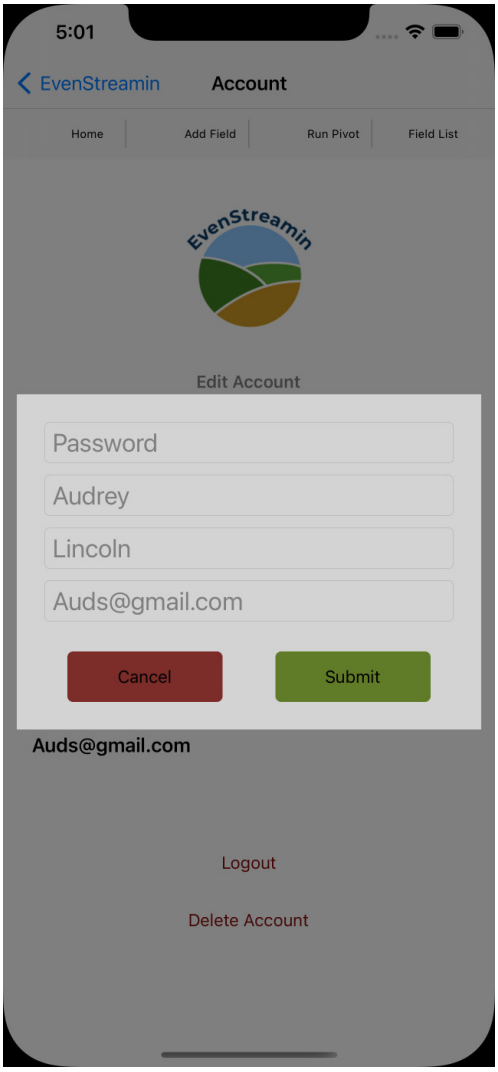


Figure 39: iOS

Account Edit Alert

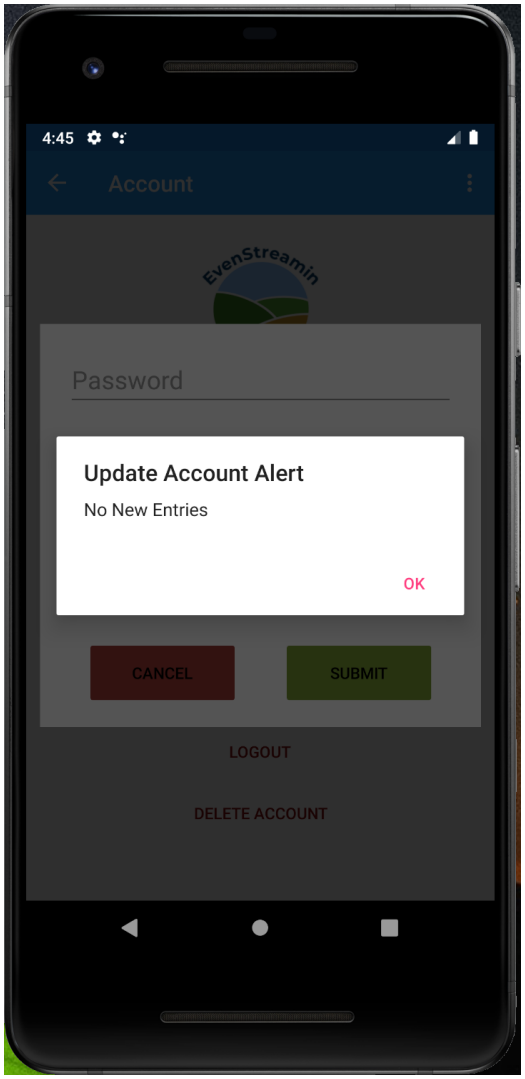


Figure 40: Android

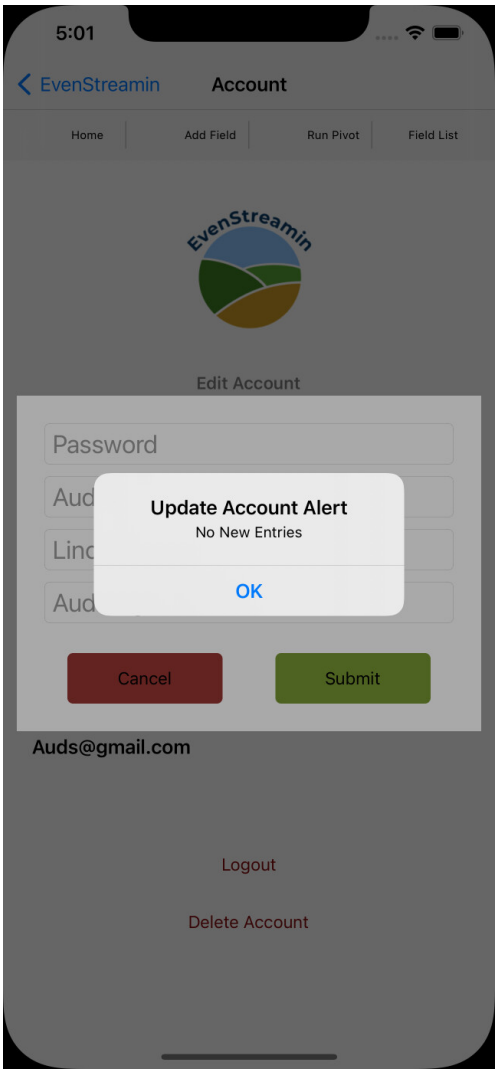


Figure 41: iOS

Account Logout Alert

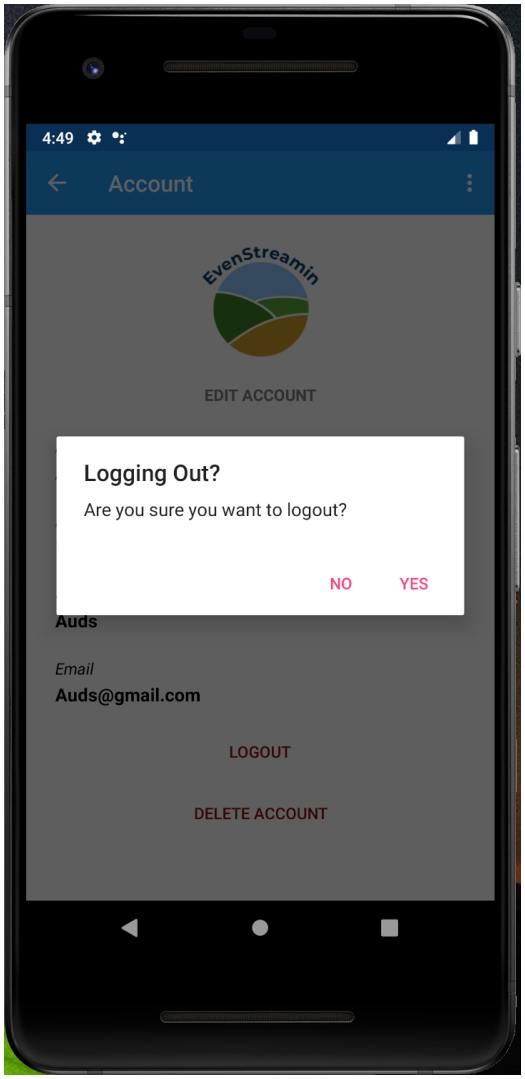


Figure 42: Android

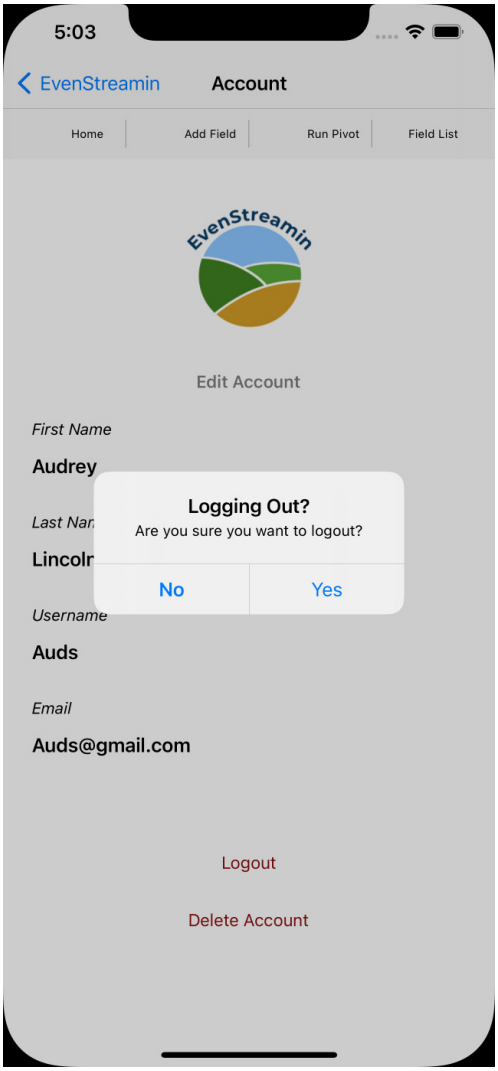


Figure 43: iOS

Account

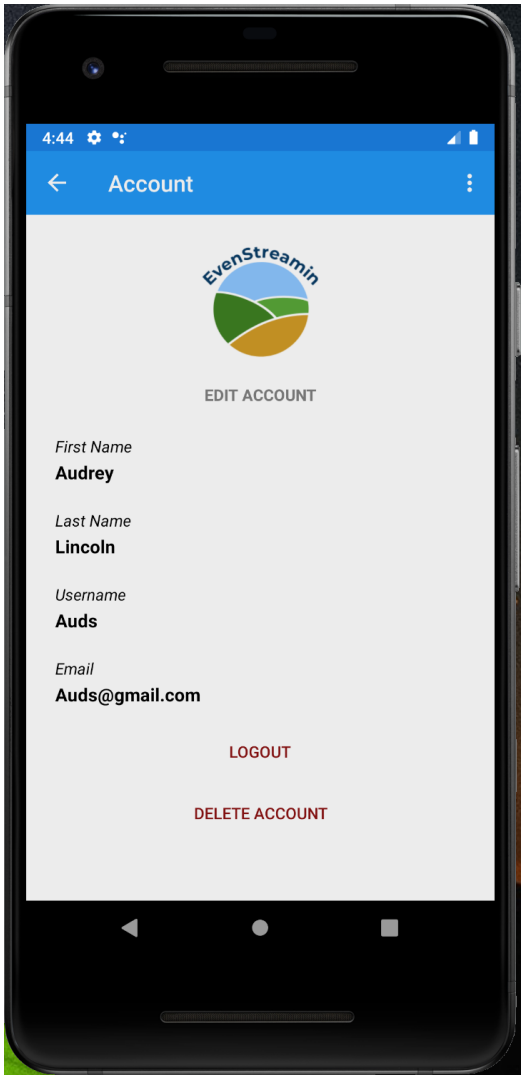


Figure 44: Android

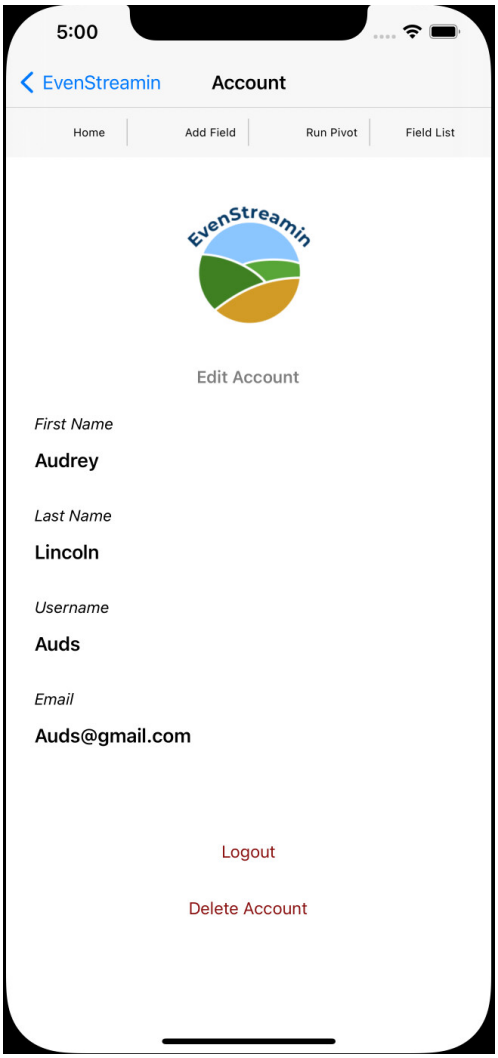


Figure 45: iOS

Add Field Fill Out Alert

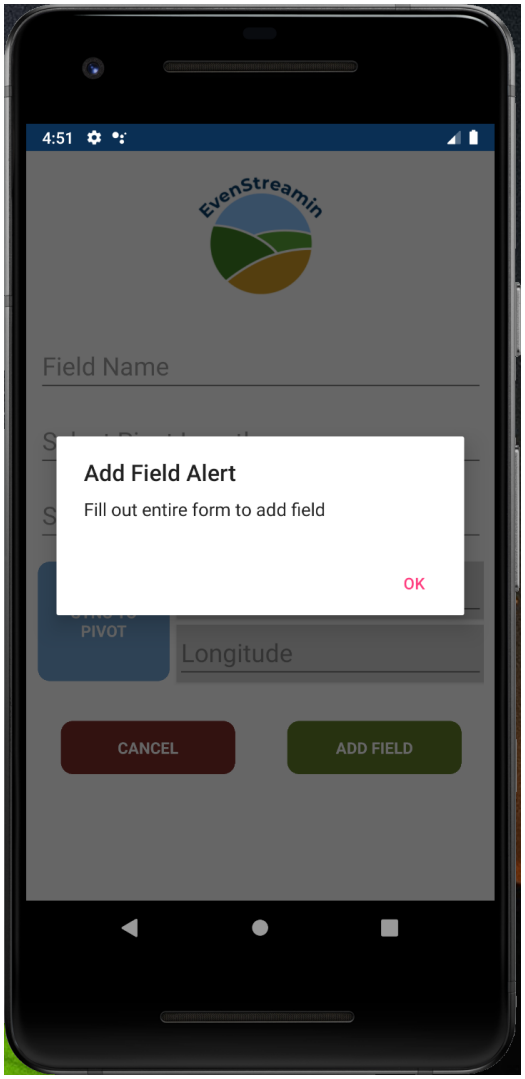


Figure 46: Android

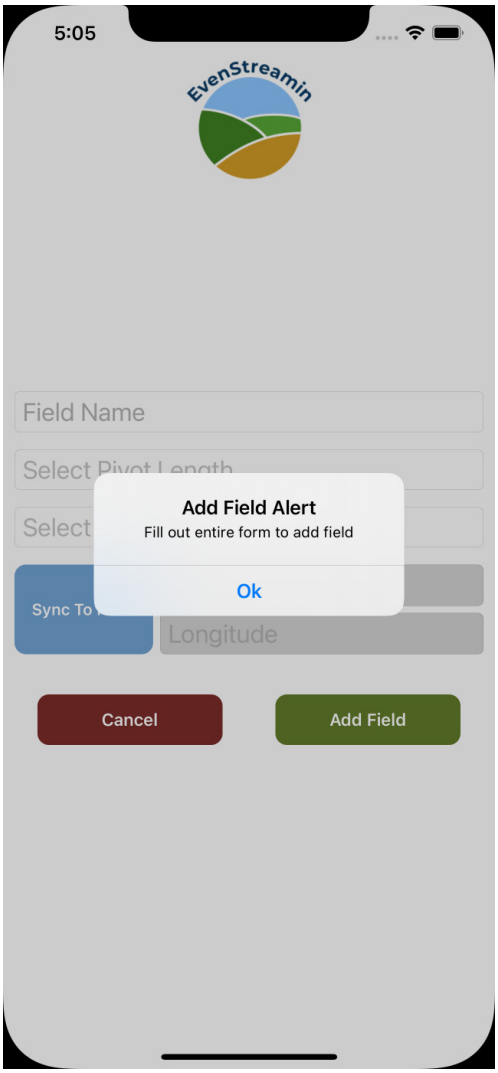


Figure 47: iOS

Add Field Location Alert

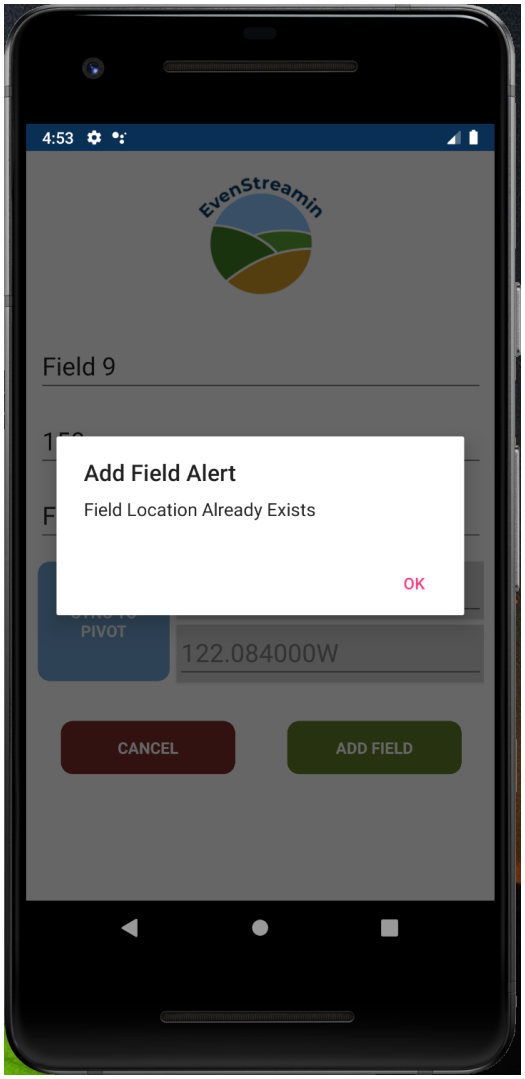


Figure 48: Android

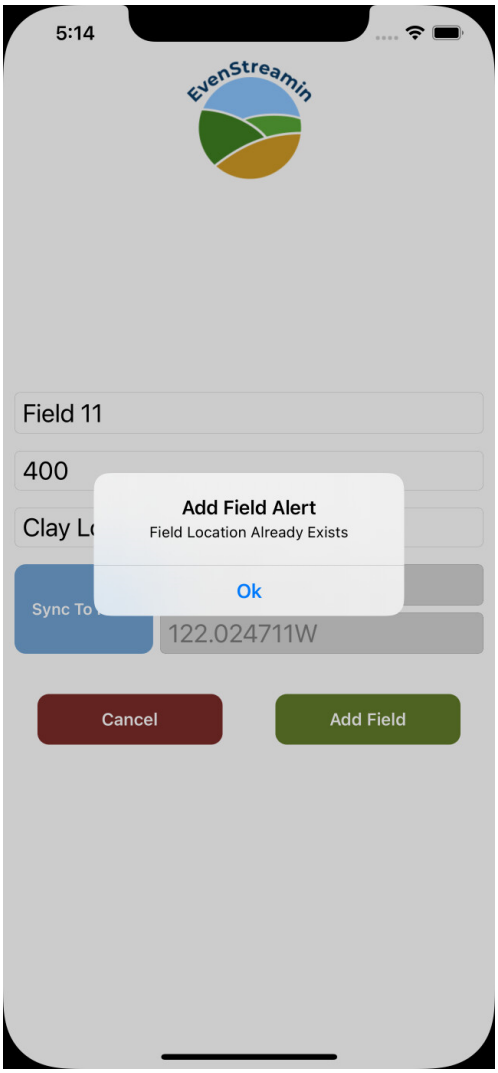


Figure 49: iOS

Add Field Pivot Selection

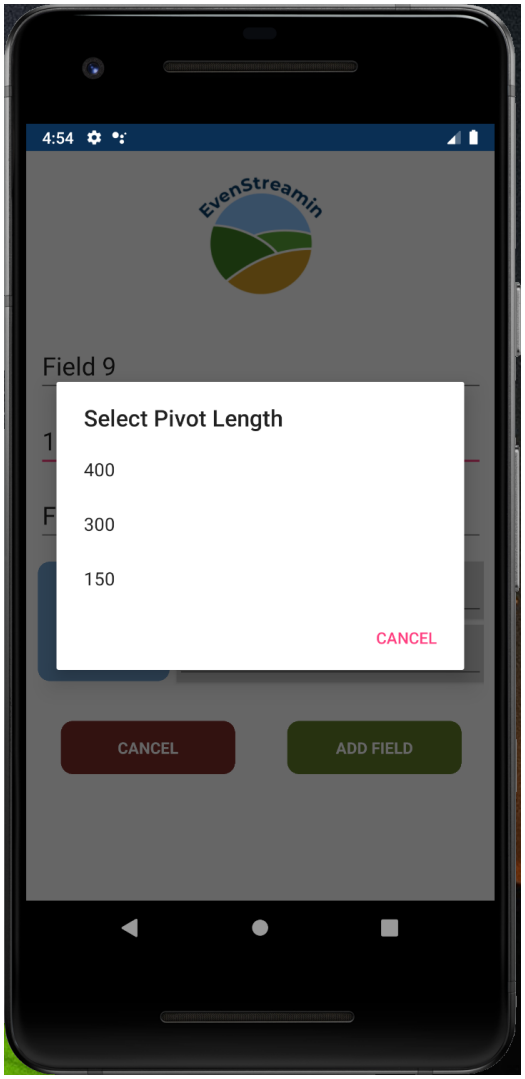


Figure 50: Android

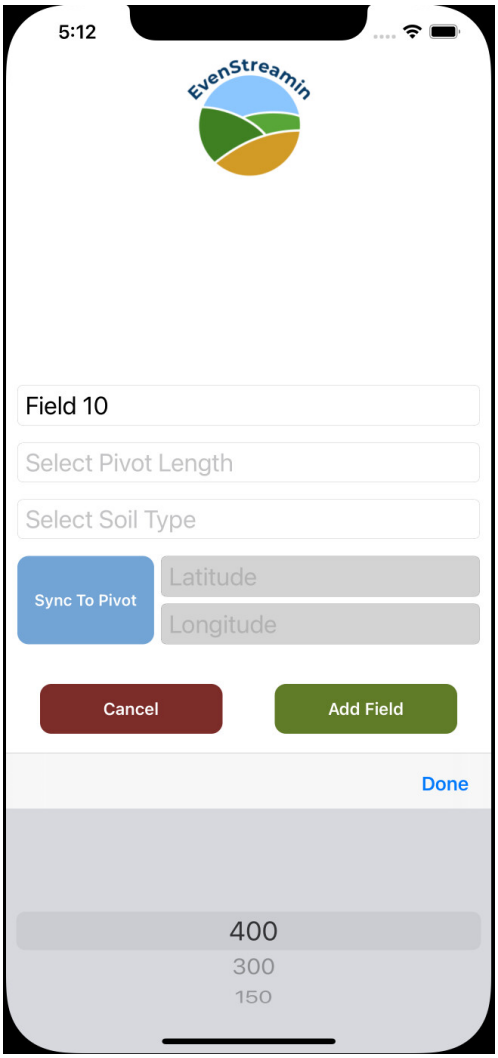


Figure 51: iOS

Add Field Soil Selection

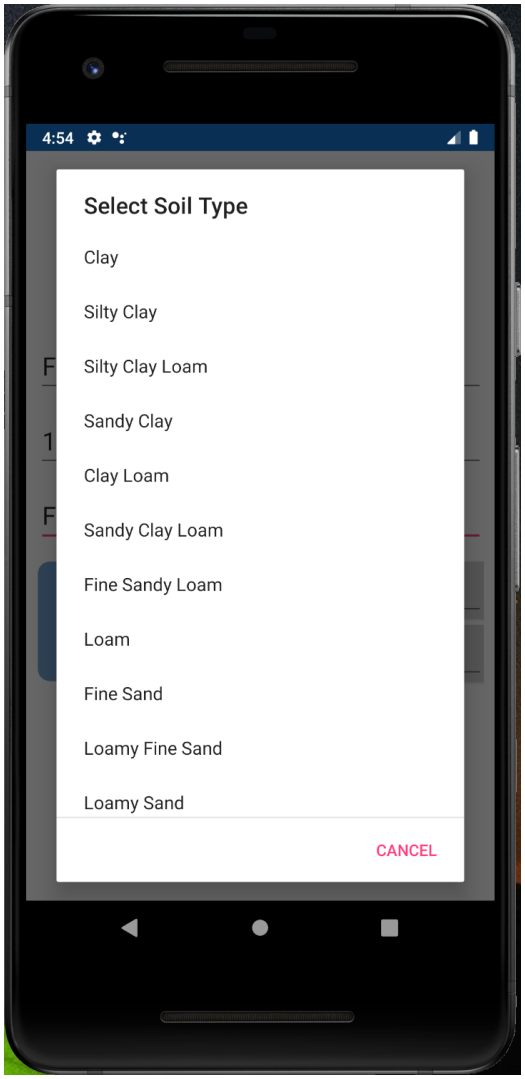


