

Spring 2015

A nearly autonomous, platform-independent mobile app for manure application records

Matthew R. Koester

Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_theses



Part of the [Bioresource and Agricultural Engineering Commons](#)

Recommended Citation

Koester, Matthew R., "A nearly autonomous, platform-independent mobile app for manure application records" (2015). *Open Access Theses*. 483.

https://docs.lib.purdue.edu/open_access_theses/483

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

**PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance**

This is to certify that the thesis/dissertation prepared

By Matthew R. Koester

Entitled

A NEARLY AUTONOMOUS, PLATFORM-INDEPENDENT MOBILE APP FOR MANURE APPLICATION RECORDS

For the degree of Master of Science

Is approved by the final examining committee:

Dr. Dennis Buckmaster

Chair

Dr. Danial Ess

Dr. James Krogmeier

Dr. Mark Tucker

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Dr. Dennis Buckmaster

Approved by: Dr. Bernard Engel

Head of the Departmental Graduate Program

4/29/2015

Date

A NEARLY AUTONOMOUS, PLATFORM-INDEPENDENT MOBILE APP FOR
MANURE APPLICATION RECORDS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Matthew R. Koester

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2015

Purdue University

West Lafayette, Indiana

ACKNOWLEDGEMENTS

I would like to thank my committee, Aaron Ault, Andrew Balmos, Shawn Ehlers, Elizabeth Hawkins, Alex Layton, and Sam Noel for technical expertise and guidance along the way.

I would also like to thank my wife for her support throughout this process.

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	viii
CHAPTER 1. INTRODUCTION	1
1.1 Background.....	1
1.2 Research Objectives	3
CHAPTER 2. LITERATURE REVIEW	5
2.1 Value of Manure.....	5
2.2 Manure Plans and Records	9
2.2.1 Existing Manure Management Tools.....	10
2.3 Regulatory compliance.....	12
2.4 Manure Spreading Calibration	15
2.5 Georeferenced Manure Application	18
2.6 Sample Applications That Expand Mobile Device Functionality	19
2.7 Minimizing User Input	21
2.8 Developmental Tools.....	22
2.8.1 Application Program Interfaces and Libraries	22
2.8.2 Hardware.....	24
CHAPTER 3. METHODS	27
3.1 Functional Specification and Interface Design.....	27
3.2 Brief Introduction to Web Development	32
3.2.1 Hyper Text Markup Language (HTML).....	32
3.2.2 JavaScript.....	33

	Page
3.2.3 Cascading Style Sheets (CSS)	36
3.3 Development of version 1.0	37
3.3.1 UI Development.....	38
3.3.2 Implementation of UI functionality	43
3.4 Development of Version 2.0.....	45
3.4.1 User Interface Improvement	45
3.4.2 Implementation of Google Maps API.....	50
3.4.3 Conversion to Native applications	53
3.5 Development Version 3.0.....	54
3.5.1 Spreader Identification and Status	54
3.5.2 Multi-Device Synchronization.....	57
3.5.3 Automatic Field Identification.....	65
3.6 Sensor Testing and Threshold Development.....	65
CHAPTER 4. RESULTS.....	68
4.1 Final UI.....	68
4.2 Map Implementation	70
4.3 Test Results	72
CHAPTER 5. CONCLUSION AND RECOMMENDATIONS	79
5.1 Conclusion.....	79
5.1.1 Version 1.0.....	79
5.1.2 Version 2.0.....	80
5.1.3 Version 3.0.....	80
5.2 Recommendations	81
5.2.1 Future Work	81
LIST OF REFERENCES.....	85
Appendix A Functional Specification	89
Appendix B Manure Records Documents	100
Appendix C Creating User's Remote Database.....	104

LIST OF TABLES

Table	Page
Table 1: Estimates of total production and contained nutrients in manure from various animals (Moore and Gamroth, 1993).....	7
Table 2. Percentage of manure nutrients useful for cropping for various storage types (Moore and Gamroth, 1993).	8
Table 3 Common manure conversions (Beegle, 2003).....	18
Table 4: Results of resultant acceleration	74
Table 5: Acceptable acceleration calculated from speed (rpm) & radius (in) highlighted in green. Cells colored in red generate excessive acceleration and could lead to sensor unreliability	75

LIST OF FIGURES

Figure	Page
Figure 1: Diagram showing different spreader types and the accompanying calculations for determining amount contained (Beegle, 2003).	17
Figure 2: Texas Instruments CC 2541 Sensor Tag (approximately actual size).....	25
Figure 3: Implementation of ISOBlue with a Tractor's CAN Bus (ISOBlue, 2014)	26
Figure 4: Approved UI Design of the Manure App Main Page/Dashboard.	29
Figure 5: Showing Spreader selection (a), spreader chosen for editing (b), and editing spreader description data (c).	30
Figure 6: Field selection (a.) and drawing a field boundary (b.).	30
Figure 7: Draft of generated manure record data.....	31
Figure 8: Version 1.0 functioning UI showing dashboard (a.), add spreader form (b.), and generated spreader table (c.).	41
Figure 9: The record page of the manure app displaying records in a table format (total record data not shown).....	42
Figure 10: Version 1.0 dashboard (a.) and version 2.0 dashboard (b.).....	46
Figure 11: Updated UI tables for selecting the spreading implement Version 1.0 (a.) and Version 2.0 (b.).	47
Figure 12: Dashboard not spreading (a.) and during spreading event (b.).	48
Figure 13: Created map pages: field boundary editor (a.), and operation map page (b.).	49

Figure	Page
Figure 14: Interface of field data entry form (a.) and field list page (b.).....	51
Figure 15: Showing the required files to convert into a native app using Cordova (PhoneGap, 2014)	54
Figure 16: Cloudant database UI displaying all databases created by the Manure App...	61
Figure 17: Cloudant database showing the record objects contained within the records database.....	62
Figure 18: Cloudant database showing the selected record object's characteristics.	63
Figure 19: Database credential input to enable multi-device synchronization.	64
Figure 20: Final dashboard design (a.), spreading overlay and animation (b.), and paused indication for when the user wants to pause the unloading process (c.).....	69
Figure 21: Final UI of the object lists: spreader list showing the available spreading implements (a.), field list showing available fields (b.), source list showing available sources (c.), and operator list showing available operators (d.).....	70
Figure 22: UI of connection process to the Bluetooth sensor tag. The main page not connected (a.), the app is scanning for a sensor tag after the start button has been pressed (b.), connection to the Bluetooth tag achieved (c.), reading data from the sensor tag's accelerometer (d.).	72
Figure 23: Sample data from the rotation of the sensor at varying speeds.	73
Figure 24: Test location for simulated manure application.	76
Figure 25: Spread paths displayed on map from simulated spreading operations.....	77
Figure 26: Exported manure application records displayed as a CSV file.	77

ABSTRACT

Koester, Matthew R M.S., Purdue University, May 2015. A Nearly Autonomous, Platform-Independent Mobile App for Manure Application Records. Major Professor: Dennis Buckmaster.

A major part of modern manure management is accurate application records; a key to their creation and maintenance is ease. This project involved the integration of existing technologies (smartphones, Bluetooth tags) in mobile web and native Android applications (apps) which enable the autogenic creation and upkeep of manure hauling records. This approach greatly improves the efficiency of the recording process which should help to improve the management of applied nutrients. Features of the app include: computation of a suggested travel speed to ensure target nutrient application (based on desired application rate and spreading width); minimized keystrokes/screen taps to accurately capture data for source, date, time, spreader, operator, georeferenced spread path, and field ; and data export for later aggregation and analysis. Autonomous operation was facilitated with a Bluetooth capable sensor tag which can automatically detect the spreader identity and spreading status (via accelerometer readings). The GPS capability of mobile devices facilitated the automatic detection of field and the creation of the georeferenced spread path.

The app was developed in stages and initially developed as a web app; Apache Cordova was then used to convert the code into a native app which can operate in the

background, giving near autonomous operation. This app approach could be readily adapted to other field operations in agriculture and related industries.

CHAPTER 1. INTRODUCTION

1.1 Background

Livestock operations are widely varying in size with some being very large (thousands, even tens of thousands of animals); this increases the need for management efficiency – including the management of the manure. Manure, when properly managed is a resource that can be of great value to cropping enterprises. A major part of manure management is the need for accurate and complete manure hauling records – to increase the efficient hauling and proper nutrient application and to ensure regulatory compliance. This can be a stressful and time consuming task for workers in a livestock operation. There are devices and technologies (e.g., smartphones, Bluetooth tags, and CAN bus messages) that can help workers and managers with autogenic (autogenic means created with semantic meaning without the need for manual human input; Welte, et al., 2014) data collection. Recently developed mobile technologies are capable of complex applications which automate work orders and other record keeping tasks. By using mobile applications capable of running on multiple mobile platforms and synchronized with cloud storage of records, the goal of automatically creating and maintaining accurate manure application records is within reach. Furthermore, a successful approach with automatic records could be applied to many other operations in agriculture, e.g., tillage, planting, tiling, harvest, spraying, etc. Accurate creation and upkeep of these manure

hauling records will help operations minimize human error, increase labor efficiency, decrease tedium, and likely contribute to better crop utilization of the nutrients provided. It is normal for operators to forget (or simply choose not) to write things down (generate records), for one reason or another; even if it is done, it may not be done in a timely manner nor correctly. Apps with autogenic capabilities will not forget nor procrastinate.

Livestock manure, being a key macronutrient source, should be applied according to a nutrient management plan; this logically leads to a certain application speed which is related to application rate, load size, and unloading time, and spread width.

Unfortunately, many livestock operations spread manure at a rate which conveniently fits the spreader capacity to a multiple of the length of the field in which application is occurring. Application of the proper amount of nutrient to a specific area of land is important for the utilization of materials that benefit the yield of crops; but additionally, is important for insuring the amount applied is utilized and does not migrate off of the land due to run-off when a rain event occurs. Runoff concerns pertaining to the over application of nitrates and phosphates can lead to immense environmental effects such as the eutrophication of both coastal waters and freshwater sources (Massey and Gedikoglu, 2011). This can be reduced by imposing nutrient limitations at the operation level. An application which uses N, P, and K nutrient data for a specific manure source to calculate the proper rate for the desired nutrient limitation on a current field would be extremely helpful and likely contribute to an improved nutrient distribution.

1.2 Research Objectives

This project bridges livestock and crop production systems and will meet a critical need for record keeping and decision making. The goal of this research was to develop a mobile application (the “Manure App”) with autogenic capabilities. The project proves and demonstrates the concept of autogenic applications which can improve data collection efficiency data quality, and therefore agricultural systems management. The specific objectives of this work were to:

1. Create a functioning Manure App version 1.0 capable of :
 - Recording a manure application event’s date, time and operator.
 - Recording load and field information.
 - Exporting usable data in the form of a comma separated values (CSV) file.
2. Refine the Manure App version into version 2.0 which will include all aspects of version 1.0 and additionally:
 - Improve the user interface (UI).
 - Use device’s GPS sensor to track load spread path.
 - Utilize the Google Maps application program interface (API) to draw field boundaries and show spread path and field completion.
 - Convert the web app into a native app using Apache Cordova open source software.
3. Implement autogenic capabilities into version 3.0 including basic structure and polished UI in versions 1.0 and 2.0 with additional optional features of:
 - Minimized user input through reduced tap events on the device required for data generation and collection.
 - Utilize web-based database for multiple device syncing.
 - Implement device geolocation to auto generate data based on location speed and geo-fencing of field boundaries.

- Enable Bluetooth connection to tags installed on spreader to derive spreading status, and additionally, quick data gathering of implement characteristics.
4. Evaluate the data collection process and overall management potential :
- Utilize the app during simulated hauling events to assess functionality.

CHAPTER 2. LITERATURE REVIEW

2.1 Value of Manure

Manure was considered a valuable commodity long before the use of commercial fertilizers (Lemmermann and Hehrens, 1935; Haynes and Naidu 1998). Farmers spread manure over their fields to replace consumed nutrients and increase the organic matter level in the soil. Using manure for soil nutrient replenishment has sustained agriculture for centuries. By the 20th century, some concentrated animal operations in certain regions sometimes dealt with manure as a waste material (Nowak, et al., 2005). Manure, when managed properly, can be considered a valuable byproduct of livestock operations even when compared to the efficiency of additional commercial fertilizers.

The work of Massey and Gedikoglu (2011) explored the economic impact of spreading manure according to three different nutrient limiting practices. First (N-limiting), investigating the practice of applying manure based on available N in the manure and the amount of N that can be utilized by the crop in the next growing year; second (P-limiting), applying manure by limiting the amount of available phosphorus that the crop can utilize in the next growing year; and third (P-banking), applying manure at a rate that allows a buildup of phosphorus to a level that can be utilized by the subsequent crops raised on a given piece of land before the next application event occurs. For example, using N as the limiting nutrient for application on maize would mean that one

would apply manure at the rate that the crop will use for the desired yield. Using P as the limiting nutrient is generally more restrictive which requires more land. As a consequence, P-limiting requires more labor and machine time due to further travel and more time in the field spreading manure. P-banking is similar however, it takes into account the phosphorus utilized by the subsequent years crop. P-banking is a logical approach when considering a maize/soybean rotation. Manure application would occur before the planting of maize, the growing season would take place and then harvest of the maize would take place. Soybeans would then be planted and harvested. P-banking would allow the manure to be applied at the rate of N utilized by the maize and then the remaining P would be utilized by the soybeans. Application rates would be based upon the amount of N utilized by the coming crop but allowing P to build up for the next growing season. Their (Massey and Gedikoglu, 2011) work found that P banking was the most profitable and that manure can have an economic benefit which offsets the expense of the purchase of artificial fertilizers. Manure was applied to 1600 acres of land using the P-banking method of limiting nutrients with a net economic advantage of \$2,178. This value was purely due to the savings incurred from applying manure rather than buying commercial fertilizer. This value does not reflect any other benefits that manure has on the soil profile such as: improved microbial activity and increased amount of soil organic matter.