Figure 52: Android

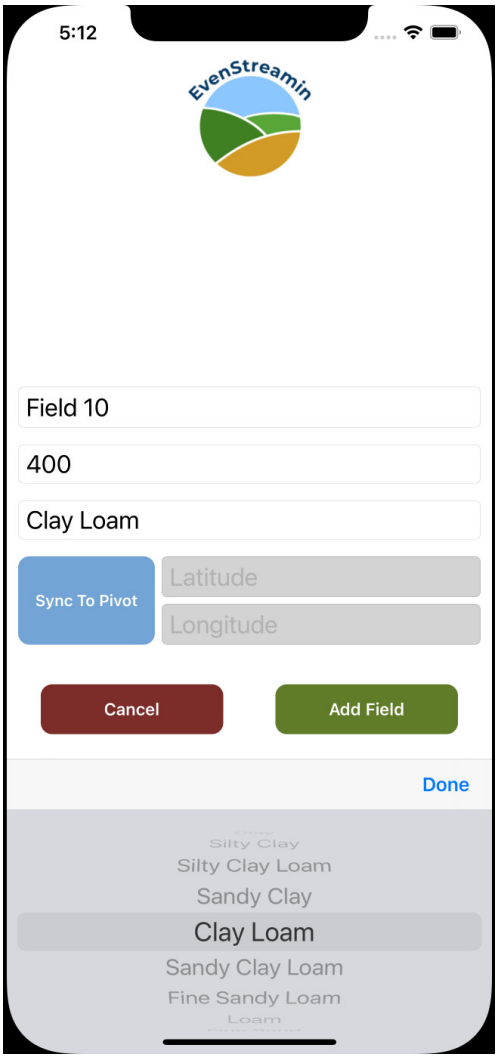


Figure 53: iOS

Add Field Sync To Pivot

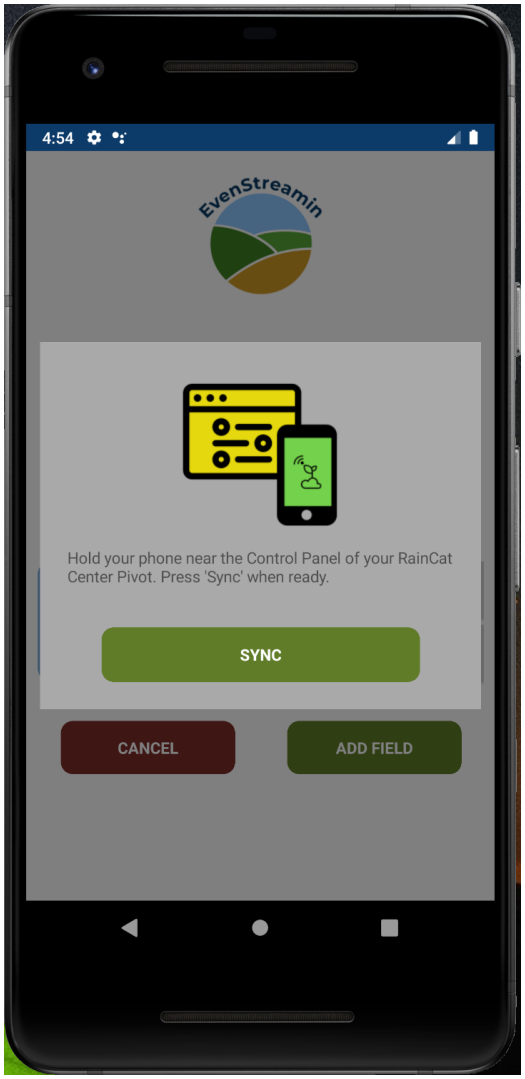


Figure 54: Android

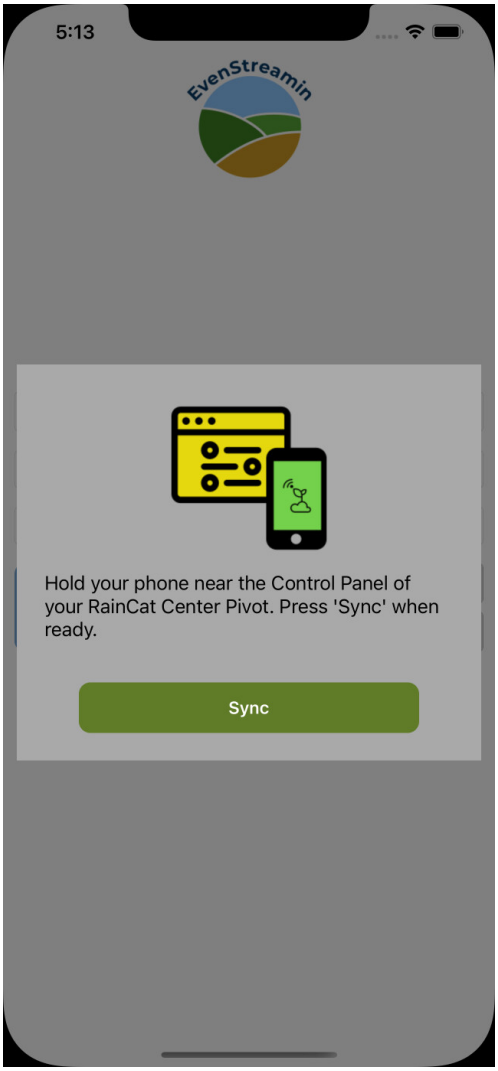


Figure 55: iOS

Add Field

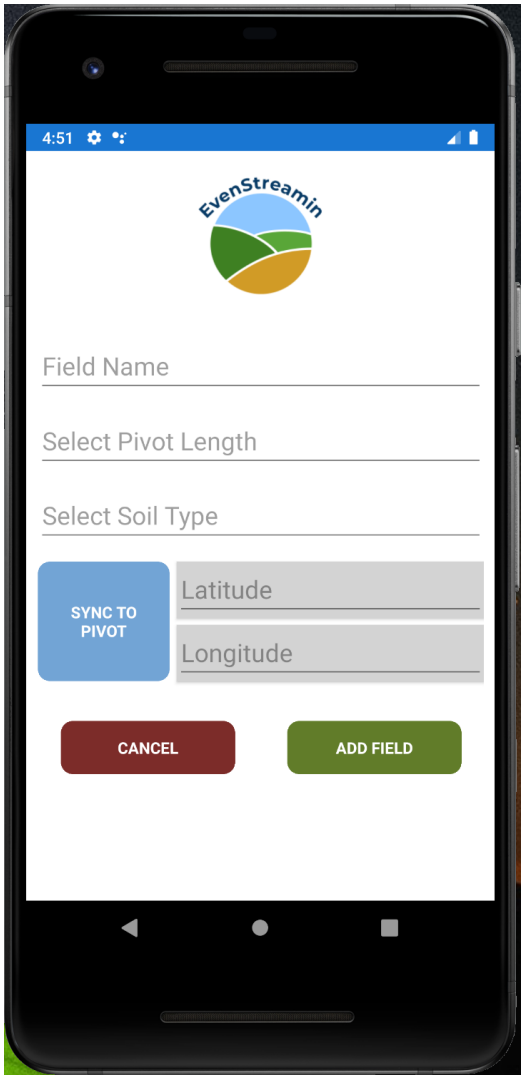


Figure 56: Android

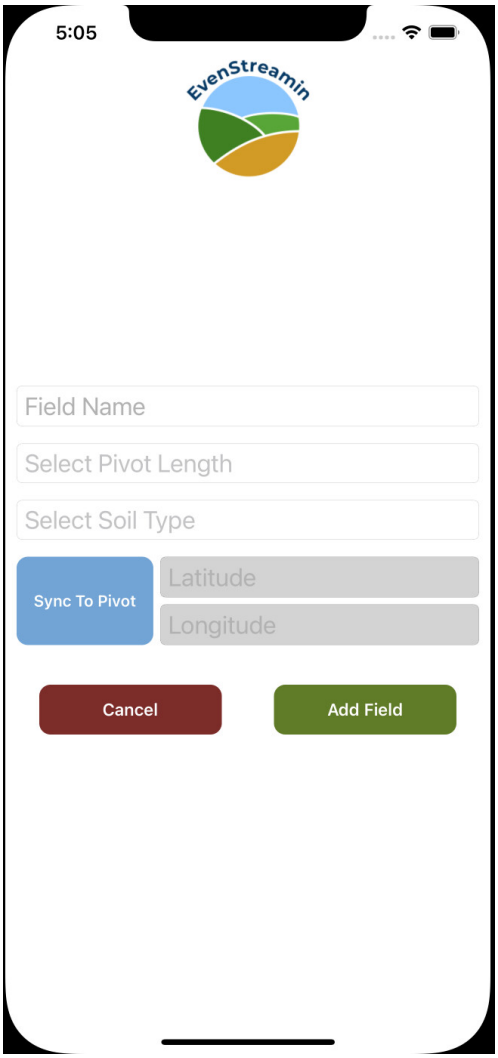


Figure 57: iOS

Field List Delete

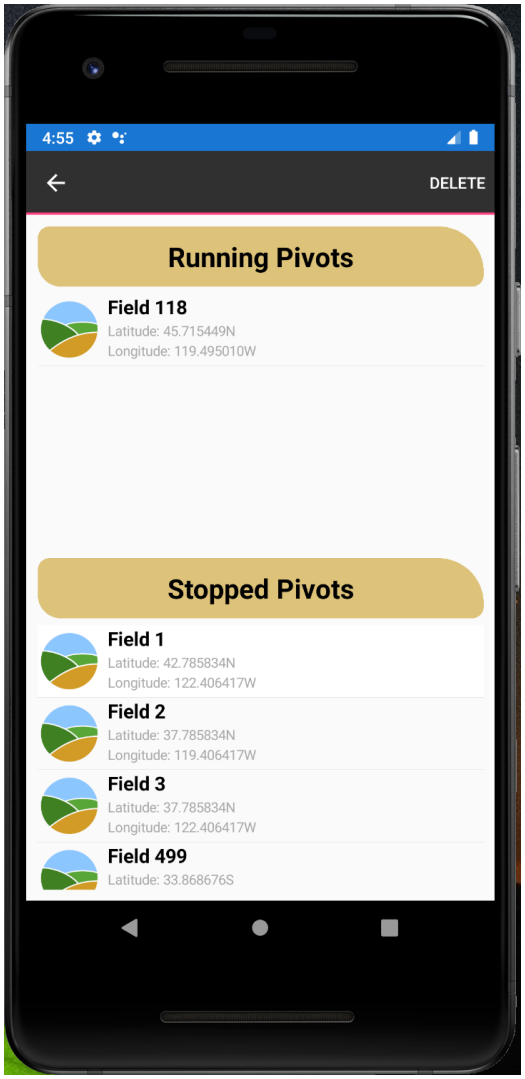


Figure 58: Android

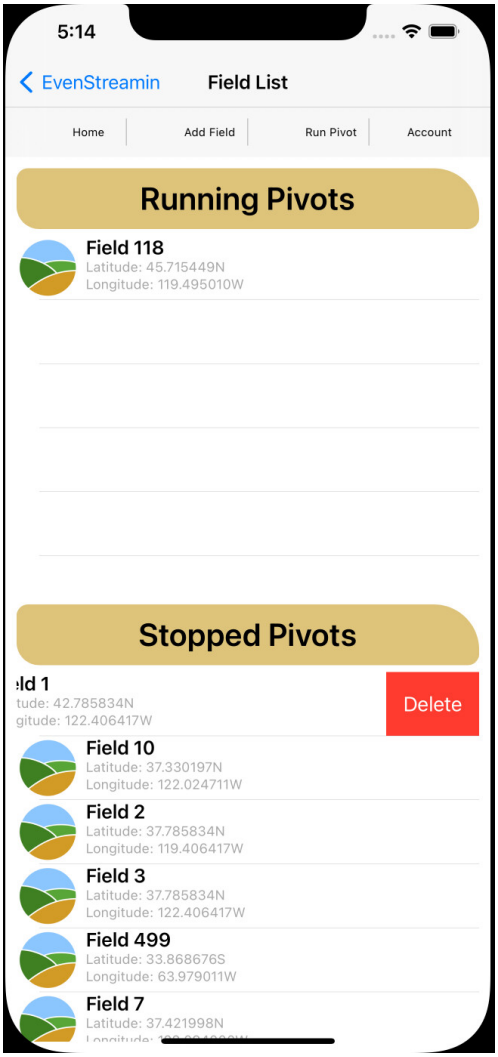


Figure 59: iOS

Field List Delete Alert

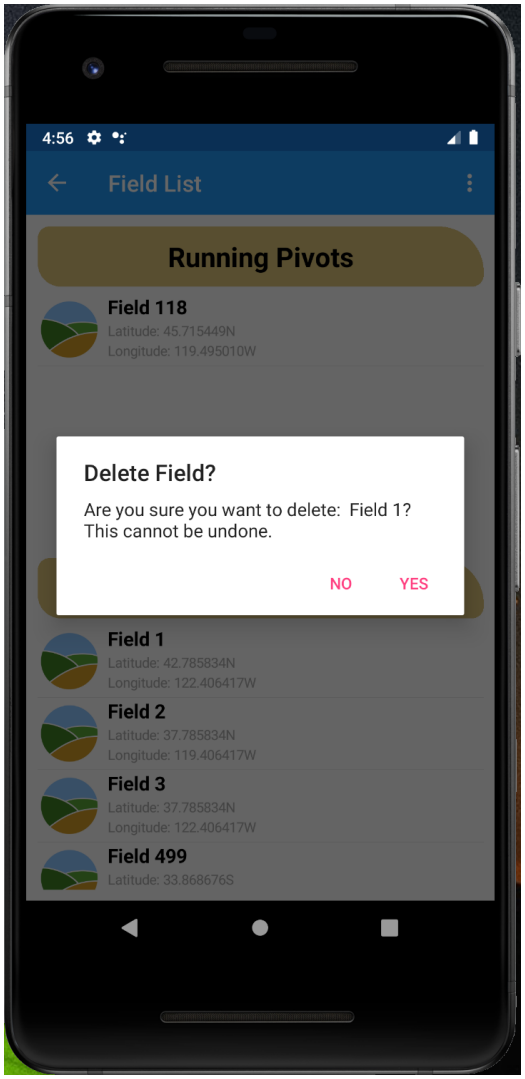


Figure 60: Android

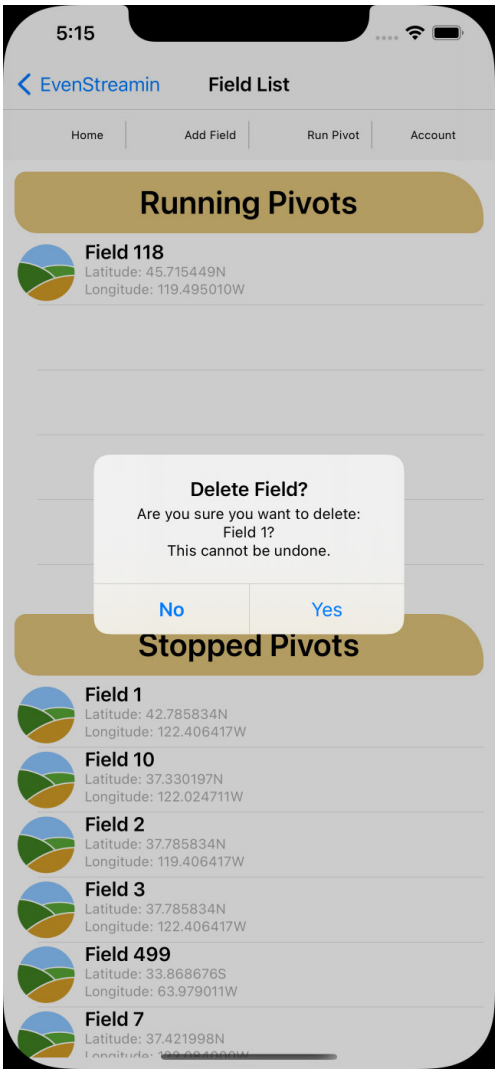


Figure 61: iOS

Field List

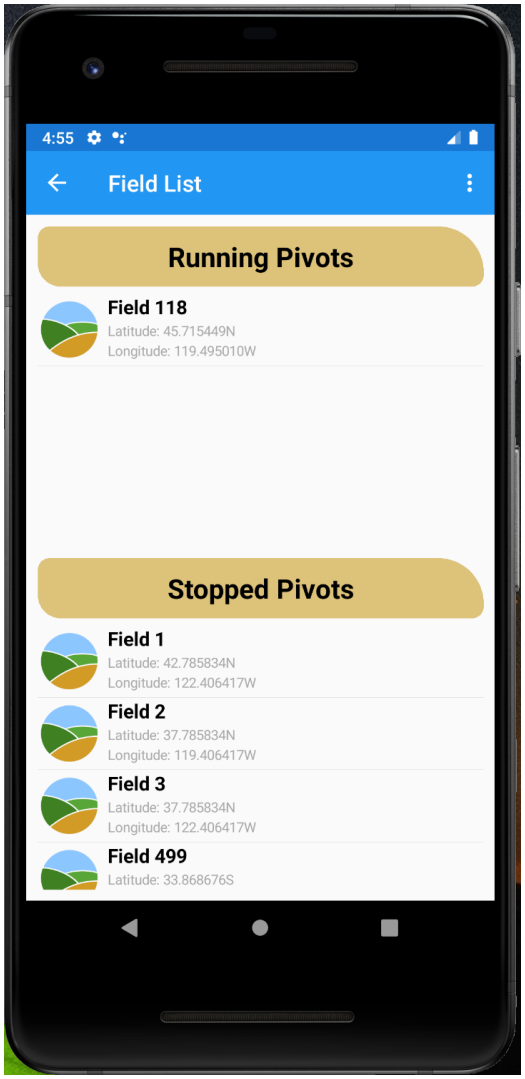


Figure 62: Android

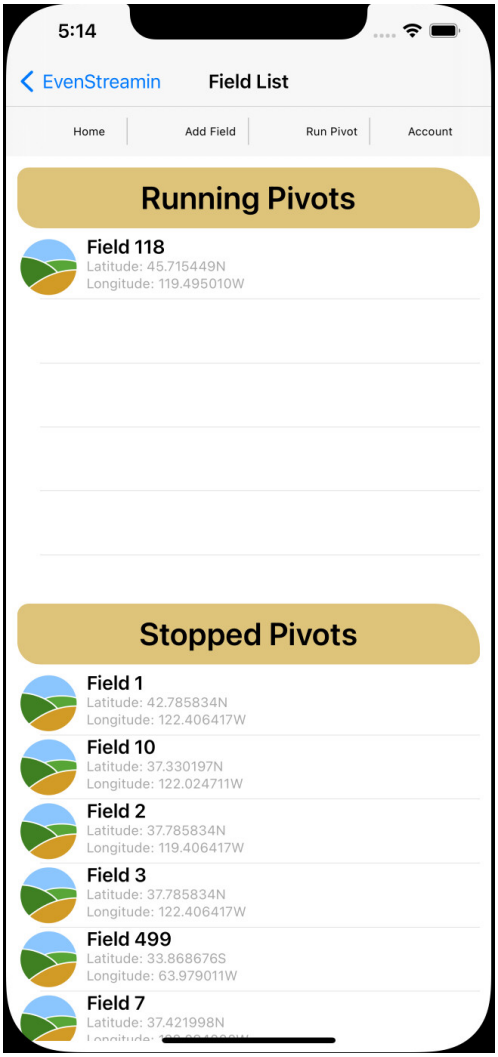


Figure 63: iOS

Home

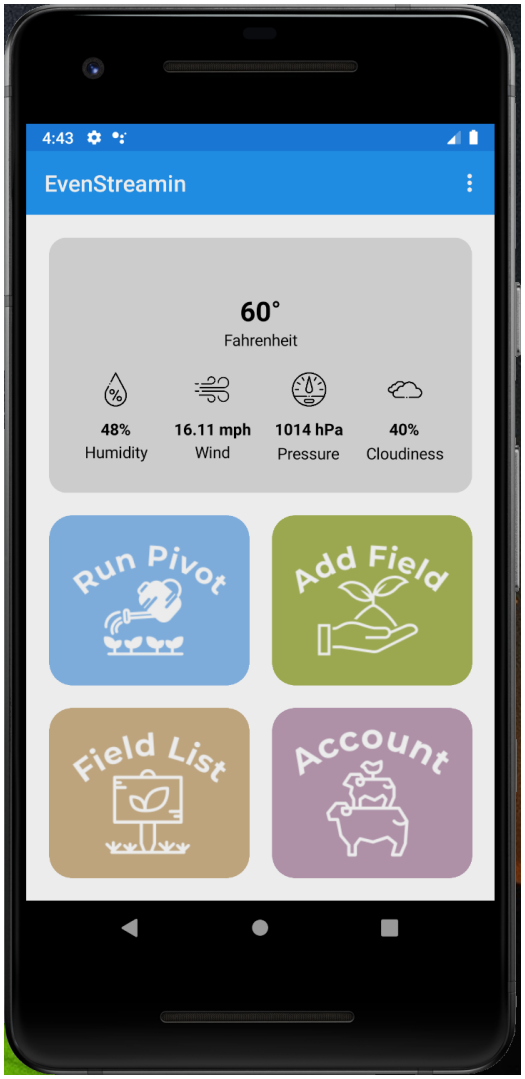


Figure 64: Android

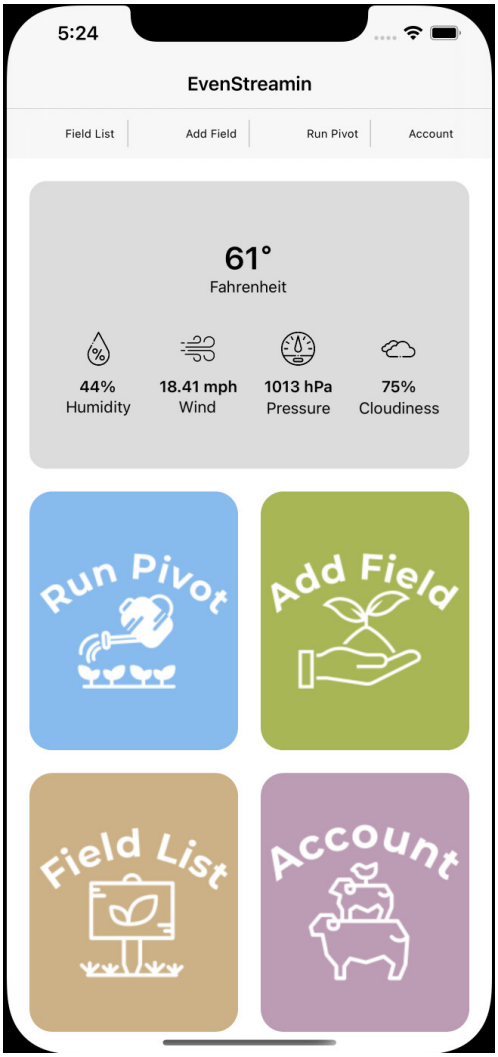


Figure 65: iOS

Login Fill Out Alert

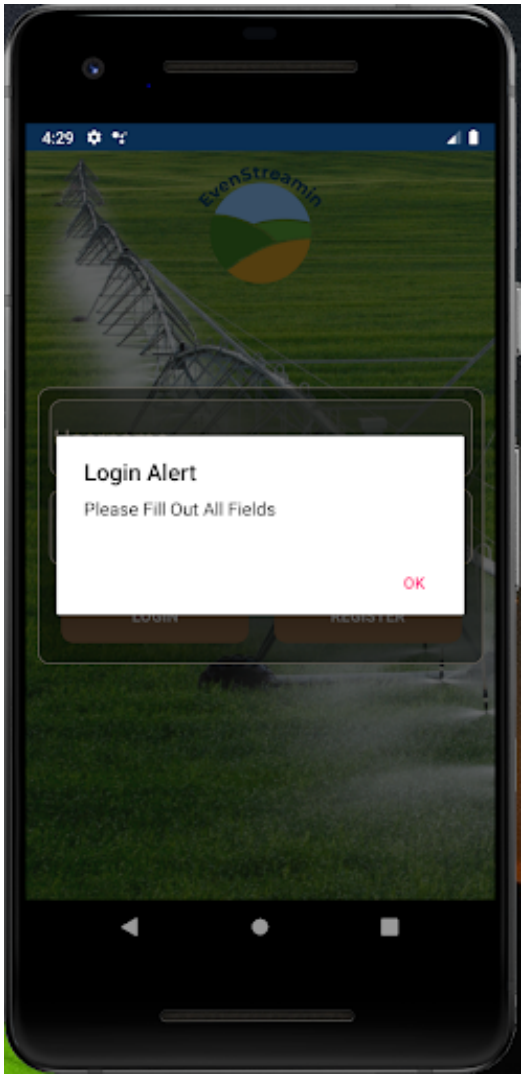


Figure 66: Android

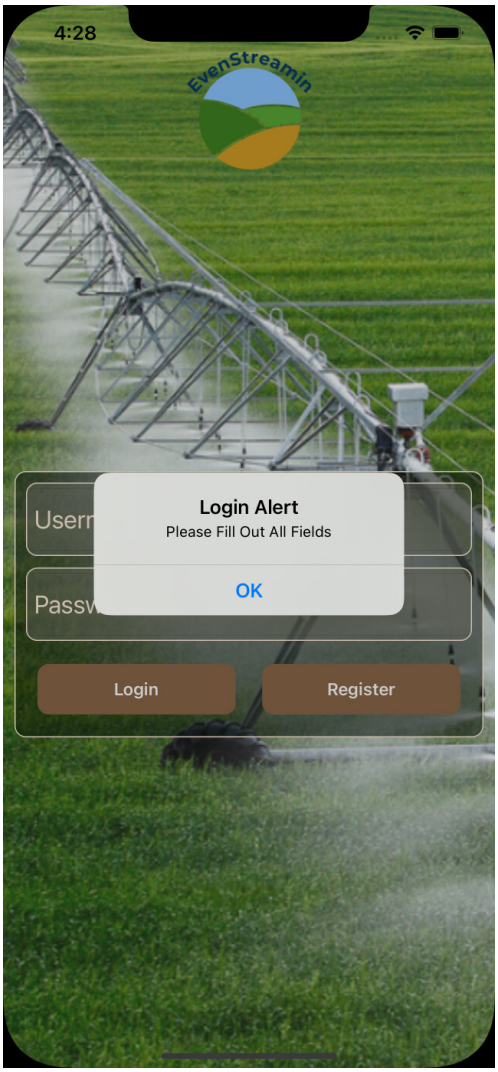


Figure 67: iOS

Login

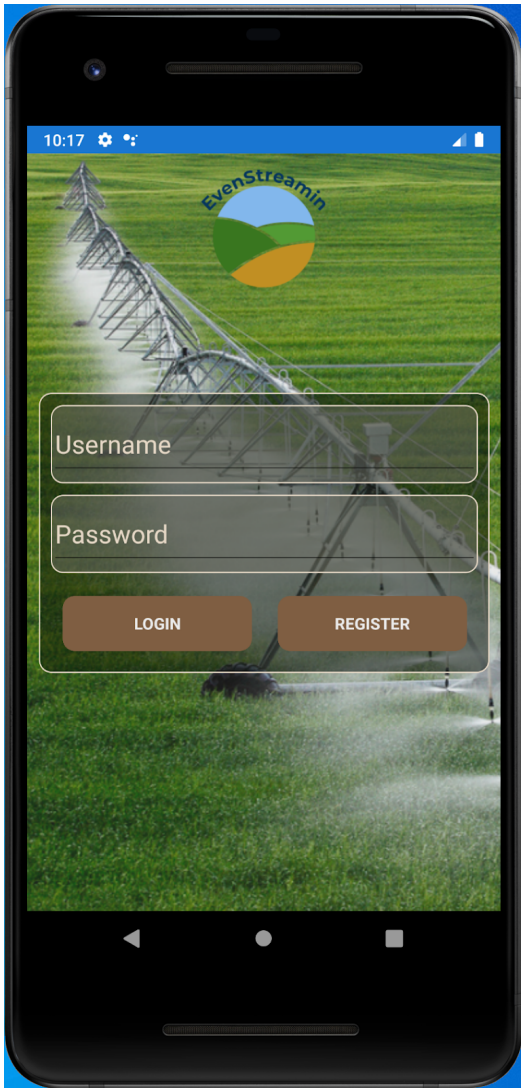


Figure 68: Android

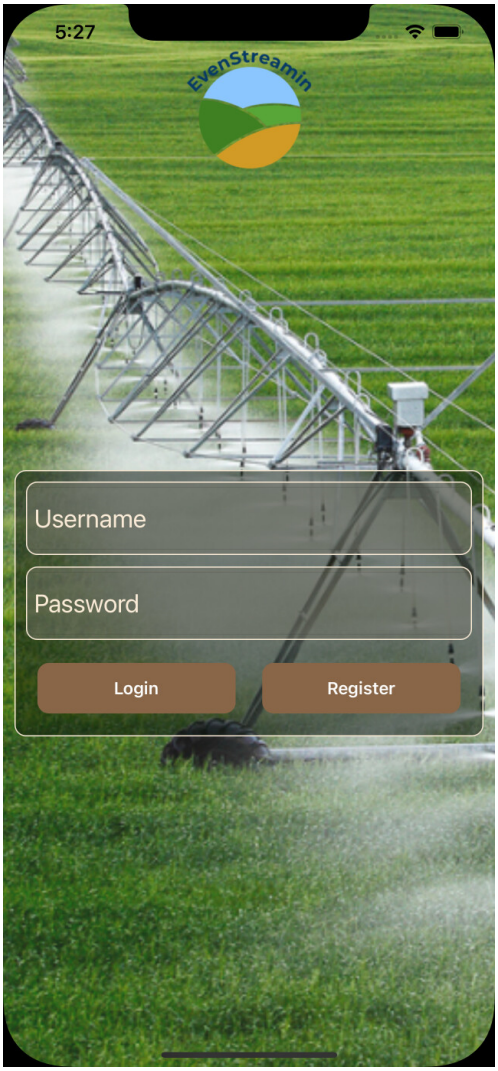


Figure 69: iOS

Registration User Exists Alert

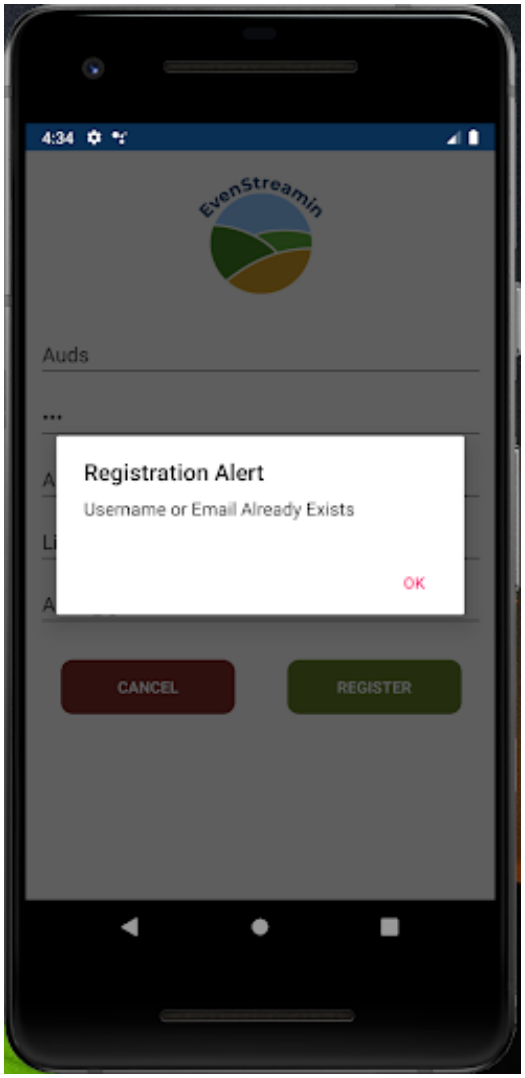


Figure 70: Android

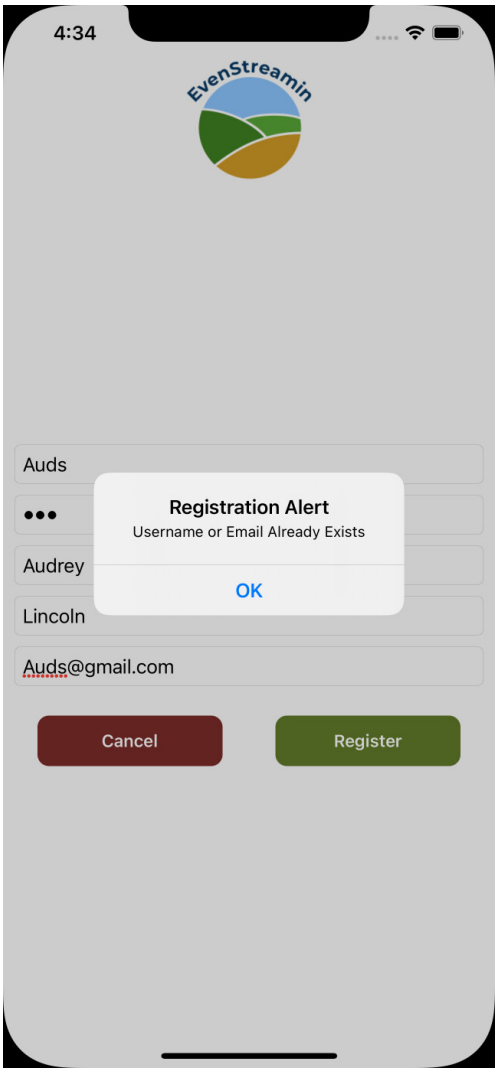


Figure 71: iOS

Registration Fill Out Alert

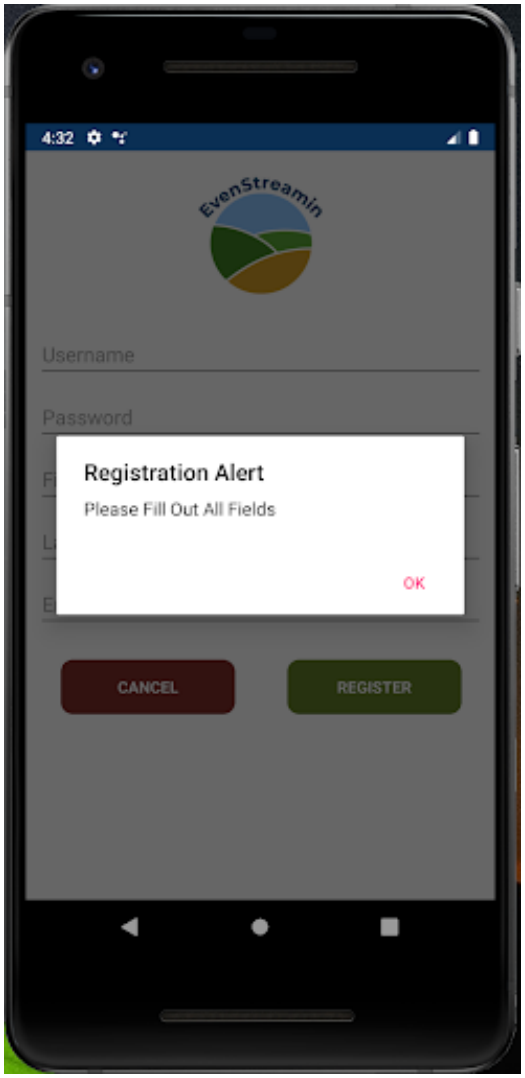


Figure 72: Android

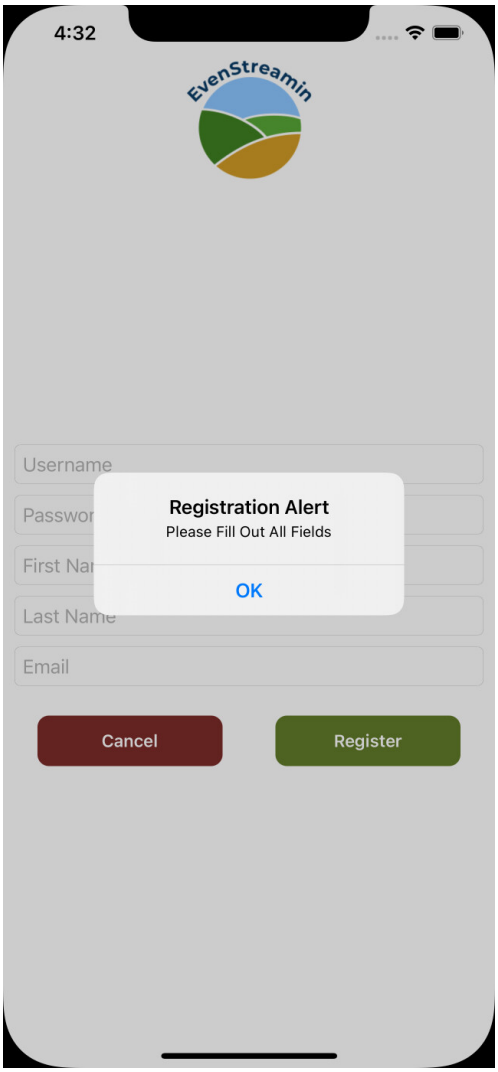


Figure 73: iOS

Registration

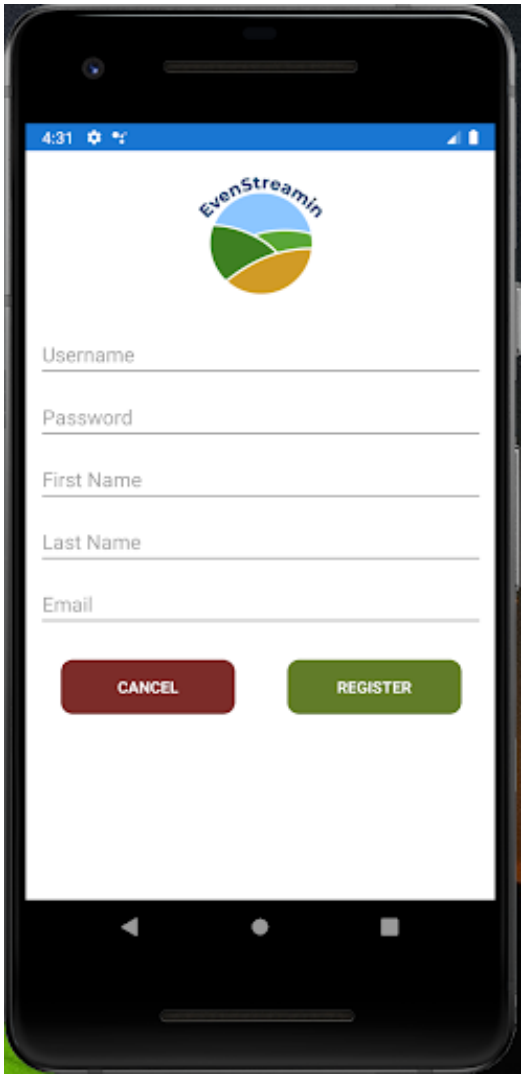


Figure 74: Android

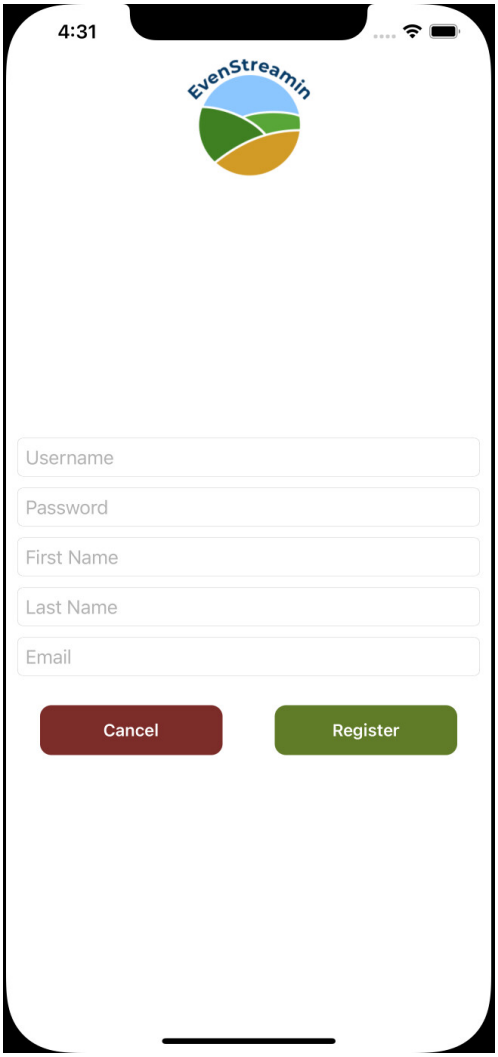


Figure 75: iOS

Running Field

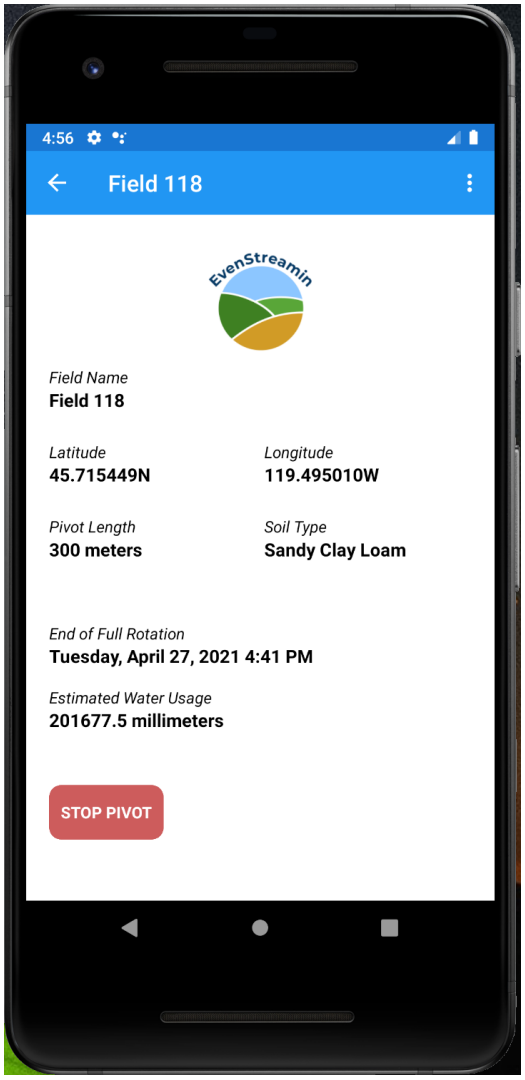


Figure 76: Android

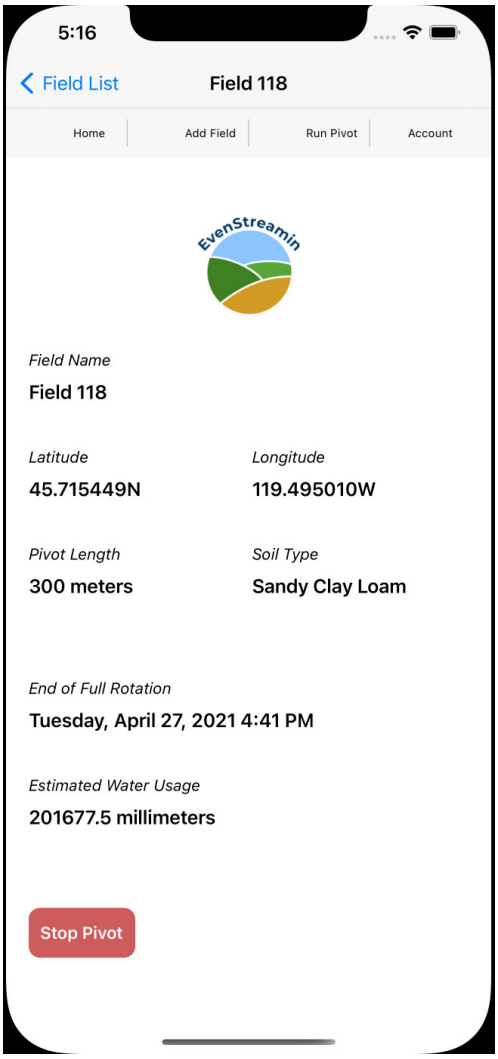


Figure 77: iOS

Run Pivot Alert

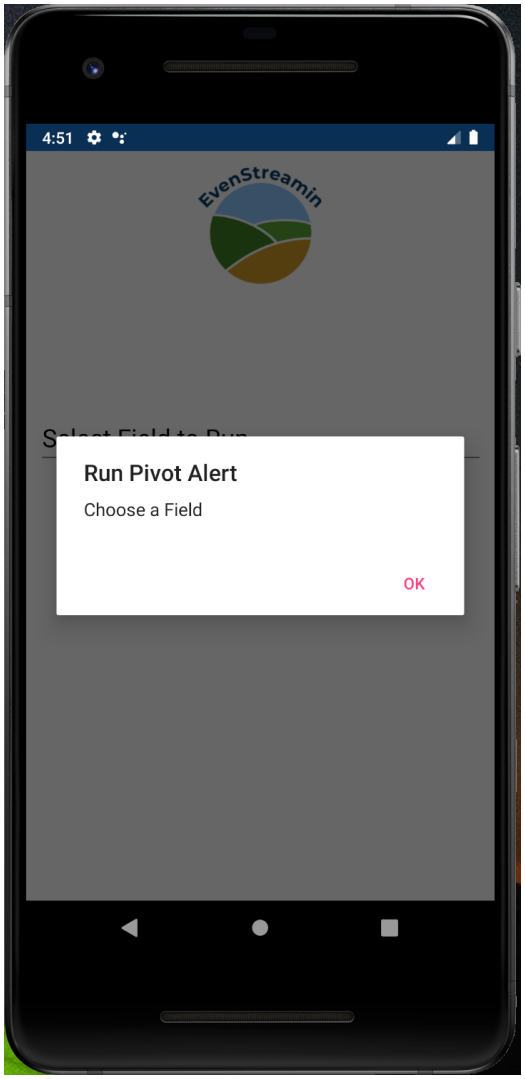


Figure 78: Android

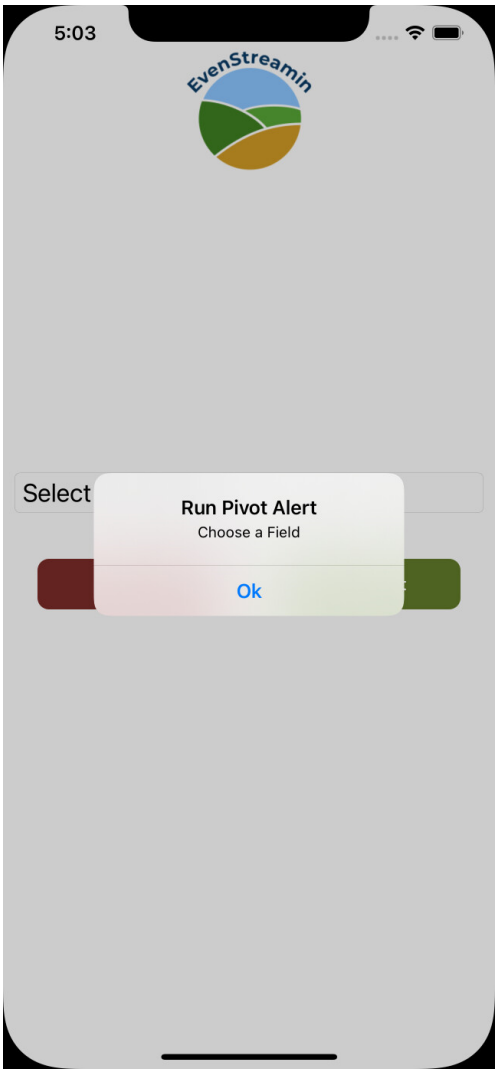


Figure 79: iOS

Run Pivot Selection

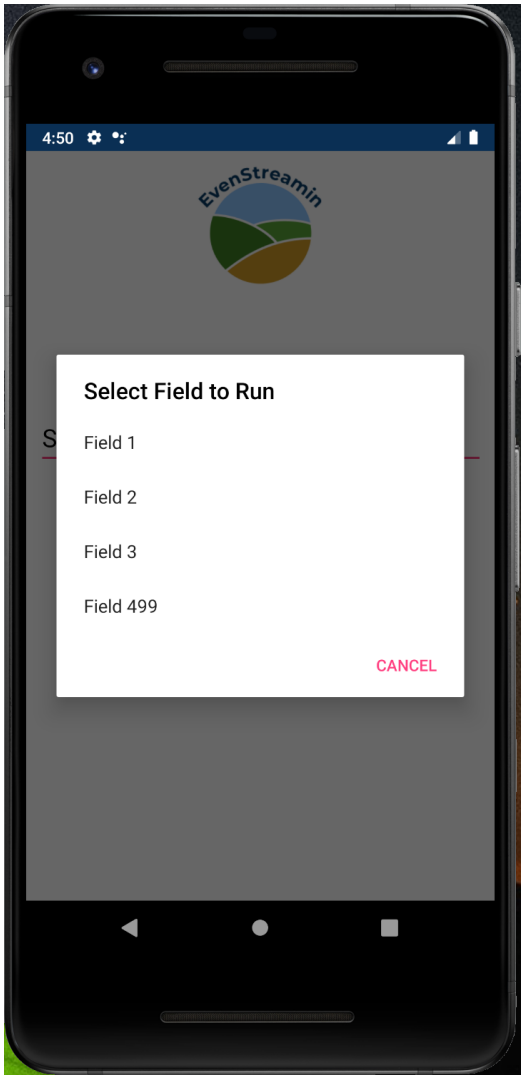