Utilizing manure's true value (as well as properly determining application rates) requires testing the nutrient content since nutrient content varies due to the livestock feeding regimen, and storage and application handling practices. Manure type, storage system and application practices can be used to get an estimate of nutrient content for

proper application (Moore and Gamroth, 1993), but sampling and testing are recommended. The following tables show the estimated amount of manure and contained nutrients created by various animals each day and the percentage of remaining nutrient levels based on manure storage and application practices (Moore and Gamroth, 1993).

Table 1: Estimates of total production and contained nutrients in manure from various animals (Moore and Gamroth, 1993).

Animal	Animal size (lb)	Total manure production			% water	Nutrient content			
		lb/day	cu ft/day	gal/day		N lb/day	P lb/day	K lb/day	
Dairy	150	13	0.19	1.5	88	0.06	0.011	0.04	
	250	22	0.32	2.4	88	0.11	0.023	0.07	
	500	43	0.66	5.0	88	0.22	0.047	0.15	
	1000	89	1.32	9.9	88	0.45	0.094	0.29	
	1400	120	1.85	13.9	88	0.59	0.131	0.41	
Beef	Cattle	500	30	0.50	3.8	91	0.17	0.051	0.12
		750	45	0.75	5.6	91	0.26	0.079	0.19
		1000	60	1.0	7.5	91	0.34	0.109	0.24
		1250	65	1.2	9.4	91	0.43	0.12	0.31
		Cow	63	1.05	7.9	91	0.36	0.11	0.26
Swine	Nursery pig	35	2.9	0.038	0.27	89	0.018	0.0052	0.01
	Growing pig	65	5.3	0.070	0.48	89	0.034	0.0099	0.02
	Finishing pig	150	12.4	0.16	1.13	89	0.078	0.023	0.045
		200	16.6	0.22	1.5	89	0.104	0.036	0.059
	Gestate sow	275	11.3	0.15	1.1	89	0.069	0.023	0.04
	Sow and litter	375	15	0.21	1.4	89	0.1	0.031	0.054
	Boar	350	14	0.19	1.4	89	0.081	0.023	0.051
Sheep	100	4	0.062	0.46	87	0.045	0.0066	0.032	
Poultry	Layers	4	0.26	0.0035	0.027	84	0.0034	0.0012	0.0012
		2	0.17	0.0024	0.018	78	0.0024	0.0006	0.0008
		Broilers	2	0.17	0.0024	0.018	78	0.0024	0.0006
Horse	1000	51	0.75	5.6	85	0.31	0.072	0.25	

Table 2. Percentage of original manure nutrients remaining for crop utilization with various storage types (Moore and Gamroth, 1993).

Method	Beef			Dairy			Horse			Poultry			Sheep			Swine		
	N	P	K	N	P	K	N	P	K	N	P	K	N	P	K	N	P	K
Daily spread				80	90	90	75	90	90	65	90	90	75	90	90			
Dry (with roof)				70	90	90	70	90	90	60	90	90	65	90	90			
Earthen storage	65	80	85	55	60	70										60	60	70
Lagoon/flush	30	40	60	30	40	60				25	40	60				30	40	60
Open lot	60	70	60	60	70	65	60	70	65				55	70	60	60	70	65
Pits (slats)	75	95	95	75	95	95				70	95	95	75	95	95	75	95	95
Scrape/storage tank	70	85	90	70	90	90												
None (grazing)	100% of nutrients retained																	

There are other tools available on the Web that use specific material data to calculate the net worth of manure nutrients (University of Michigan Extension, 2013; UNL Water, 2014; Ohio State University Extension, 2009).

Manure application has been shown to increase the organic matter (OM) in soils. Haynes and Naidu (1998) showed, in arable soils with OM content beginning at approximately 26 Mg ha⁻¹, that regular application of farmyard manure at a rate of 35 Mg ha⁻¹ (15.6 tons acre⁻¹) for a period of 140 years saw an exponential increase in organic matter. This plot had in excess of three times the amount of OM in the soil profile than plots that had no manure applied on them over this period of time. The manure applied plot had reached over 75 Mg ha⁻¹ (33.45 tons acre⁻¹) OM in that time period. Comparing OM content with another plot that had manure applied for 19 years (from the years 1852 to 1871) with no more manure applied from 1871 to 1986, the soil OM reduced but at a slow rate. After 104 years with no manure application, the OM in this plot was still significantly higher than in plots that had no manure applied during the 140 years.

Nitrogen alone can have a beneficial impact on soil's creation and maintenance of soil OM. By applying nitrogen to soil with a high level of plant residue enables soil microbes to utilize the added nitrogen to assist in the breakdown of plant residue with high carbon to nitrogen ratio. Gillespie, et al. (2014) determined effects of N source on soil OM in soils growing maize; the soils with manure applied had significantly more soil OM. The soil samples were taken from a long-term test plot initiated in 1992 with continuous implementation of five treatments, including:

- Maize with no N applied
- Maize with N applied at 200 kg ha⁻¹ as ammonium nitrate (NH₄NO₃)
- Maize with manure applied at 100 Mg ha⁻¹ of wet weight composted dairy manure
- Maize and soybean on a two year rotation with no added N
- Land left in fallow with no fertilizer applied.

Soil samples were taken in 2009 and found that the treatments with the highest levels of soil OM were the: maize with commercial fertilizer applied N having an soil OM of 18.4 g kg⁻¹, and soils with applied organic manure had a soil OM level of 54.0 g kg⁻¹. Manure applied soils had the greatest measure of OM (Gillespie et al., 2014).

2.2 Manure Plans and Records

In recent years, there has been a rise in concern with the over application of manure, as well as fertilizers and other chemicals on tillable land. Over application of manure has been found to lead to negative environmental effects from large amounts of nutrients migrating off of the land. Over application of manure can also result in inefficient use of the manure's nutrients. When manure is over applied, there is also an opportunity cost; those nutrients could have been properly utilized by other crops had the

manure been applied over a greater area. The practice of applying nutrients from manure and other sources at rates matching uptake will reduce environmental damage and improve profitability of cropping operations.

2.2.1 Existing Manure Management Tools

In order to document the amount of nutrient applied during manure application, keeping accurate manure application records are needed. Purdue University's Agronomy department has developed a Windows-based computer program called the Manure Management Planner (MMP) (MMP, 2014). This program allows the farmer to input information about the animal type, storage practices, field data for the planned application location, and information about the coming crop that will be utilizing the applied nutrients. The MMP uses the input data to determine if land area can accommodate the amount of manure generated, seasonal land availability, and manure storage capacity. It also checks the sufficiency of manure application equipment to apply the type and amount of manure generated by the operation.

The University of Nebraska has developed a manure management mobile app capable of calculating the value of manure nutrients applied. The app has three steps that the user can work through to gain the most accuracy in the calculation process (UNL Water, 2014). First, calculate the amount of material spread by inputting the specific implement characteristics. Second, calculate the amount of agronomic nutrient contained within the manure by looking up the book value or using manure test values. Third, calculate the economic value of the manure. The app keeps a history of past manure entries and uses email to export one entry or the entire list of entries. The Manure

Calculator is available on iTunes and Google Play store for a fee of \$0.99 (UNL Water, 2014).

Larry Theller (Larry Theller, Purdue Agricultural & Biological Engineering, personal communication, 10 April 2015) is currently developing INFertMapper, a mobile app that enables farmers to visualize regulation defined setbacks for clear indication of where fertilizer should not be applied. The app is supposed to help operations understand and comply with Indiana regulations on fertilizer application near sensitive environmental areas (public water supply, surface waters, sink holes, water wells, drainage inlets, property lines, and public roads). INFertMapper also calculates the total area of application surface and creates a suggested application rate. The project includes the development of a related web app for added functionality of tasks using a PC while keeping the mobile app version simple and easy to use.

A web tool was created by the University of Minnesota that calculates the value of manure on a given operation based upon their livestock, manure storage practices, type of bedding used, and application practices. The tool is in spreadsheet format and compares the cost of fertilizers purchased for an operation with and without the application of manure (University of Michigan Extension, 2013).

A spreadsheet tool for calculating manure value has been developed by the University of Nebraska-Lincoln and is available for free download (UNL Water, 2014). The tool accepts user input data based on the available nutrients in the manure and additionally, the soil nutrient content in the fields that are planned for manure application.

Ohio State University Extension has developed and made available a spreadsheet tool that calculates the desired manure application rates and the value of the manure

contained nutrients based on user input (Ohio State University Extension, 2009). The tool requires input of crop, expected yield for calculating the application rate. Manure nutrient data is calculated based upon input source nutrient values.

Using existing manure planning and rate calculation tools enable the creation of accurate manure application rates and records. Keeping accurate and up-to-date manure application records benefit the operation in documenting the total nutrient application to each field, improving nutrient utilization, and reporting for maintaining regulatory compliance.

2.3 Regulatory compliance

Regulatory compliance is one of the main motivations for keeping accurate manure hauling records. There are both federal, and state regulations with which livestock operations must comply or fines can be imposed up to \$27,500 per day for willful or negligent violations (Meyer and Mullinax, 1999), and criminal fines range from \$5,000 to \$50,000 per day and up to three years in prison (Ess, et al., 1996). Keeping complete and accurate manure hauling records are key for maintaining regulatory compliance and is required under law (Code, 2012a). The completion of a manure management plan is part of gaining approval for the construction of a Confined Animal Feeding Operation (CAFO). CAFOs in Indiana are defined by the numbers threshold of (IDEM, 2015):

- 700 mature dairy cows
- 1,000 veal calves
- 1,000 cattle other than mature dairy cows
- 2,500 swine above 55 pounds

- 10,000 swine less than 55 pounds
- 500 horses
- 10,000 sheep or lambs
- 55,000 turkeys
- 30,000 laying hens or broilers with a liquid manure handling system
- 125,000 broilers with a solid manure handling system
- 82,000 laying hens with a solid manure handling system
- 30,000 ducks with a solid manure handling system
- 5,000 ducks with a liquid manure handling system

In order to gain approval for the construction or expansion of a CAFO, the owner must submit a complete manure management plan. The manure management plan must consist of: manure nutrient test data, planned area application soil fertility test data, soil test frequency (minimum of every four years), manure test frequency (minimum of every year), and any other practices conducted for the proper management of facility created manure (Code, 2012a). The manure management plan includes the keeping of accurate manure application records. The application records must contain specific information and are required to be kept and maintained by the owner/operator of the CAFO.

Information that must be documented for in manure application records include (Code, 2012b):

- Expected Crop yields.
- The date of manure application occurred to each field.

- Precipitation events at the time of application and 24 hours preceding and following application.
- Test method used to analyze manure nutrients and soil fertility.
- Results from manure and soil testing.
- Explanation of the basis of determining manure rates to be applied.
- Calculations used for determining application rates.
- Total amounts of nitrogen and phosphorus applied to each field, including documentation and calculations used to determine totals.
- Method of application.
- Dates of equipment inspections.
- USDA soil survey maps of land application sites.
- Type of manure applied.
- Written conservation plan explaining the practices used completed prior to application to on if applying manure to highly erodible land.

The required record data is extensive and must be completed and documented in the operation's maintained manure application records. There are multiple manure record documents available from IDEM (see Appendix B) that can be used for the input of the specified data (IDEM, 2015). It appears that manure application records, aggregated by "load" will suffice and therefore, that is the aggregation level target of this work

2.4 Manure Spreading Calibration

Calibration is necessary to ensure the proper application rate of manure with a spreading implement. Knowing the rate at which the implement discharges the applied material, how much material is loaded into the spreader, and the width of the spread path under specific conditions are a few of the key factors that are needed for proper application. In order to determine the application rate (tons acre⁻¹ or Mg ha⁻¹), one can utilize common rate equations to calculate the speed of travel required to achieve the desired rate for a given spreading implement. By taking the equation for field capacity one can determine the most appropriate travel speed given the characteristics of the implement (Field and Solie, 2007 pp. 124 - 126).

$$C_a = \frac{S*W}{8.25} \quad (\text{eq. 1})$$

Where:

C_a = area capacity (acres/hour)

S = Average speed of travel (mph)

W = Effective width of the implement (ft.)

This equation was used to solve for recommended speed of application having known the desired application rate and the time required for the implement to unload the contained material. The edited equation is as follows:

$$S = \frac{\left(\frac{C_t}{T_m}\right)*8.25*60}{W*R_d} \quad (\text{eq. 2})$$

Where:

C_t = Weight capacity of the spreading implement (tons)

T_m = Amount of time the implement needs to fully unload (minutes)

R_d = Desired rate of application (tons/acre)

Alternatively, one can derive an approximate application rate by taking a measure of the spread path distance, the spread width, and total amount of material applied to calculate the rate of application; since this requires information from an actual spreading event, this is most applicable after the application process has occurred. The approximate capacity of a spreader can be found in manufacturer's specifications; however, more accurate data requires (ideally) weighing or alternatively computing volume and measuring density. Determining the volume of the spreader is fairly straight forward using the diagram of Figure 1. Additionally, in the absence of actual manure density data, the data of Beegle (2003) in Table 3 could be used for approximation.

SOLID OR SEMISOLID**[A] Box spreader (level load)***

$$\text{volume} = \text{length} \times \text{width} \times \text{depth}$$

[B] Box spreader (piled load)*

$$\text{volume} = \text{length} \times \text{width} \times [\text{depth} + (\text{stacking height}^{**} \times 0.8)]$$

[C] Round-bottom open-top spreader (level load)

$$\text{volume} = \text{length} \times \text{depth} \times \text{depth} \times 1.6$$

[D] Round-bottom open-top spreader (piled load)

$$\text{volume} = \text{length} \times \text{depth} \times 1.6 \times (\text{depth} + \text{stacking height}^{**})$$

LIQUID**[A] Box spreader (level load)***

$$\text{volume} = \text{length} \times \text{width} \times \text{depth}$$

[C] Round-bottom open-top spreader (level load)

$$\text{volume} = \text{length} \times \text{depth} \times \text{depth} \times 1.6$$

[E] Tank spreader (round)

$$\text{volume} = \text{length} \times \text{tank diameter} \times \text{tank diameter} \times 0.8$$

[F] Tank spreader (noncircular)

$$\text{volume} = \text{length} \times \text{width} \times \text{depth} \times 0.8$$

*For a box spreader with sloping sides, use an average width.

**Stacking height is the height of any mounded manure above level.

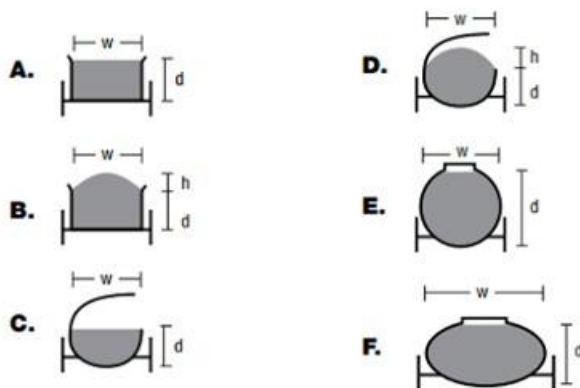


Figure 1: Diagram showing different spreader types and the accompanying calculations for determining amount contained (Beegle, 2003).

Table 3 Common manure conversions (Beegle, 2003)

TO CONVERT FROM	TO	MULTIPLY BY
bushels	cubic feet	1.24
gallons	cubic feet	0.134
gallons	pounds	8.3 (liquid)
gallons	tons	0.0041 (liquid)
tons	gallons	240 (liquid)
cubic feet	gallons	7.48
cubic feet	tons	0.031 (liquid) or 0.0275 (solid)
cubic feet	pounds	62 (liquid) or 55 (solid)

2.5 Georeferenced Manure Application

The concept of georeferencing manure application maps is not new. The work of Ess, et al. (1996) developed a system for site specific variable rate manure application. This was accomplished by installing a developed flow control and recording system onto a pull type manure set up for injection of liquid manure. The created system was able to generate as applied georeferenced application maps using commercial software. Additionally, the system used geographical information system (GIS) software for the creation of field boundaries. These field boundaries could then be displayed on the tractor monitor. As application occurred, the system constantly monitored ground speed, flowrate, and position. The information was combined to create as applied maps using

Rockwell Vision®. The system was able to create and display georeferenced application maps using differential GPS installed in the tractor.

2.6 Sample Applications That Expand Mobile Device Functionality

There are multiple applications that have been developed with the purpose of mobile location reminders and are currently available for free or for a small fee. iPhones running iOS 4 and later (nominally 2012 and later) have location based location services with location reminder capabilities built into the operating system (Apple, 2014). These apps and OS enable a mobile device to remind the user of predefined task based on the location of the device. For example, an individual with an iPhone 5 tells Siri, the mobile device's voice command interface, to remind them to pick up milk from the store when they drop off their children at school. Provided the device has the location of the school stored, once the device is within a predetermined range of the school location, the device will notify the user that they need to go buy milk from the store. This capability is included on iOS devices, but other devices can gain these capabilities on their devices by downloading similar apps from their respective app stores.

Geobells is a location reminder app available on the Google play store that enables the customization of the device based on the location (Geobells, 2015). This app allows the user to see all reminders on the map as well as customize device settings like the silencing of the device when entering a movie theater or church.

Geofencer is an Android app that is very similar to the Geobells app with added user control for power users (Geofencer, 2015). Users can set as many locations for geofences as they want and accompanying notifications. Each location geofence radius

size can be edited by the user to change the distance to the location that will trigger the user-created notification. The app uses the Google Geofence API to turn a specified Lat/Lng point into and notification trigger once a distance from the point is specified. Once the user is within the specified distance, the app triggers a notification.

Agent is another Android app that changes the user settings of the user's mobile device based upon the current activities of the user (Agent, 2015). The Agent app screens calls at night when it knows the user is sleeping, sets a marker on the a map when the user parks their car to help the user remember where they parked, responds to texts and reads out loud incoming messages when the user is driving, and automatically silences phones during predetermined meeting times created by the user.

The Automatic App is an application that pairs the user's mobile device to a link that plugs into the diagnostic port which is standard in on-road vehicles since 1996 (Automatic Labs, 2015). This port enables the harvesting of data from the user's vehicle that can then be used for driving analysis and coaching for better fuel economy achievement. By observing the driving style of the user and receiving data on fuel consumption from the vehicle's CAN, the app provides information on the driving statistics and suggestions for future improvement at the end of each trip. Additionally, the app has a safety feature built-in with the capabilities of calling for help when an accident occurs. The app can also decode check engine light codes for the user, coach new drivers, and help users remember where they parked their car.

The same functionality that enables many apps to remind users of tasks or change phone notification settings based on the user's current location could be utilized in data

collecting apps; recording operational data based on the users location and operation is often the goal.

2.7 Minimizing User Input

The work of Welte, et al. 2013 considered the importance of UI development as well as the ability to have autogenic functionality in mobile applications for agriculture. A suite of collaborative mobile applications called the Open Ag Toolkit (Open Ag Toolkit, 2014) were developed for the improvement of current Farm Management Information Systems (FMIS) design and operation data collection. On-farm data collection usually consists of hand written notes that in small notepads that can easily be lost or damaged rustling around inside a tractor cab or on vehicle dashboards. Useful apps that can replace the handwritten notes of the past are fairly easy to develop with the added benefit of multiple device synchronization of data, and the ability to back-up data for safe storage. However, creating an app that farmers will use is much more difficult. Most people will not use mobile apps for data entry if it appears to take more work, a higher learning curve, and longer entry times (Welte, et al., 2013).

Autogenic data collection capabilities were also a focus of app development. Mobile apps developed with the intent to be part of a FMIS could assist in the data collection efficiency. Welte, et al. (2013) suggested the use of inexpensive wireless Bluetooth sensors that could be installed around the tractor and implement for assisting in the autogenic data collection. Bluetooth is a low energy wireless communication standard for reliable communication up to distances of 100 meters and is used with many existing

apps for pairing with other devices to transfer data pertaining to voice, music, photos, and data (Welte, J., 2014).

Manure application is a fairly involved task, operators need to be focused on the task in order to operate the machinery to apply nutrients at the proper rate. An app with autogenic capabilities to minimize user input needed to generate accurate and complete manure application records would benefit operators, keeping their attention on the application task.

2.8 Developmental Tools

2.8.1 Application Program Interfaces and Libraries

There are multiple free tools that can be utilized in the effort of developing web apps capable of serving complex tasks. In addition to the basic development tools of Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript, the basic three elements of web development; there are other tools that can be utilized in the effort of creating more functionality while reducing the challenge of development. jQuery is a JavaScript library that enables developers to lessen complexity of, and write less, code for the development of web app functionality (jQuery, 2015). jQuery Mobile is a framework that utilizes jQuery and creates the ability to make web pages appear to have a similar appearance and functionality to native applications on iOS and Android (jQuery Mobile, 2015). Google maps API is another tool for the proverbial web development toolbox. With the Google Maps JavaScript v3 API, one or multiple Google map tools are available for embedding within the app. This is enables the use of drawing polygons on a map, calculating the area of a field, drawing georeferenced polylines on a map, viewing

field location, etc. (Google, 2014). Many capabilities that come with the Google maps API can easily be manipulated for uses in agriculture.

New application development tools have been created to enable the creation of native applications using web development platforms. Apache Cordova/PhoneGap is an open source API framework that enable app developers to create an application one time using basic web development file format and logic (HTML, CSS, and JavaScript libraries), converting these files into native apps using the targeted platform's (Android, iOS, Blackberry, and Windows) software development kits (SDK) (Cordova, 2015). Being open source, Cordova software is openly available to the public free of charge. Converting a web application into a native app enables the program to run on the device without an internet connection and gives the app greater access to sensors integrated into the mobile device (i.e., GPS based location, accelerometer, Bluetooth communication, etc.) Additionally, native apps can run in the background allowing the app to run without interruption in the event that the mobile device is used for another function (make calls, use of other apps, and internet browsing, etc.). Utilizing this development strategy can streamline production of apps across platforms because they can be created one time and then "converted" or packaged to alternative mobile platforms. Traditionally, apps were developed separately for each platform requiring multiple developers for completion. Using Apache Cordova to convert a web application to native apps gives the app increased functionality and drastically reduces development time.

The use of Bluetooth connectivity is popular with app development due to its ability to connect a mobile device to multiple other sensors for expanding the capabilities and usefulness of the device. Evthings Studio is a tool that that helps developers create

apps for the Internet of Things (IoT) quickly with example files for quickly copying code for connecting to sensors and manipulating data (Evothings Studio, 2015). Additionally Evothings Studio incorporates a workbench for quickly connecting the app under development for debugging and diagnostics. Example apps are available for quickly developing apps in HTML, CSS, and JavaScript for conversion into native applications using Apache Cordova.

2.8.2 Hardware

The Texas Instruments CC2541 sensor tag (Figure 2) is a Bluetooth low-energy (BLE) tag that is capable of connecting to a mobile device with the purpose of transmitting contained sensor data for extending application functionality (Texas Instruments, 2015). The sensor tag includes six sensors inside a rubber shock resistant case. The sensor tag contains an IR temperature sensor, a humidity sensor, a pressure sensor, a 3-axis accelerometer, a 3-axis gyroscope, and a magnetometer. Having these sensors built into a device capable of Bluetooth communication with mobile smart devices creates many opportunities for innovative tasks to be accomplished using this technology. The Texas Instruments CC2541 sensor tag comes as part of a development kit available for the purpose of creating applications that use the tag for seamless integration of the sensor's output with a mobile smart device. Texas Instruments has made libraries available with source code for iOS and Android development.



Figure 2: Texas Instruments CC 2541 Sensor Tag (approximately actual size).

The ISOBlue project is an open source project that began at Purdue University out of the growing necessity to access machinery data the CAN bus on agricultural equipment (ISOBlue, 2013). Access to “raw” data gives farmers more control over their data and increases opportunities and flexibility to use the data within aggregating and analytical tools of their choice. As systems exist currently, use of the data for decision making and record keeping is constrained by software compatibility. The ISOBlue hardware (Figure 3) attaches to the tractor’s controller area network bus (CAN Bus) via a diagnostic port and logs the parameter group number (PGN) as well as the data. The PGN identifies the message content (e.g., start power-take-off (PTO) shaft, activate hydraulic channel one, increase engine RPM, etc.)

The ISOBlue unit can then connect to a user’s mobile android device via Bluetooth connection, and can then transfer this data to the device and/or cloud storage for later use (see Figure 3).

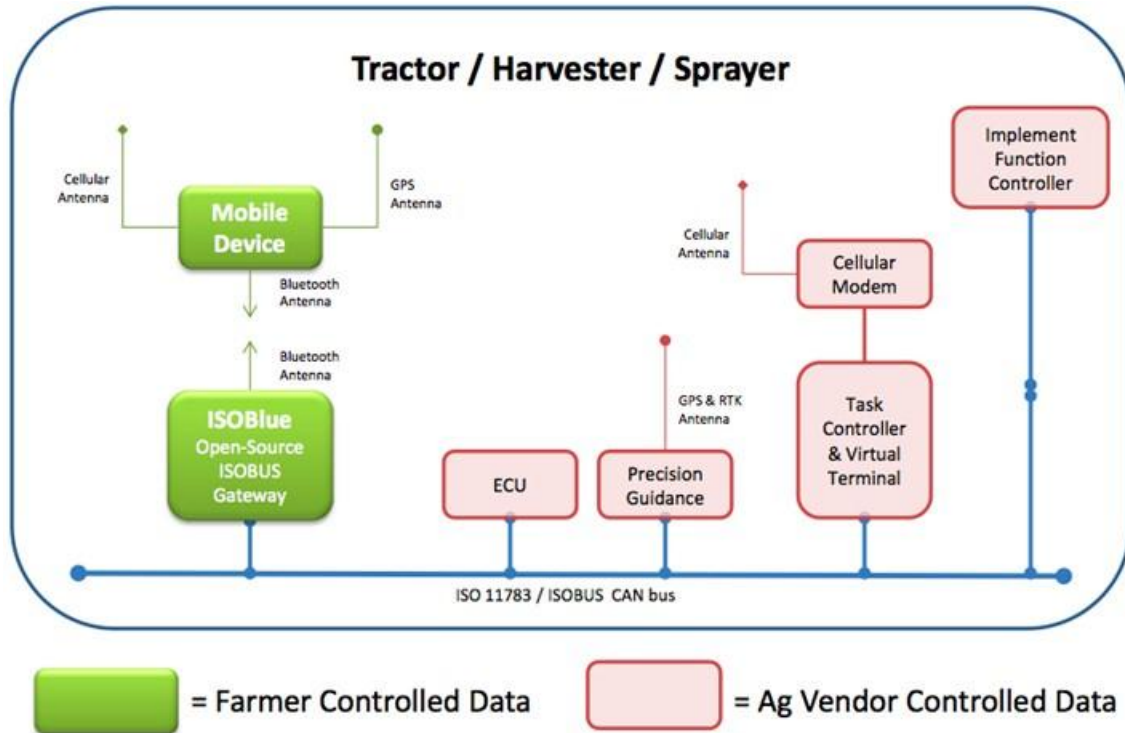


Figure 3: Implementation of ISOBlue with a Tractor's CAN Bus (ISOBlue, 2014)

CHAPTER 3. METHODS

3.1 Functional Specification and Interface Design

The development of the Manure App was conducted in three stages (versions 1.0, 2.0, and 3.0) in order to be more efficient in the creation of the app and to make room for improvement of the app along the way. It was logical to expect changes in the direction of the app as new techniques were made known and new tools become available. Versions 1.0 and 2.0 were not created with the intent of them becoming publicly available tools; they were created for the purpose of achieving each tier of the development process and reassessing the next version's developmental goals. Version 1.0 would be developed purely as a web app hosted on a web server only accessible in a mobile device's browser, while Version 2.0 would be converted into a native application. Version 3.0 was developed to be a publicly distributed product also shared as an open source project. Following this step-by-step development procedure allowed the development to be continuously focused on one outcome making for a more efficient and productive developmental process.