Figure 80: Android

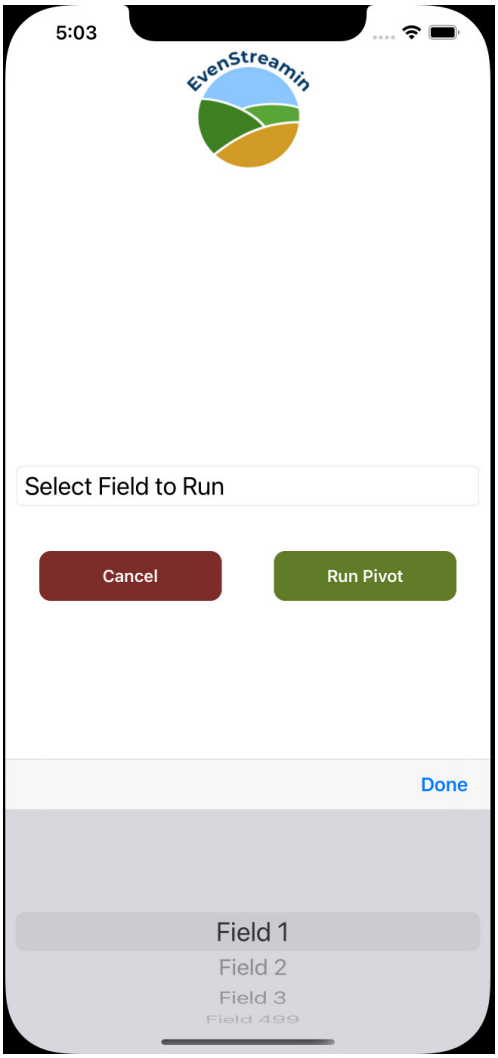


Figure 81: iOS

Run Pivot

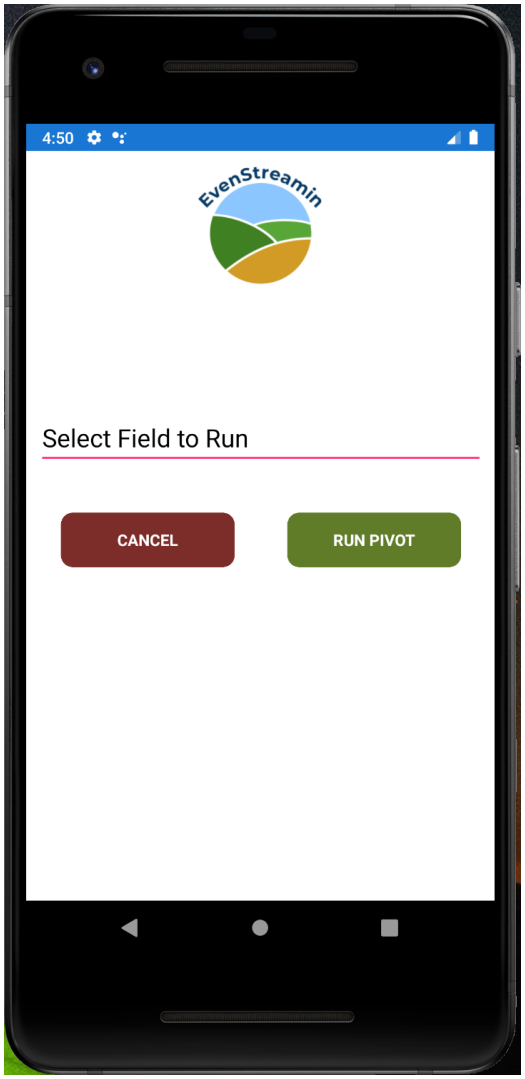


Figure 82: Android

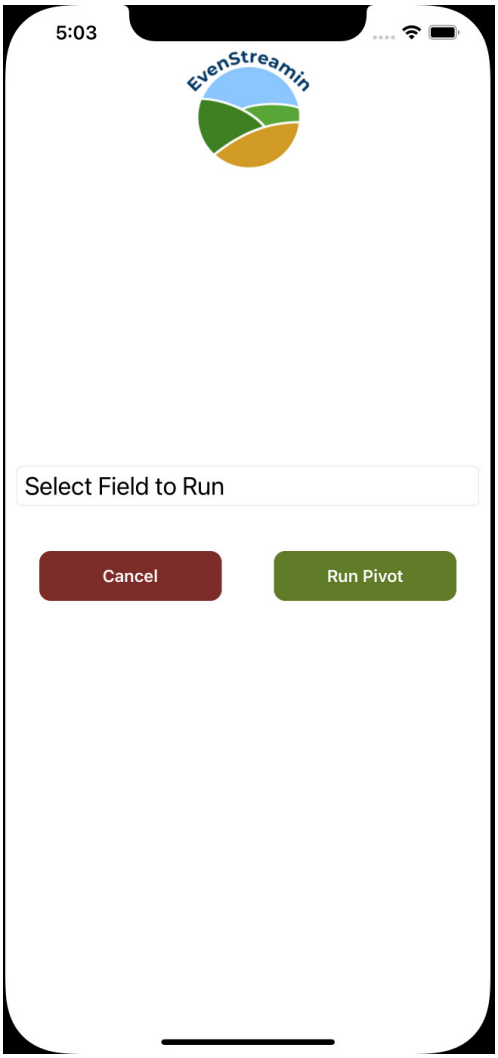


Figure 83: iOS

Stopped Field Edit

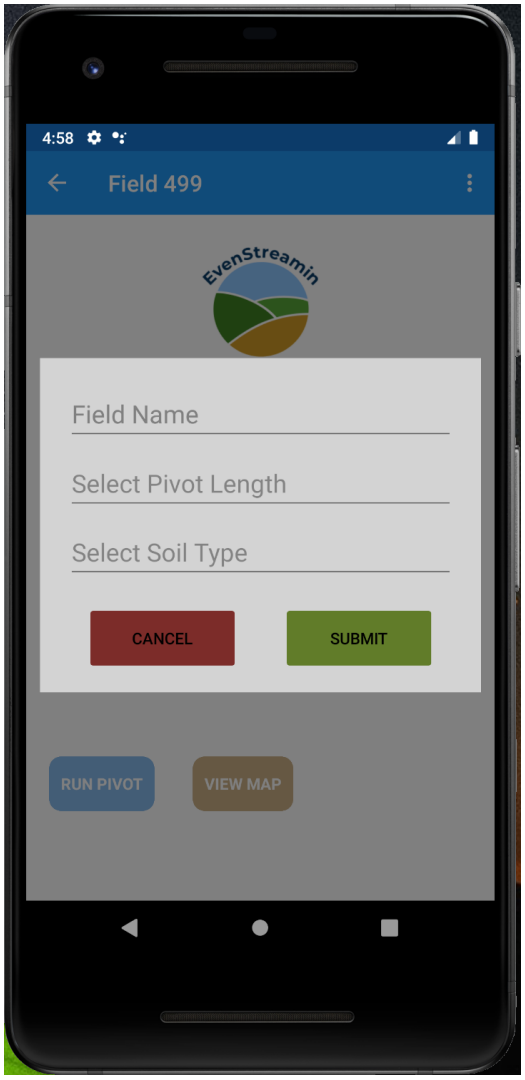


Figure 84: Android

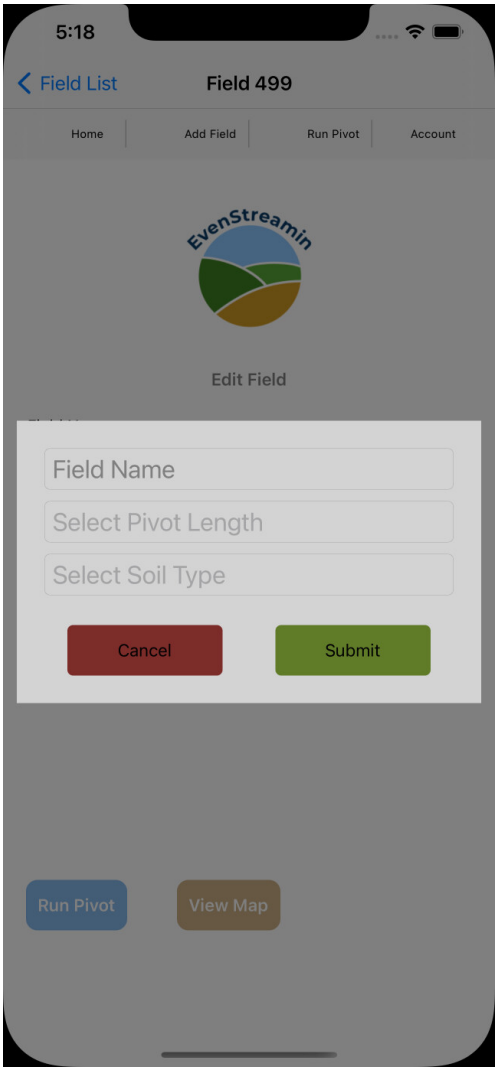


Figure 85: iOS

Stopped Field Edit Alert


Figure 86: Android

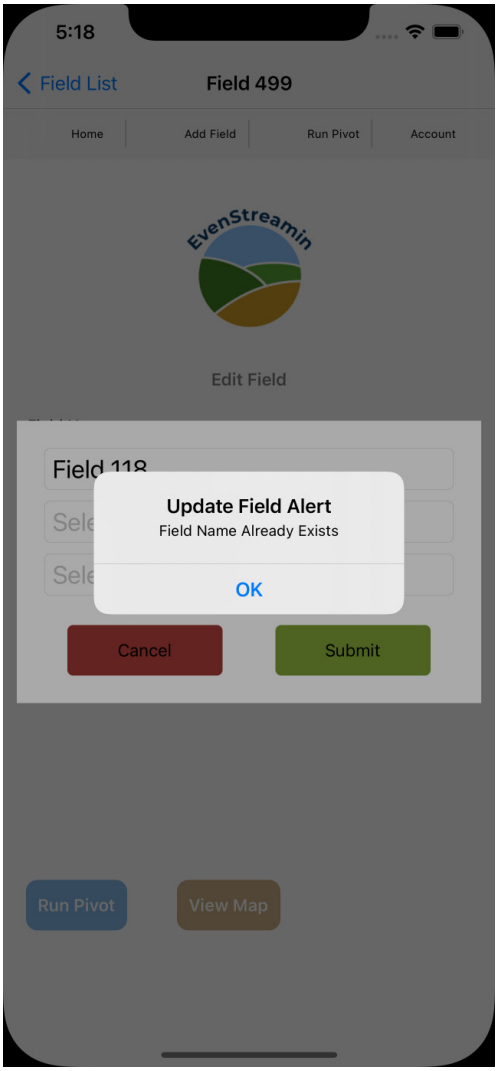


Figure 87: iOS

Stopped Field Map View


Figure 88: Android

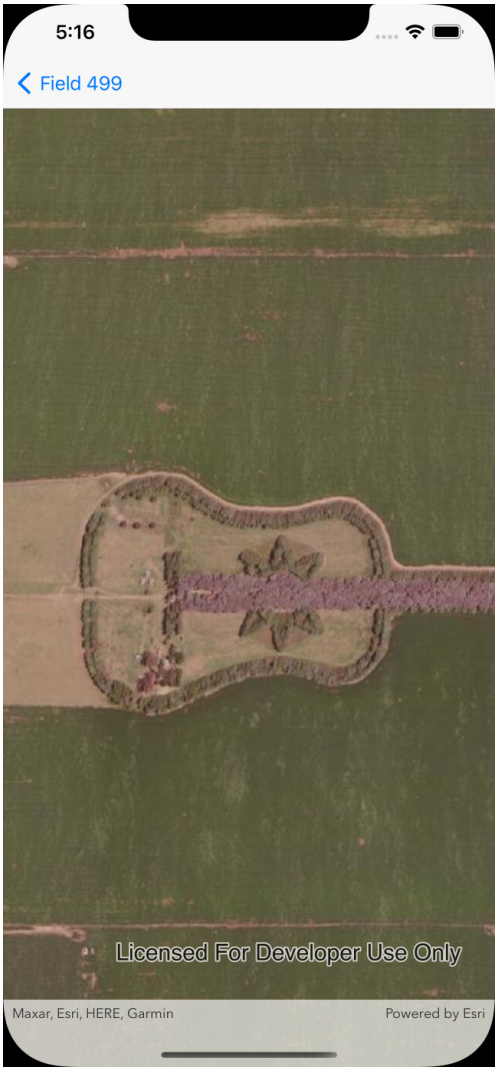


Figure 89: iOS

Stopped Field

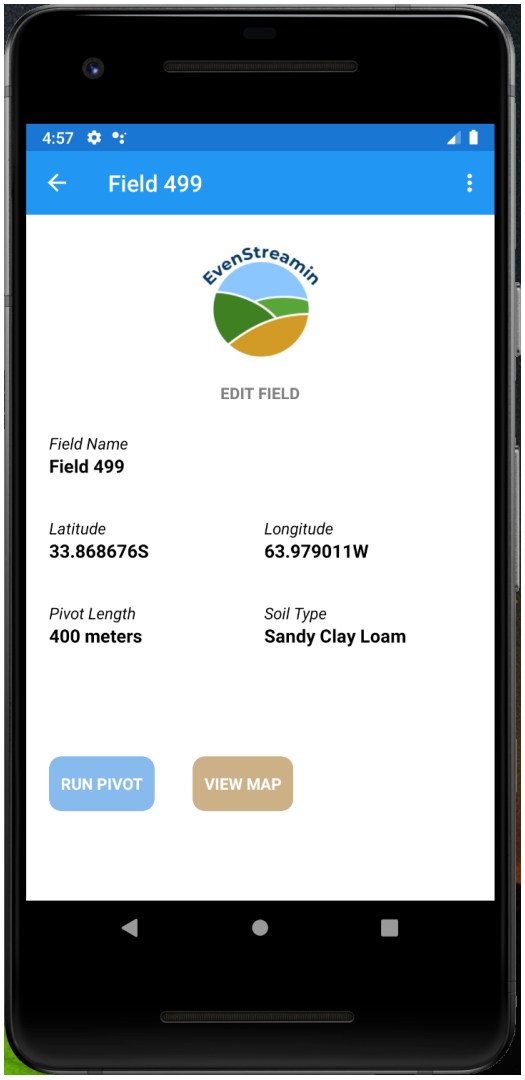


Figure 90: Android

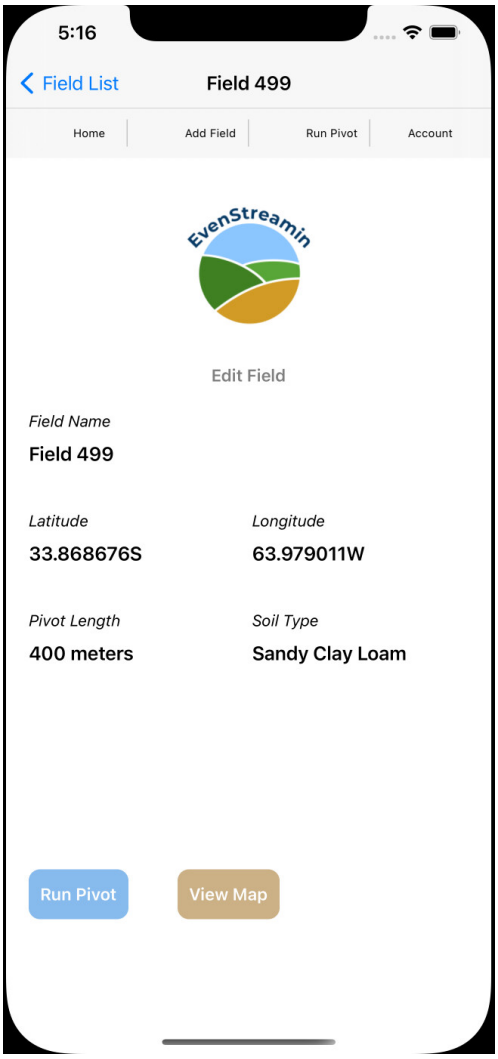


Figure 91: iOS

Toolbar

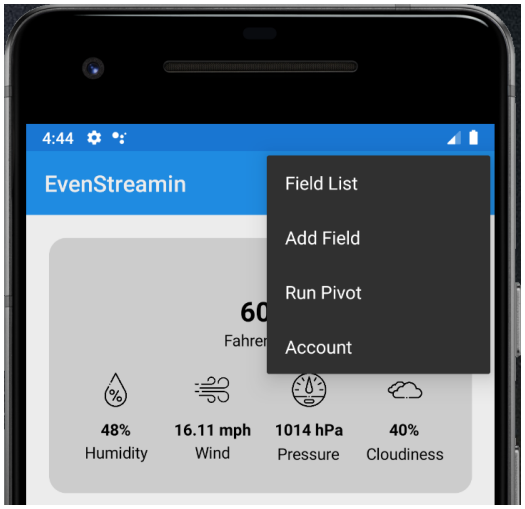


Figure 92: Android


Figure 93: iOS

## 7.2 *Appendix 2 - CS Fall Sprint Notes*

### 7.2.1 *Fall Sprint 1*

Localization

- Hardware Localization
  - Add GPS Tracking with the Google Maps API to Your Raspberry Pi Project