Before the development process began, a functional specification was created. A functional specification helps with the project focus; it identifies specifically what the tool should do for those who will be using it (Spolsky, J., 2000a). A clear definition of the targeted user group(s) ensures that the end product accomplishes the original goal and

functions in a way that the user group(s) find useful. User stories were developed to create different scenarios in which the users will be using the app; this tightens the focus onto individuals that will be using it in their everyday operations. The functional specification also includes a clear definition of the applications objectives for each version developed, keeping the development focused on multiple tiers of development (see Appendix A).

With specific use cases laid out in story form as the functional specification leads logically to mockups which help visualize the usage of the working app. The visuals of the proposed UI aid the development process by enabling the approval of UI before any code has been written. This practice reduces the headaches that occur late in the development process when (sometimes huge amounts of) code would need to be edited in order to change the UI (Spolsky, J., 2000b). Mockups were created using Balsamiq, an online UI mockup generator (Balsamiq Mockups, 2013).

Using Balsamiq, mockups were created displaying a proposed UI. A key aspect of the Manure App design is the “Dashboard” – a main screen of the app that would impart the most important information to the user at a glance (Figure 4). Based on the field (which includes a desired spread rate) and spreader chosen, the dashboard displays a suggested speed of application. From this view, the user can commence data recording by tapping the start spreading icon. The dashboard provides valuable information on the current progress of the spreading task: cumulative number of loads applied to current field, cumulative number of loads removed from source, current fill level of the load being spread, and speed. Keeping the more input heavy tasks required for the

functionality of the app on other pages helps keep the dashboard less cluttered; this helps the operator to stay focused on the task of manure application.

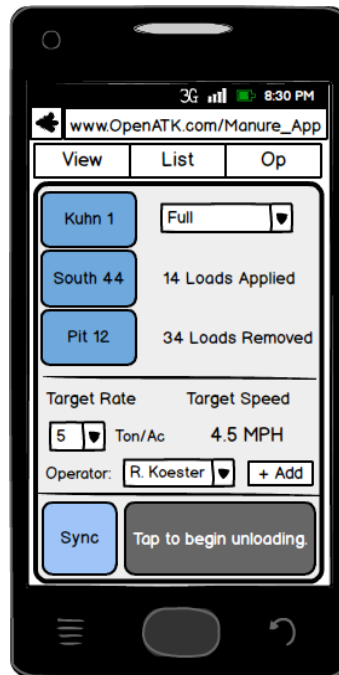


Figure 4: Approved UI Design of the Manure App Main Page/Dashboard.

Additional UI mockups were created to display the task of adding in data objects that will be used for generating manure application records. The Manure App contains multiple pages, each containing a list of objects under each category: field, spreader, source, and operator. Each page will display as many of the elements as the user has created. Figure 5 shows the desired layout of the spreader list and the navigation to adding or editing a spreader object. The mockup shows the ability to add or edit a spreading implement for use in application events. Mockups were not created to display every function of the app, but the ability to add an edit manure sources and operators will be the similar in structure and function.

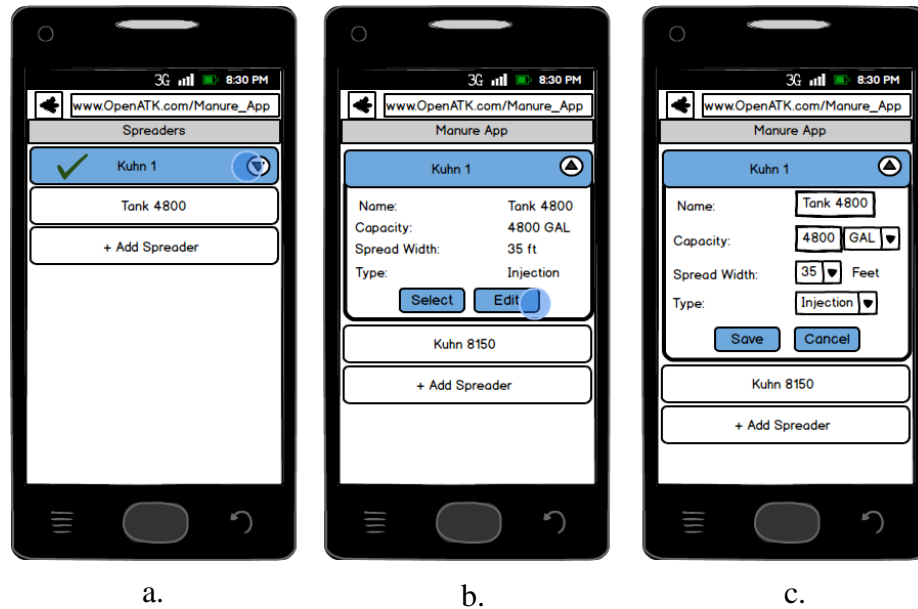


Figure 5: Showing Spreader selection (a), spreader chosen for editing (b), and editing spreader description data (c.).

Later versions of development (versions 2.0 and 3.0), will have the ability to include field boundaries. Mockups were created showing how the user could navigate to the add field boundary tool in the manure app (see Figure 6).

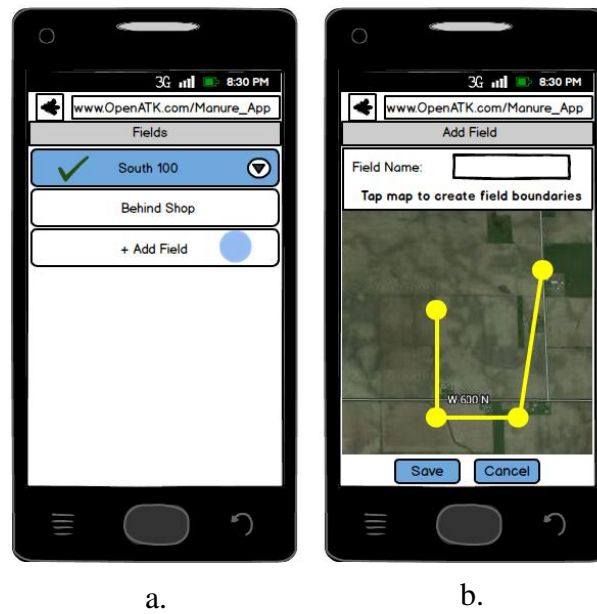


Figure 6: Field selection (a.) and drawing a field boundary (b.).

3.2 Brief Introduction to Web Development

This project uses basic web development techniques. For the sake of being thorough, a brief overview and example code was created for the reader to understand the basic requirements of developing a web application. The overview will cover the different file types that are used and how they are combined to create a web application.

3.2.1 Hyper Text Markup Language (HTML)

The HTML code contains the main foundation of the app's UI in the form of HTML tags (HTML, 2015). These tags are specific elements wrapped inside less than and greater than signs : `<tag element>` which tell the browser the contents of the web page:

```
<html>  
    Contents of the web page/app.  
</html>
```

All elements contained within the opening and closing html tags are what the browser sees and generates into a web page/web app.

The body tag (`<body>`) is the portion of the HTML file that will all become the UI of the app being developed. At the end of the HTML file when the UI has been completely created, the body tag must be closed : `</body>`. Each portion of the UI is split up into divs, indicated by an HTML tag (`<div>`); these are divisions (divs) that enable elements to be displayed in specific locations on the screen (HTML, 2015).

Divs are given an id for later referencing to JavaScript or jQuery functions for UI updating. Within each div, other HTML tags are placed depending on what the divs' purpose will be in the UI. For example, the div containing the icons for “start spreading” and “stop spreading” would look like:

```

<html>
  <body>
    <div id="unload_div">
      <input id="rateInput" type="text" id="rate" value="Pause" />
      <button id="start" onclick="loadComplete()" />
    </div>
  </body>
</html>

```

The above div is placed within the body tag of the html file and has an id of “unload_div”. The div contains two html elements with ids of “rateInput”, and “start”. The input tag needs an accompanying type specified because there are multiple types of defined inputs in HTML. The input type is “text” creating a text box for user input. The button tag tells the browser that it is a button element and contains an “onclick” event. An onclick event gives the button functionality by calling the specified JavaScript function (startUnload) each time the button is pressed.

3.2.2 JavaScript

JavaScript is a popular web development programming language which can modify HTML pages, execute code on events (mouse clicks movements, and keyboard inputs), and send requests to servers without reloading HTML pages (JavaScript, 2015). Since jQuery is a JavaScript library, an example will only cover the use of JavaScript functions due to their similarity. JavaScript, implemented with HTML, gives a web app or web page functionality. The example HTML contains a button that calls a specified JavaScript function. For the sake of this example, the function is related to what the Manure App will do when the “Start Unloading” button is pressed. The JavaScript functions can be created in the HTML file as long as the code is wrapped within HTML

tags and is identified as script tags (<script>JavaScript Code</script>). An example JavaScript function created in an HTML file would look like this:

```
<script>
    function startUnload () {
        gathers user data
        records gps location
        places all data in browser storage for later queries
    };
</script>
```

The example code is created by first stating that it is a function, the following variable (startUnload) is the name of the function followed immediately by an opening and closing parenthesis: (). This has created a function that can be called in whatever manner that the developer desires. Directly after the creation of the function, there are opening and closing braces: {contained code}, any JavaScript code contained within these braces will be returned when the function is called.

JavaScript arrays are another key aspect of web development. Arrays allow web browser to order variables inside brackets that can be added to other arrays or queried for specific data. JavaScript Arrays are created by defining a variable and setting it equal to a set of open and closing brackets.

```
var array = [];
```

JavaScript objects that have been created using user input data can be stored inside arrays for use in the apps functionality or stored in the browser's local storage. An example of an array storing specific spreader data in the Manure App would be created in this manner:

```
var spreaders = [{"Name": Kuhn1, "Capacity": 20, "Unit": tons, "SpreadWidth": 30}]
```

The spreaders array contains an object of user created spreader data. Formatting the data in this manner helps keep the data in order and simplifies access. If the user added another spreader to the Manure App, the the array would then contain two objects.

```
var spreaders = [{“Name”: Kuhn1, “Capacity”: 20, “Unit”: tons, “SpreadWidth”: 30},
                {“Name”: Balzer, “Capacity”: 4800, “Unit”: gallons, “SpreadWidth”: 30}]
```

To return a specific value from the array and the order of which the values have been added is known, the returned value of “Kuhn 1” could be returned by the following code:

```
var first spreader = spreaders[0].Name;
```

spreaders[0].Name refers to the first object in the array (count starting at 0) and is followed by the variable of interest (Name) to identify the object to be returned. If the order of the array is not known, a “for loop” must be used. For loops are used in JavaScript to query through arrays to return an object containing a value. For example, if an entire JavaScript value needs to be returned for adding to the records array and the only value that is known is the name of the object (Kuhn1), the array can be queried using a for loop

```
for (var i = 0; i < spreaders.length; i++) {
    if (spreaders[i].name === “Kuhn 1”){
        var returned spreader = spreaders[i];
        break;
    };
};
```

The above code returns the entire object that contains “Kuhn 1” referred to in the array by “Name”. The returned object would look like:

```
{“Name”: Kuhn1, “Capacity”: 20, “Unit”: tons, “SpreadWidth”: 30}
```

This overview illustrates the JavaScript functions and variables used in development; the JavaScript adds functionality to web development projects.

3.2.3 Cascading Style Sheets (CSS)

CSS is a language that enables the editing of the UI appearance that has been laid out in the HTML file (CSS, 2015). CSS gives the developer the ability to: select the color of the web page/web app and all contained elements within the HTML file, change the aspects of the layout, and enhance the style and size of text font. There are many ways that CSS can be used to customize specific elements of the UI. CSS consists of selectors, properties and values.

Selectors point to a specific element in the HTML file. By using the selector of a specific div, like the one created in the HTML file example; CSS can be used to customize the div. Immediately following the selector is an opening and closing braces (similar to a JavaScript function) that contains the styling properties to be imposed on the selected HTML element (see CSS Example). Each property is given a value as a measure of what level or selection of the desired outcome of each property. There are many properties that can be used for customizing HTML with CSS. For the sake of brevity, an example was created but the only a few methods of customization were included.

```
#unload_div{
    width: 100px;
    height: 100px;
    background-color: red;
    margin: 10px;
}
```

The CSS code in this example shows that the selected element to be customized is the `unload_div` (created in the HTML example). The properties that will be customized are

the width of the div, the height of the div, the background color, and a margin surrounding the div (dimensions are pixels).

There are many more HTML, Javascript, and CSS code types and functions involved with web pages and applications, but these examples provide an introduction to the purpose of each file and language type which combine to accomplish the desired outcome in web and app development. The final version of the code developed for this project is available for download at: <https://github.com/OpenATK>.

3.3 Development of version 1.0

The goals for version 1.0 of the manure app was to create a functioning web application capable of recording manure hauling events with the user tapping the start/stop status of the manure app to begin the data recording process and the specific data that goes along with each load applied including: time and date, amount applied, and operator conducting the operation. Since this version is a web application, the goal for data exporting was to enable the app to download the stored data files as a CSV file directly into the user's device download folder. Records acquired this way can be exported easily by email or connecting the device to a computer for spreadsheet viewing and utilization.

The Manure App was developed using the most common web development techniques including: HTML5, CSS3, and JavaScript as outlined in previous sections. Additionally, jQuery was used to help minimize total code required for complete functionality of the app. jQuery Mobile has built in styling for the rapid creation of functioning web applications; its use greatly reduced the complexity and duration of

development because it can quickly generate an app like UI. Any changes needed for the forms, inputs, and buttons, were edited after the fact in the CSS file of the project folder.

The app was hosted on GitHub pages (GitHub, 2015). GitHub is a code development collaboration tool that allows teams of developers to place code in one location while the developers edit the project. Additionally, GitHub allows users to host a project as a webpage.

3.3.1 UI Development

The UI of the main screen or Dashboard was created by setting up divs with specific sizes and location to be generated, displaying data fashioned after the mockups created prior to development. jQuery Mobile was used to create a web page with appearance and functionality of a mobile app. Once jQuery mobile was loaded into the HTML file, the inputs and buttons took on the styling created by the jQuery mobile framework. This sped up the development process and reduced the workload required for the styling of the web app.

The Dashboard was created by placing button elements in the previously created and customized divs (see Figure 8 a.). The Dashboard is split in two horizontally, making the selecting of the spreading objects in the top half of the Dashboard; and buttons controlling the spreading action located at the bottom. Within the object selection div, button elements were created with the ability to navigate to other pages. When tapping the spreader icon, the app will take the user to the spreader page where the user can select the spreader in use, create a spreader implement, and edit existing spreaders. The same process will occur for the other user selected data (field, source, and operator).

The bottom of the Dashboard is the location of the icons that the user will press to begin and end the manure application data collection process, creating a manure application record. The icon was created by inserting a button tag in the HTML file calling the JavaScript function, startUnloading. Once the startUnloading function has been called, the button is updated to become the ‘Load Complete’ button. By tapping the ‘Load Complete’ button, the data of all currently selected object data is stored into a JavaScript object becoming a new manure application record accessible from the records array. The JavaScript that handles the collection and placement of user input will be discussed further in the next section.

There are key pieces of data (input) required for the Manure app to create complete and informative manure hauling records: spreader characteristics, field name and description, manure source, and operator name (see Figure 8 b.). Each object must be defined and specific data must be stored in the app for later calculations. Adding a spreading implement, for example, requires the user to first add specific data defining the following characteristics: capacity, spread width, and type (right, left, rear discharge, or injection). These pieces of information are an important aspect of a spreading operation that should be stored in the record space; but also important for accurate calculations for proper application rates.

Separate pages were generated in the app by creating multiple divs containing text and numerical input elements for collecting data pertaining to specific user defined objects. Once again, using the spreader input as an example, an HTML div was created in the HTML file containing the following input tags:

```
<div id = "SpreaderInput">
```

```
<input id = "sprName" type = "text" />  
<input id = "sprCap" type = "number" />  
<input id = "sprWidth" type = "number" />  
<input id = "sprUITime" type = "number">  
<button id = "sprSave" onclick = "saveSpread()">Save Spreader</button>  
</div>
```

The div containing the inputs are generated in the spreader list page where the user can input the data pertaining to their spreading implement. In addition to the text input elements, a button element with an id of "sprSave" was created to handle the calling of the JavaScript function for saving the data input of the user; this is explained in further detail later.

Similar to adding a spreader, adding a field and its characteristics is required for the functionality of the app. Information required for later use in the app would be: field name, field boundary, field area, and desired application rate for the specified field. For version 1.0 of the Manure App, the field information used is limited to field name and desired application rate. The field list in the Manure app is a separate page where the added fields will be located in a tap-selectable table. This shows the field characteristics of each field and allows the user to edit, add, and delete fields from this listing.

The source listing of the Manure App is a separate page of the app that allows the user to add, delete, and edit manure sources. The source information collected includes: the source name and the nitrogen, phosphorus, and potassium levels in manure from that source. This information is necessary for the calculation of total nutrients applied to the field for regulatory compliance and proper nutrient management. This information is most accurately derived from testing, but, as noted earlier, can be can be estimated using

benchmark values (Moore and Gamroth, 1993). With the source information, the total amount of nutrients applied can be calculated on a per-load basis.

The operator information stored is limited to the operator's name only, and is stored for the purpose of keeping management records of who completed each task for the operation.

Once the data pertaining to each selectable object (spreader, field, source, and operator) has been created, the app displays the information so the user can select by tapping the object that is being used. HTML tables were used to display the data for selecting by the user with one object per row (see Figure 8 c.). In order to select an object in use, the user can tap the applicable table row. In order to alleviate confusion as to whether or not the desired object is selected, the table row that is currently selected changes color to indicate the object currently selected.

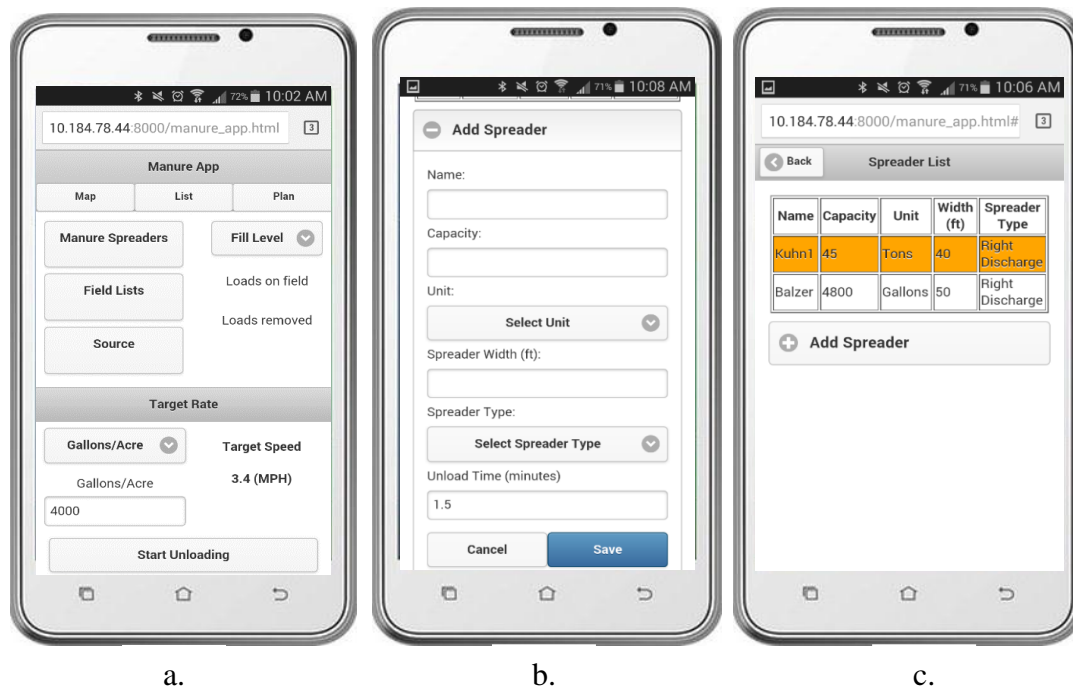
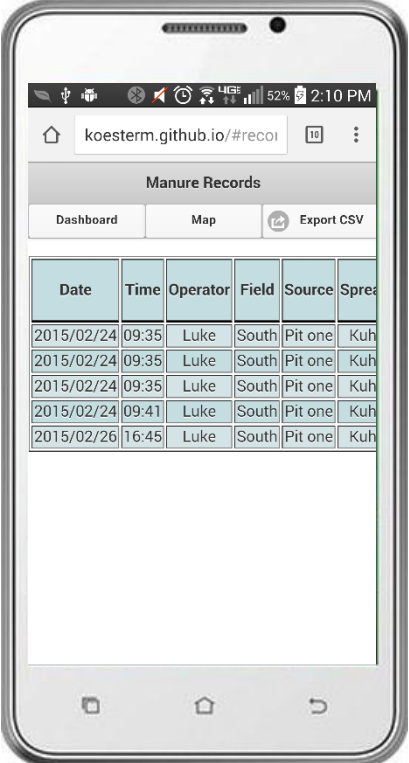


Figure 8: Version 1.0 functioning UI showing dashboard (a.), add spreader form (b.), and generated spreader table (c.).

Once all required selections have occurred, users can generate manure hauling records by tapping the “Start Unloading” followed by “Load Complete” to generate a new manure hauling record. Similar to the development of the object lists, a record page was created for displaying the manure hauling records, and additionally, an export icon for exporting the data from the mobile device (see Figure 9).



Date	Time	Operator	Field	Source	Spread
2015/02/24	09:35	Luke	South	Pit one	Kuh
2015/02/24	09:35	Luke	South	Pit one	Kuh
2015/02/24	09:35	Luke	South	Pit one	Kuh
2015/02/24	09:41	Luke	South	Pit one	Kuh
2015/02/26	16:45	Luke	South	Pit one	Kuh

Figure 9: The record page of the manure app displaying records in a table format (total record data not shown)

By pressing the Export CSV button on the record page of the manure app, the user creates a copy of their records from the app’s stored data, converts this data into a CSV file, and saves it into the device’s download folder.

3.3.2 Implementation of UI functionality

Once the text and numerical elements have been filled in by the user, JavaScript functions must be written for the handling and storage formatting of the user data. JavaScript arrays were created for the storage and querying of object data input by the user. A total of five arrays were created to hold each type of created object: spreader, field, source, operator, and records. This was done in order to save all objects created by the user and for the sake of querying this data when the user creates another manure application record. The save button created in the div (spreader input div example code), calls a JavaScript function that takes the values of user input data and stores these values into specific arrays for later use.

The arrays holding the user input making up the specific selectable objects were displayed in HTML tables for viewing and selecting by the user. One table row was created for each object in the array. A click listener was implemented into the table enabling the use of tap events for selecting specific objects being used in spreading operations. For example, the user wants to select the Kuhn 1 for today's manure hauling operations. The click listener returns the number pertaining to the row of the table being selected. Additional code was created that would return the value of the first cell of the table (the name of the spreading implement). Once the name of the spreader was defined, a "for loop" was used to return an object in the spreader array with the name "Kuhn 1". Similar code was created for generating the tables to display selectable objects on their respective page.

The final step in the process of creating manure application records is the ability to export the data for use in the operation and reporting. A button was placed in the div

creating the page for displaying manure records. A JavaScript function was created for looping through the records array creating a CSV file. The function queried through the records array for specific data that will be displayed in the CSV file. A shortened version of the information taken is shown below as a JavaScript object.

```
var csvArray = { "Date":record.date, "Time":record.Time,"Operator":  
    record.operator,"SourceName":source.name,"N":source.N,"P":source.P,"K":  
    source.K,"Nutrient Measure":source.nutrientUnit,"Spreader": spread.name,  
    "Spreader Capacity":spread.capacity, "Load Fill Level":spread.fillLevel,  
    "Field": field.name, "Rate": record.field.rate };
```

Once the data has been extracted from the record array, it is converted into a CSV array by taking each value in the array and separating one from another with a comma. A download function is then called, downloading the data into the download folder of the device and can then be exported or edited in whatever manner the user desires.

A storage service was needed in order to store data that has been input into the app. Without the use a storage service, each time the web app was closed or the page was refreshed, the previously input data would be eliminated. This would be unacceptable to require fresh inputs each time the app was opened. To alleviate this occurrence, browser local storage was used to store the data created by the user, then queried each time the app is loaded to load all the data that had previously input by the user. Every time an array was updated with additional information, the array was converted to a JavaScript object and written into the browser's local storage. The browser local storage is only accessible by the web page that created it. The data is not accessible or stored in accessible files within the user's device, in order to access the data created by the app and stored in the browser, a download function must be called.

3.4 Development of Version 2.0

The goals of the version 2.0 of the Manure App was to improve the UI, enable the use of the mobile device's GPS for location and speed data, implement the Google Maps API for placing maps within the app for creating field boundaries and displaying recorded GPS points making up the application path, and convert the Manure App into a native app using Cordova/PhoneGap. Version 2.0 uses the same data structure for each recorded load data that was developed in Version 1.0, but adds the recorded GPS path to each load record for later drawing on the Google map. The stored data location in the browser local storage contains each load record as a JavaScript object in an array. The Google Maps API can then query through each load object, finding and displaying each polyline as a georeferenced representation of the load application path.