  - How to read GPS data with Python on a Raspberry Pi
  - How to Use a GPS Receiver With Raspberry Pi 4
- Software/API Localization
  - Qt Positioning API
  - Xamarin Forms Geolocation

Platform Research

- Cross-Platform Mobile Development
- 11 Popular Cross-Platform Tools for App Development in 2020
- 7 Popular Cross-Platform App Development Tools That Will Rule in 2020

Xamarin Research

- Xamarin documentation
- Xamarin.Forms Documentation
- How to Run a Xamarin.Forms iOS App from Windows (Using Visual Studio 2019)
- eXtensible Application Markup Language (XAML)
- Microsoft Docs on Nuget

ArcGIS

- What is ArcGIS Online
- Get started with ArcGIS Online
- ArcGIS for Student

ArcGIS and Xamarin

- ArcGIS Runtime SDK for .NET
- ArcGIS for Developers

- ArcGIS Runtime SDK for .NET: Building Xamarin Apps

Database Research

- Store Data in a Local SQLite.NET Database
- Xamarin.Forms Local Databases
- Working With SQLite In Xamarin.Forms Application

### 7.2.2 *Fall Sprint 2*

Xamarin Hello World

- Getting Started With Xamarin Using Visual Studio 2019 for Android and iOS
- How to Build a Xamarin Forms App

App Development

- Xamarin.Forms ListView
- Xamarin.Forms CollectionView Layout
- Xamarin: Open page from string
- Xamarin.Forms Navigation in C# — Xamarin 101 [10 of 11]
- Adventures in Mobile
- Understanding Xamarin Forms Navigation
- Xamarin.Forms Grid
- Xamarin.Forms ToolbarItem

UI Design

- Xamarin forms Good Looking UI Samples pages
- Absolute Layout Bounds for centering - Stack overflow
- Free Logo Design
- XamUI Login Page UI Kit

### 7.2.3 *Fall Sprint 3*

App Development

- How to use Navigation.InsertPageBefore in a ViewModel class in xamarin forms?
- Dropdown In Xamarin Forms Dropdown List Picker Xamarin form Dropdown
- Xamarin.Forms.PancakeView

## 7.3 *Appendix 4 - CS Spring Sprint Notes*

### 7.3.1 *Spring Sprint 1*

Weather API

- 03 - Working With Weather API — Complete Mobile App In Xamarin Forms - The Weather App