3.4.1 User Interface Improvement

Development for version 2.0 began with the same capabilities that had been developed in Manure App version 1.0 with the additional improvements of the UI and the insertion of Google Maps Web API v 3.0 for enabling the use of Google maps within the Manure App. The UI improvements were small edits to streamline user interactions and data input. The changes made to the UI only involved the relocation of data entry forms from one page to another and editing the size of divs to better fit the screen.

The dashboard of the app was shortened to eliminate the need for scrolling (see Figure 10). This was accomplished in CSS by passing a value to the height parameter creating the UI of the dashboard. The ability to adjust the desired rate of application was removed from the dashboard and placed in the field edit page; this seemed most fitting since nutrient management plans are generally made by field and this also improved the

look of the dashboard. This was done by removing the HTML input tag from the div in the UI to the div containing the field input tags. The dashboard and other navigation buttons were enlarged for better use of the screen area and improved ease of use when operating machinery. Once again, the use of CSS was implemented for the customization of the button size. The height parameter was given a larger value in the CSS selecting the specific button ids that a larger height was desired.



Figure 10: Version 1.0 dashboard (a.) and version 2.0 dashboard (b.).

Tables generated for tap selection of desired field, operator, source, and spreading implement were modified for making tap selection of these objects more intuitive (see Figure 11). When a user would see the table displaying selectable objects for the first time in version 1.0, it would not be clear that tapping on a row of the table would select

that specific object (spreader, field, source, or operator). To accomplish this, the JavaScript function that created the table was edited, setting the vertical cell boundaries to zero in the app's JavaScript file (code is available at: <https://github.com/OpenATK>). This removed the vertical cells of the table making each object appear to be a selectable object rather than a table to display information.

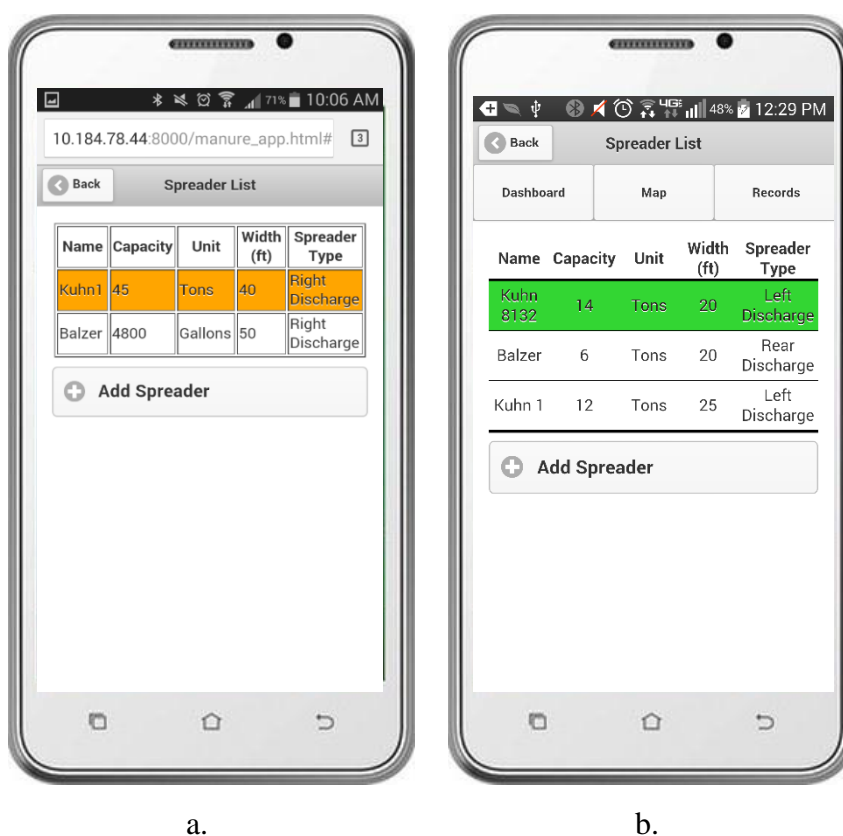


Figure 11: Updated UI tables for selecting the spreading implement Version 1.0 (a.) and Version 2.0 (b.).

An overlay and animation was added to make the user aware when spreading operations are occurring; this helps to reduce the probability that a user will navigate to other pages of the app while spreading (to de or re-select the current spreader or cause other errors). The overlay is a partially transparent layer that removes the ability to tap on

any button except the “Load Complete” and “Pause” buttons. The Overlay was generated by creating another div in the HTML file with an id of “overlay”. In the CSS file, the overlay div was customized to fit exactly over the top portion of the Dashboard by adjusting the height and width values. The div is set to be hidden until the “startUnload” function has been called. Additionally, the CSS parameter called the “z-index”, a parameter that controls the order of stacked HTML elements, was adjusted to generate the div in front of the Dashboard page (see Figure 12).

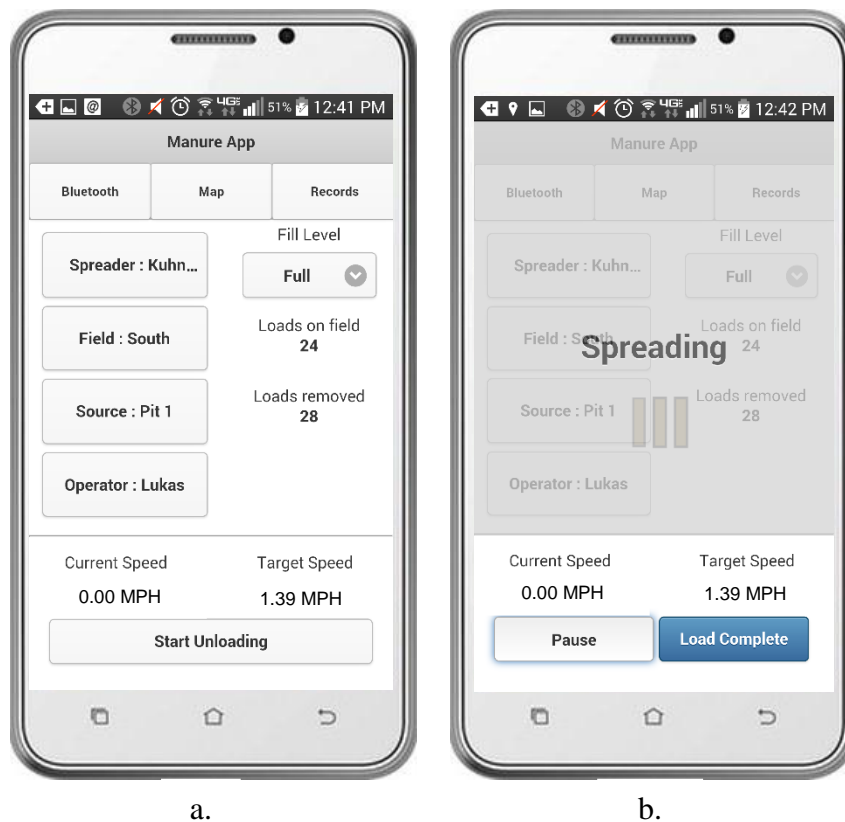


Figure 12: Dashboard not spreading (a.) and during spreading event (b.).

The Google Maps API was implemented into the app to enable use of geolocation tools for calculating travel distance, and area of selected boundaries. (Google, 2015). Two additional pages were created in the HTML file that generate two separate maps

using the Google Maps API, the first map was implemented giving the user the ability to draw, edit, or delete field boundaries (see Figure 13 a.). Figure 13 (b.) shows the second map created giving the user a view of the overall progress of manure application and displaying: all fields and their boundaries listed in the app, info windows displaying field information when the field polygon has been tapped, and polylines that accurately display the spread path of each individual load of manure hauled on the field.



Figure 13: Created map pages: field boundary editor (a.), and operation map page (b.).

3.4.2 Implementation of Google Maps API

Utilizing multiple APIs made it possible to create records from data that has been georeferenced to specific locations on the earth, e.g., field location, field boundary, load spread-path polyline, etc. Accessing the user's mobile devices GPS data was accomplished by using the HTML5 geolocation API.

The Google Maps API was implemented by loading the source file into the HTML file. JavaScript code was then written that handled the loading of the maps and the generation of the maps in specific pages created in the HTML file. Adding a field boundary is accomplished by navigating to the field list of the app and tapping the "Add New Field" button at the bottom of the list, opening a form for field data input. Once the user has selected the field name, desired rate, and unit of manure application (tons/acre, gallons/acre) the "Add field boundary" button can be tapped. This brings up a Google map where the new field boundary can be drawn. The Google Maps API has built in functionality for: placing points, drawing polylines, and drawing polygon objects by tapping on the map. The app use this API to enable the drawing and editing of field boundaries for saving in the field array for later calculations. Polygons are created by tapping the map, placing points with each tap. To close a field boundary polygon, the user taps back to the first point of the boundary or double taps the map, closing the polygon and creating a new field polygon object. A "Done" button (see Figure 13 a.) was added to the HTML file for the purpose of navigating away from the map page when the new field object is ready to be saved. On the map pressing the "Done" button on the field boundary map, navigates back to the New Field form, where the user can save the new field (see Figure 14 a.).

When the new field has been saved, the function that creates the field list is called to recreate the table on the page, updating the page to show the list of fields with the addition of the new field that has been created. The newly added field can now be selected as the current field for manure application and edited (see Figure 14 b.).

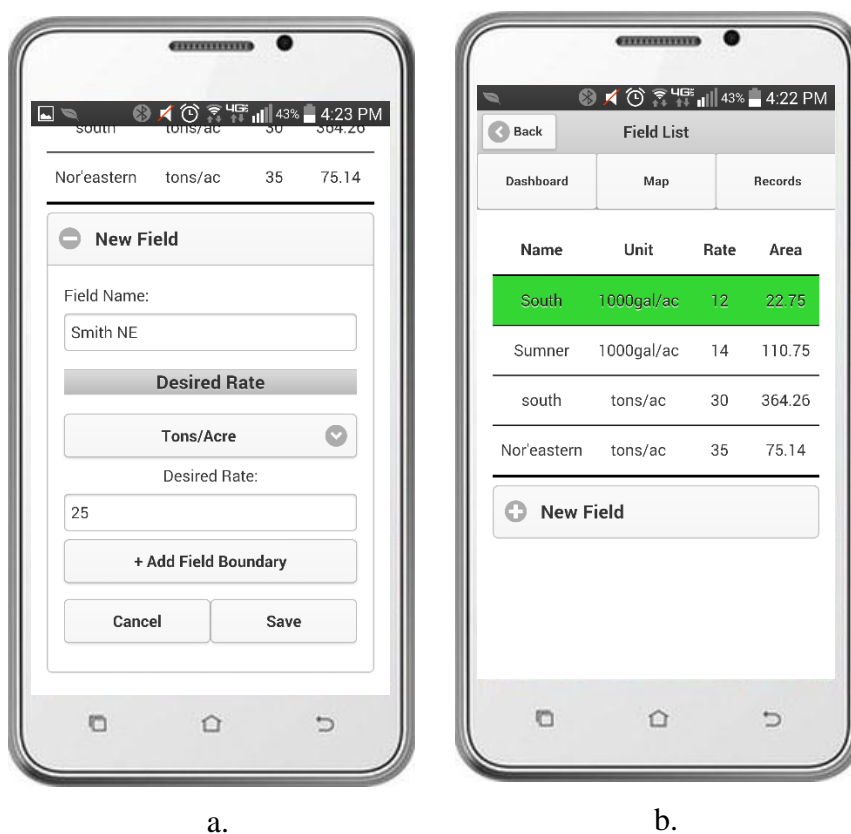


Figure 14: Interface of field data entry form (a.) and field list page (b.).

The HTML5 geolocation API is accessible from the HTML file and can be used without the loading in additional resources. The API enables the use of the mobile device's GPS sensor data (W3C, 2014). The API can be used for one-time requesting user location, as well as, continuously updating location data. The user location can be requested with the use of the following function (W3C, 2014).

```
function getUserLocation(){  
  navigator.geolocation.getCurrentPosition(successCallback,errorCallback,  
    {maximumAge:3000});  
};
```

The function makes a request to the GPS sensor for the user's position. The function has a success and error callback for indicating the status of the request. The use of the "maximumAge" option indicates that the position is no more than three seconds old before requesting an updated position. By giving the web application access GPS data from the mobile device, the app records the position of the spreader at three second intervals as the operator drives through the field during manure application. The GPS data generated by the mobile device includes the error of the device in meters. When the device first begins to record GPS data, the error is, in some instances 20 meters (65.6 ft.). The JavaScript function that records the points was restricted to adding points to the array that contain an error less than 10 meters (32.8 ft) to help reduce mapping errors. The GPS data points are continually collected and stored in a temporary array during the spreading operation, then stored with the newly formed spread record when the load is complete. A convenient feature of mobile device technology is that GPS data can be collected without WIFI or data connection. The maps may not be shown properly during field operations without internet connection but the GPS points will be collected. The spread paths can then be displayed properly when the user has an internet connection. A function was written to query through the stored record array for spread path GPS data; this data is then handed to the Google Map, displaying all recorded spread paths (see Figure 13 b.). The length of the spread path is then calculated using a Google Maps API function. Once the length of the spread path is returned, the average rate of the manure application is

calculated based on the total material applied and the width of the spread path; it is also saved to the record array.

Much of the JavaScript Code used for the implementation of the Google Maps and editing its use is fairly extensive, for this reason, the code used can viewed at: <https://github.com/OpenATK>.

3.4.3 Conversion to Native applications

As a web app on GitHub, the app required the browser to be active (on top) during manure hauling events. However, if the user was to use his/her mobile device for any other purpose (phone calls, web surfing, playing music, taking pictures or anything which halted browser activity), the Manure App would cease to function. Having the app run in this fashion is not acceptable. Conversion to a native application using Apache Cordova solves these problems (Cordova, 2015). By downloading the Cordova software, and the targeted platforms (in this case, Android) SDK; the app's JavaScript, CSS, and HTML files were passed into a created Cordova project file for assembly into a native running android app (see Figure 15). Having the Manure App converted to a native application on the Android platform enabled the app to run while offline and function concurrent to other activities conducted on their mobile user's mobile device.

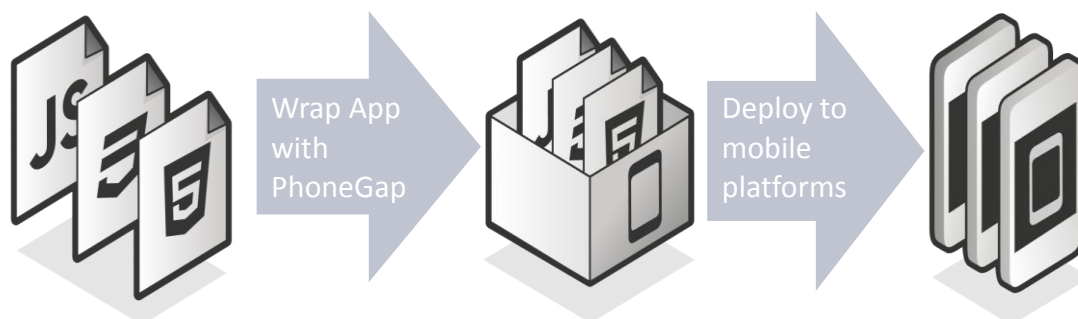


Figure 15: Showing the required files to convert into a native app using Cordova (PhoneGap, 2014)

3.5 Development Version 3.0

Version 3.0 was intended for public release. In early versions, the user controlled the spreading status by tapping the “Start Unloading” and “Load Complete” button for recording each manure hauling event. By using an external Bluetooth identification tag with integrated accelerometers, these required tap events can be eliminated. The spreader and spreader status (on/off) could be determined without user intervention. Additionally, version 3.0 implemented the use of an online database for multi-device synchronization, and the use of geofencing APIs for field application selection without the need for the user to select the field each time application was occurring. 3.0, although autogenic, still allows the user to tap for certain input and status changes (select a field, start, pause, stop).

3.5.1 Spreader Identification and Status

To truly accomplish autogenic capabilities, the Manure App must be able to derive the spreader characteristics of the spreading implement in use, as well as the field in which spreading operations are occurring. Originally the use of an ISOBlue unit which listens for specific PGN messages on the CAN bus was considered for monitoring PTO

status. To minimize cost and improve functionality by identifying which spreader is used as well as its operating status, the use of a TI CC 2541 sensor tag on the manure spreader was pursued and implemented. The Texas Instrument CC2541 sensor tag contains six sensors that include: IR temperature sensor, humidity sensor, pressure sensor, accelerometer, gyroscope, and magnetometer (Texas Instruments, 2014). The contained sensors that have the most relevance for this project were the accelerometer, and/or the gyroscope. The accelerometer and gyroscope sensors, detect and transmit directional acceleration and rotation on three (x, y, and z) axes.

Connection between the sensor tag and the mobile device running the app was achieved using the Bluetooth connection capabilities of the mobile device and the sensor tag. Evthings Studio has created an extensive library of example apps (including example code files) ready to connect to multiple Bluetooth sensors that are currently available on the market (Evthings Studio, 2015). Combining a few of the example apps code files with some additional customization, the app was able to connect to the TI CC2541 sensor tag with minimal development time and read the data being transmitted from both sensors.

Each sensor tag has a Bluetooth address that is unique to each tag. Manure spreaders that have been named within the Manure App can be referenced to each specific sensor tag. When the app connects to a sensor tag for the first time, it asks the user which created spreader it would like to be linked to. Once the user has selected the spreader it would like, the unique address is stored inside the selected spreader object contained within the stored spreader array. The next time the app connects to the sensor tag, the app queries through the spreader database for an address matching the address of

the sensor currently connected. Once the address is found, the app selects the spreading implement containing the Bluetooth address as the spreader being used. Multiple tags can now be utilized on an operation installed on each spreading implement. Once the device connects to one of the Bluetooth tags, the app selects the spreading implement that is being used by the operator. In this manner, the need for the user to select the implement that is being used for application can be removed.

In order to remove the requirement of the user tapping the start and stop buttons in the app for each load applied, the Manure App needs to know the spreading status of the implement. By connecting and reading the transmitted data from the Bluetooth sensor tag installed on a shaft which only turns during spreading, the accelerometer and the gyroscope returned values will read a significant change indicating a manure application status change. The sensor acts as a trigger to accurately start the recording of GPS location when manure application is taking place and completes the record generation when the load is complete.

By analyzing these acceleration patterns, algorithm development for field operation status can be created. Testing on the sensor is required to determine a base acceleration or rotational value for indicating the spreading status. Testing was conducted and described in full detail (see 3.6 Sensor Testing and Threshold Development). The gyroscope sensor on the Bluetooth tag returns the rate of rotation imposed on the tag on three (x, y, and z) axis. However, the gyroscope sensor is limited to 250 degrees per second (~41 RPM), and delivers a relatively noisy signal under vibration. There are many variations in manure spreader designs, but the use of either the accelerometer and/or the gyroscope should be sufficient for accurate spreading status communication with the

Manure App; in some cases, the use of both sensors may be required. Section 3.6 covers this topic further.

3.5.2 Multi-Device Synchronization

The data storage utilized in previous versions of the app were beneficial for the development process and worked well for single device use. However, the manure records “problem” requires the app to sync data across multiple devices (carried by different operators on a farm) when the app is implemented in an actual farming operations. One major change this requires is the switch from the use of browser local storage to using an in-browser local storage software that can sync to a web based database. This was accomplished using the PouchDB local storage software. PouchDB is an open source JavaScript local database that runs in the browser and can sync easily with Apache’s CouchDB compatible databases for easy synchronization across devices (PouchDB, 2015). CouchDB is a database that stores information in JavaScript Object Notation (JSON) (CouchDB, 2015). JSON data is human readable (therefore parsable) and easy to access and modify (JSON, 2015). CouchDB enables developers to create their own database that is running on a server for data storage. Using CouchDB as a database would not fit the capabilities of the targeted users for this project. Instead, there are multiple online databases that are CouchDB and PouchDB compatible that only require the creation of a user profile to be used. Additionally the use of these online databases (to the extent needed for manure records) is free.

PouchDB works similar to the local browser storage approach used in the previous versions but makes the data input simpler and handles the data writing and

reading processes, making the functionality less code heavy. PouchDB is a document oriented database, meaning that each object written to the database is stored in the database as a document and is stored with an id and revision number. The id enables id based queries while the revision number helps with syncing. The revision number is created based on the time the document was created. Knowing when the document was created lets PouchDB know when revisions occur, displaying the latest version of document. Additionally, PouchDB handles the occurrences of offline storage. When a mobile device using PouchDB is being used offline, the local database on the device stores the data until the device has internet access again. Once an internet connection is established, the local database syncs to the remote database. All devices that are connected to the remote database are then synced to show the updated data.

Creating a local database is the first step in creating a syncing database. The code creating a local database using PouchDB is shown below.

```
var spreader_db = new PouchDB('spreaders');
```

This code defines a new database called “spreaders” and can now store objects. Data is stored in the database by calling a function that writes a new document. Code showing a function writing to the database is shown below.

```
function pushSpreaderToDb(){
  var dbSpreader = {
    _id: new Date().toISOString(),
    obj: newSpreader
  };
  source_db.put(dbSpreader,function callback(err, response){
    if(!err){
    }
  });
}
```

The code creates a variable called “dbSpreader” containing an id of the date and time that it was created. The object, defined by “obj”, refers to a newly created spreader object.

The code is run and all the above information is written to the database.

Separate databases were created for data source generated by the app: spreader, source, field, operator, and records. At this point the data has been saved and functions the same way it did in version 1.0 and 2.0 with data being stored on the local device’s browser’s local storage.

Having a local database up and running is the first step toward synchronized records; the second step is to create and define a remote database within the app’s JavaScript code. IBM’s Cloudant online database was used as the remote database for this project due to its compatibility with PouchDB and the fact that it is free to use, provided that the project does not exceed the maximum data usage per month (Cloudant, 2014). Pricing for usage is calculated in multiple ways. Total storage (\$1/gigabyte/month), number of heavy data calls (\$0.015/ 100 calls/month), and number of light data calls (\$0.015/500 calls/month). These prices combined create the monthly charge. If the user stays under \$50 per month, there is no charge and usage of the database is free. Heavy data calls to the database include: data POSTS/PUTS (writing to the database), EDITS (editing documents), and DELETES (deleting documents). Light data calls are GETS (retrieving documents from the database) (Cloudant, 2014). Each record generated is approximately 1 kilobyte, the storage limit per month wouldn’t be exceeded until 50 million records were stored. During development there were tens of thousands of heavy data calls to Cloudant database each week. Even then the largest bill achieved was \$6.39. It is reasonable to believe that Cloudant is a viable database solution

for targeted users without ever being charged for the service. To use Cloudant as the remote database, a subscription must be completed. Once the subscription is completed, Cloudant sends an email containing the Uniform Resource Locator URL to the database for use in syncing. In order to enable the PouchDB to sync to the remote database, the remote database URL must be defined. Additionally, a sync method is passed into the code telling the local database to sync with the remote database. The code that defines the URL to remote database and method for syncing are as follows (PouchDB, 2015).

```
var remoteDatabase = 'http://url to remote database/spreaders';  
spreader_db.sync(remoteDatabase);
```

Once the PouchDB and the Cloudant databases have been set up, the Cloudant database URL accompanied with the username and password (created during the subscription process) for the Cloudant database, was written into the PouchDB database file in the Manure App. From this point PouchDB handles all data replication to and from the Cloudant database. Having the PouchDB and the Cloudant database connected in this manner, enables the Manure App to be used on multiple devices. The data created by the app is uploaded and synced with the Cloudant database, the Cloudant database then replicates the data back down to all other devices connected to the database in the order it was created.

Using Cloudant as a remote database has additional advantages to being able to sync data across multiple devices while being secure. Using Cloudant enables the user to edit data outside of the Manure App. The user can log into Cloudant's website (using credentials created during establishment of the database) and access any piece of data that has been stored. The data is saved to the Cloudant database in five different databases:

records, fields, sources, operators, and spreaders. Separate databases for each specific data type were implemented to simplify data queries and to simplify the search for specific data for future users. Once the user is logged in to their Cloudant account, the Cloudant dashboard displays all of the user's created databases (see Figure 16). The user can then select individual databases to view their contents.

Name	Size	# of Docs	Update Seq	Actions
fields	0.8 KB	4	4	[Replicate] [Delete] [Lock]
operators	293 bytes	7	1	[Replicate] [Delete] [Lock]
records	2.6 KB	2 ⓘ	4	[Replicate] [Delete] [Lock]
sources	382 bytes	2	4	[Replicate] [Delete] [Lock]
spreaders	0.8 KB	3 ⓘ	4	[Replicate] [Delete] [Lock]

Figure 16: Cloudant database UI displaying all databases created by the Manure App.

Selecting one of the databases displays all of the objects stored as editable documents (see Figure 17). The user can view stored objects in each database and modify, delete, or replicate specific pieces of data. Whole objects can be deleted from the

database by selecting the document and selecting the delete button (but much care is required since syntax is critical to database operations).

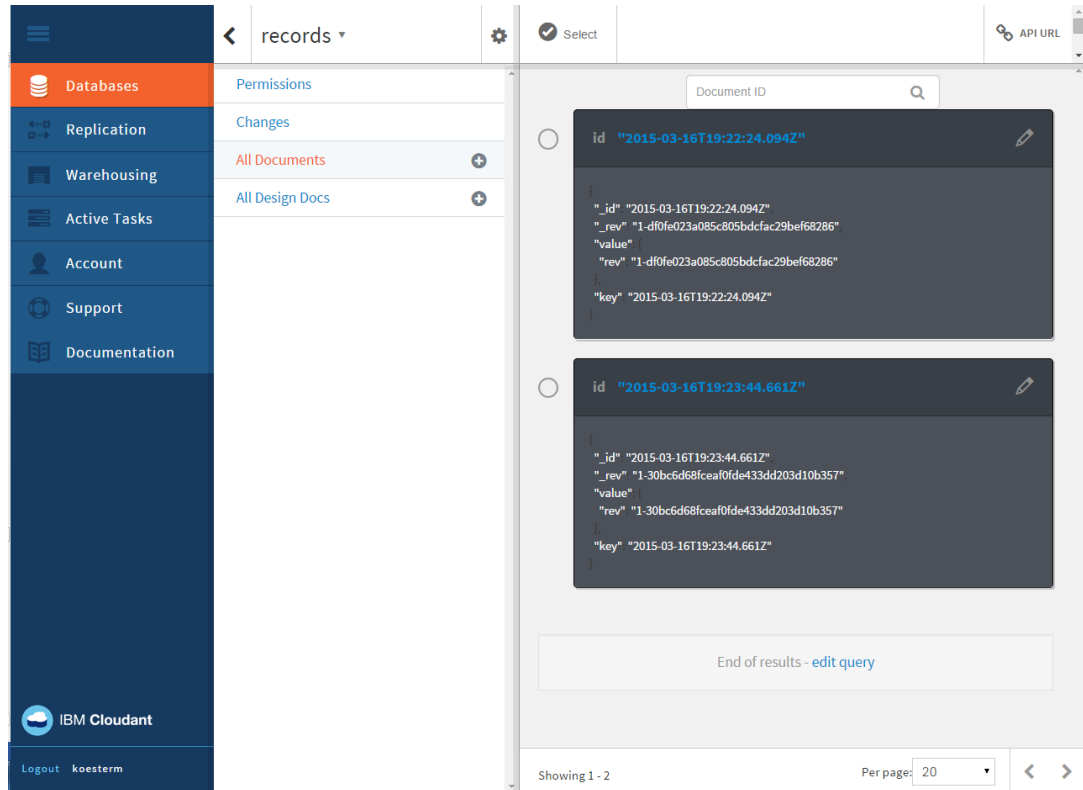


Figure 17: Cloudant database showing the record objects contained within the records database.