- OpenWeatherMap
- Agriculture Monitoring API

### 7.3.2 *Spring Sprint 2*

Soil Type API

- ArcGIS Soil Type Map
- Ambee Soil Api
- Soil Grids REST API

Visualization

- Next Level Maps With ArcGIS For .NET — The Xamarin Show
- ArcGIS Set initial map location
- ArcGIS Display Map

### 7.3.3 *Spring Sprint 3*

Azure

- Microsoft Azure for Beginners: Introduction - Scott Duffy
- Windows Azure Platform explained
- What is Microsoft Azure and How does Microsoft Azure Works
- Mastering Xamarin.Forms - very helpful book goes along with this Github Repo
- Table storage - Azure Docs
- Azure Storage documentation - Azure Doc

- Updating and Deleting Table Storage Entities with Azure Functions
- How do I update that entity's property value? - StackOverflow
- Azure Functions Table Binding: How do I update a row? - StackOverflow
- Integrating Azure Table Storage To Azure Function - C#Corner
- CloudTable Class - Microsoft Docs
- Azure Developer Tutorial: HTTP Trigger Functions with C# and .NET Core
- Azure Function returns 500 internal server error - Stack Overflow
- Explore Storage Account of Azure Functions
- HTTP Methods
- Learning Postman
- Android and iOS apps using Xamarin + Azure

UI Updates

- Weather fields in API response
- Part 2. Essential XAML Syntax

Weather Icons

- FlatIcon Weather 214
- FlatIcon Weather 142
- FlatIcon Weather 255

## 7.4 *Appendix 5 - Math Notes*

- Understanding Center Pivot Application Rate
- Managing sprinkler irrigation systems
- Reducing and Evaluating Irrigation Runoff
- LESA Sprinklers
- Inherent Factors Affecting Soil Infiltration
- CENTER PIVOT IRRIGATION SYSTEM LOSSES AND EFFICIENCY
- Development of the revised USDA–NRCS intake families for surface irrigation
- SPRINKLER PACKAGE WATER LOSS COMPARISONS
- Operating Characteristics of Center Pivot Sprinklers
- How a Center Pivot Irrigation Machine Works
- Pivot Basics
- VARIABLE RATE IRRIGATION ON CENTER PIVOTS. WHAT IS IT? SHOULD I INVEST?