Once the user has navigated to the desired database to view stored data, the user can select the edit button at the top right corner of the document to open the stored JSON document as a text file. Figure 18 displays the view the JSON data stored for a manure application record. Users are not restricted to the use of Cloudant as their remote database.

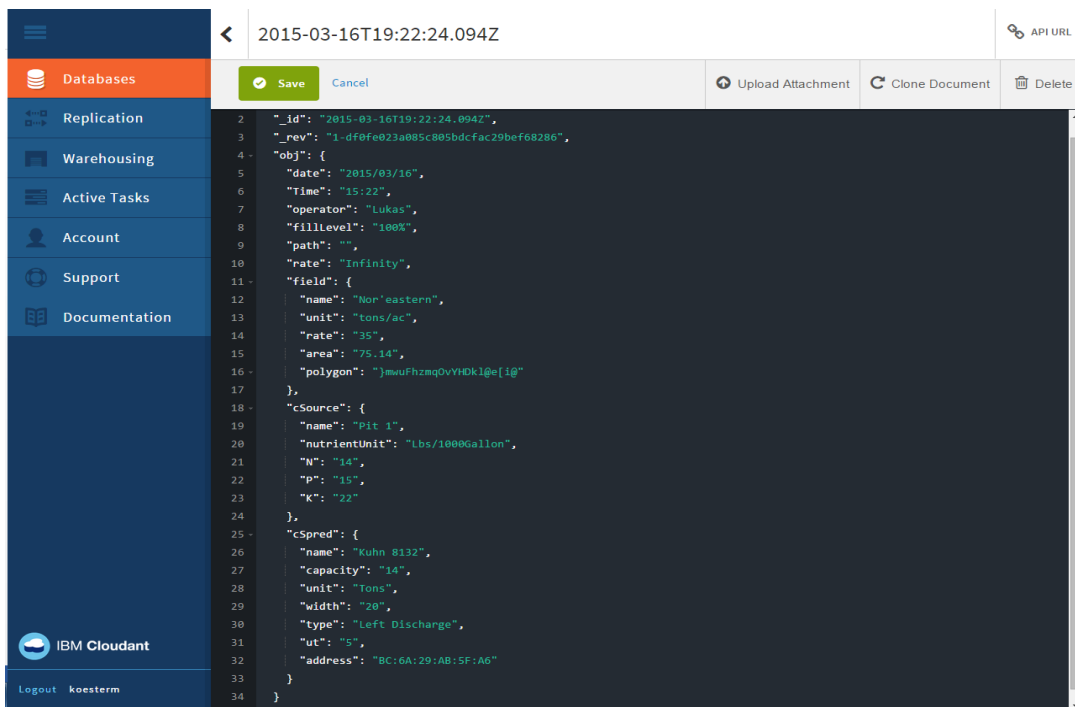


Figure 18: Cloudant database showing the selected record object's characteristics.

It is unrealistic to assume that users will desire, or be capable of editing the source code for creating a connection of their local database to a web based database like Cloudant. For this reason, an instructional document was created with full instruction on the creation of a Cloudant database (see Appendix C). Additionally, a new form was created in the Manure App where the user can input their username and password to their Cloudant database (see figure 19).

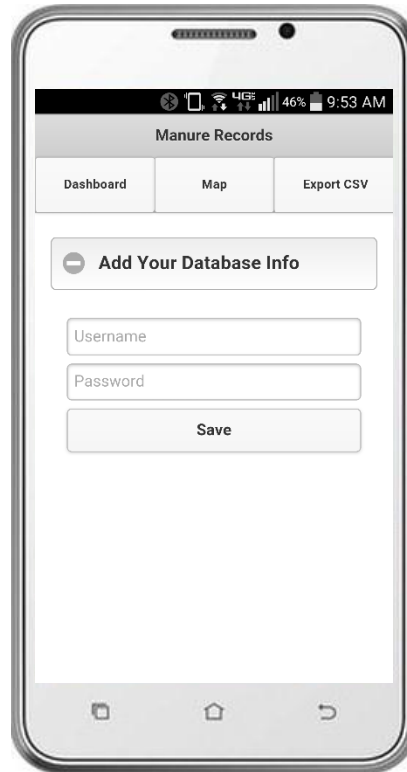


Figure 19: Database credential input to enable multi-device synchronization.

When the user inputs their credentials for the Cloudant database, the app saves this information in the local storage of the device. The information is then used to create the URL that syncs the mobile device's local database to the Cloudant database. An operation that has multiple operators can input the same credentials to sync all devices to the same database. At this time, Cloudant appears to be the best option for remote database mobile synchronization. While it is possible to edit the source code of the app to use other databases, the functioning Manure App (downloaded as is) is only compatible with Cloudant database service.

3.5.3 Automatic Field Identification

In order for the app to be aware what field application is being conducted, the app used the Google Maps API. The Google Maps API was used in the app to generate and edit contained Google maps. The API also includes the ability to return whether or not a point on a map is contained within a polygon object. Since the Manure App contains field boundaries for each field stored within the app, the field boundaries can be queried to return the field matching the current location. A function was created and called when manure application begins that returns the location of the first GPS point of the spread path. The function then queries all of the field boundaries in the field database to return the field object containing the GPS points that surround the point of application. The field object that is returned is set to being the field that manure application is occurring. Once the Manure App knows what field manure is being applied displaying the suggested application travel speed, the user no longer needs to select specific fields for application; the app handles this task on its own.

3.6 Sensor Testing and Threshold Development

Recognizing centripetal acceleration is proportional to angular velocity squared times the radius (Engineering Toolbox, 2014), there certainly are combinations of rotational speeds and radii which will and will not work well.

If the tag is only vibrating (not rotating on a shaft, pulley, or sprocket), then the average acceleration in each direction vector should remain constant (if “perfectly” oriented, might be 0 for x, 0 for y, and 1 g or 32.2 ft/s² for z). If the tag is rotating at 200 rpm at a 3” radial distance from the axis of rotation, it should experience 3.4 g (which,

when considering the 1 g due to gravity, would show up as an average 4.4 g resultant). The accelerometer data needed to be measured in order to determine a useful trigger (or threshold) for starting the recording process.

A simple sensor output logging app was written using the Evthings Studio's source code; this app receives transmitted data from both gyroscope and accelerometer sensors. The Bluetooth sensor tag transmits voltages from the current acceleration and gyroscope recorded from three axis X, Y, and Z. This data was transmitted at a recurring rate of 100 ms (every 0.1 seconds). Initially, the use of the gyroscope was anticipated to be the sensor of choice to detect rotation. After testing and reading through the sensor tag documentation, the sensor was found to be very noisy under normal operating vibrations, and had a rotational limitation of 250 degrees per second (~ 41 RPM). For this reason the accelerometer was selected for the use in communicating spreading status to the Manure App.

The transmitted data was converted to gravitational acceleration as a unit of measure (acceleration in gs = raw data/16; Sensor Tag User Guide, 2015). The sensor was tested and returned plus or minus one G at rest on each of the three axis based on the orientation of the sensor. The sensor was attached to a lathe and recorded data for 30 seconds under the following rotational speeds: 49, 138, 298, 567, 1000 rpm.

The values that were transmitted for all three axis were stored in a JavaScript array at intervals of 200ms (every 0.2 second). Additionally, to define the resultant of the vector acceleration, 5 second averages of the three axis values were squared and then summed together using the output displayed in a spreadsheet. The returned JavaScript object for each sample is shown below.

var data = {data_x, data_y, data_z}

The resulting data was then analyzed to find a base acceleration value that could not be replicated in accelerations due to other-than-manure-application tasks. By computing the resultant $((x_{avg}^2 + y_{avg}^2 + z_{avg}^2)^{1/2})$ average value, the average acceleration imposed on the sensor can be calculated.

CHAPTER 4. RESULTS

This project consisted of the development and testing of a platform independent application capable of storing data specific to manure hauling events. The objective was to assess the viability of creating a web application, being a platform independent tool that would be accessible by users irrelevant of the preferred device platform. The app was able to conduct data collection with minimal human control input specific to the operation status.

4.1 Final UI

The Final UI was completed and the screenshots are displayed below (see Figure 20 through 22). The UI is split up into four main portions of the app and its functionality: the dashboard and the spread event functionality, the object (field, spreader, source, and operator) lists showing the available objects and the current object being used for data collection, the Bluetooth connectivity page, and the Google Maps being implemented. The dashboard is the page that the user will most likely view most during operation. The dashboard shows the current spreading implement being used, the field where application is occurring, the source of manure that is being applied, and the operator conducting the operation.

Once all four of the above objects have been selected, the user can now press the start unloading button (or occur when the Bluetooth sensor triggers this action

if the sensor is connected and installed). When spreading is occurring, the spreading overlay and animation is shown. The overlay removes the option to edit objects during spreading. The animation is displayed to provide the user feedback that an action is occurring. During the unloading operation the user has the option to pause the unloading process for the case of turning around at the end of the field or needing to stop for any other reason. The Pause button triggers a pause function that stops recording GPS points for uploading into the storage array.

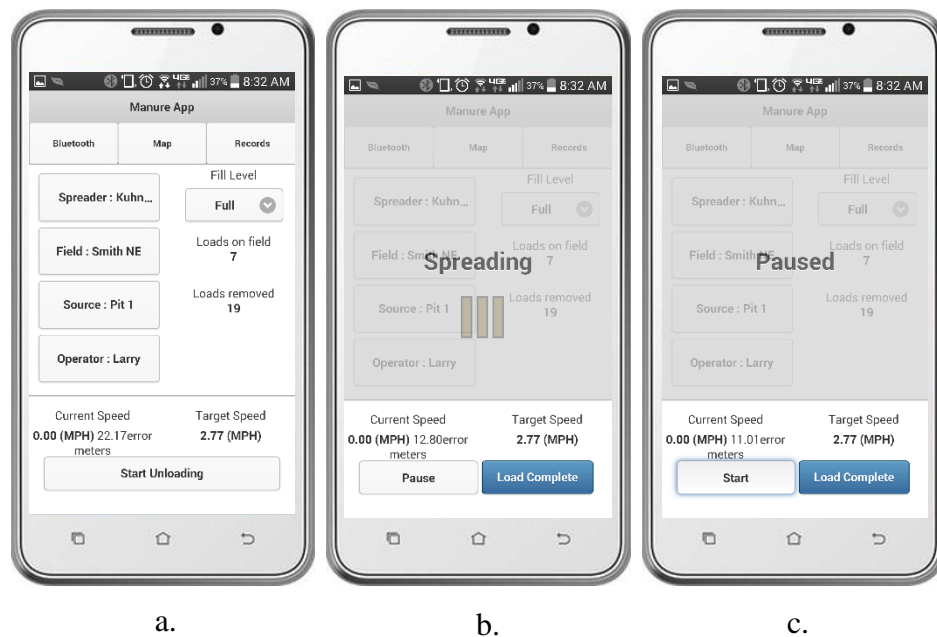


Figure 20: Final dashboard design (a.), spreading overlay and animation (b.), and paused indication for when the user wants to pause the unloading process (c.).

The object lists in the Manure App is the location where the user inputs data about their specific fields, spreading equipment, source, and operator name (see Figure 21). The object's data stored in the app is used for recording data specific to each operation for conducting accurate record generation. This stored data is needed for the case of autogenic data recording. The app needs to have stored information about each spreader

and field location in order to make the correct selection of implement and field that is currently undergoing manure application.



Figure 21: Final UI of the object lists: spreader list showing the available spreading implements (a.), field list showing available fields (b.), source list showing available sources (c.), and operator list showing available operators (d.).

4.2 Map Implementation

The Manure App uses the Google Maps API for the implementation and use of two Google maps. The main map used in the app is used for an overall visualization of field boundaries, information about each field, and polylines that represent each load of manure spread on each field. The add-field-boundary map allows the user to draw a field boundary by tapping the map at each point of the field. The Google Map API has built in map editing tools that were utilized. When a user wants to add a new field they tap the “Add New Field” icon in the Field List page of the app. Once the field name and desired rate has been added, the user taps the “Add Field Boundary” navigating to the add-field-boundary map (see Figure 13 a.). Once the user is done creating the new field boundary,

tapping the done button takes the user back to the Field List page where they can save the field. The new field is then displayed in the field list and can be selected for the next spreading action.

The main map page is a map where the user can view all of the saved fields, and all spread paths that have been created and stored during spreading events. The user can tap any of the fields and an information window will display the field's name and the area in acres contained within the field boundary (see Figure 13 b.).

The ability for the Manure App to connect to an external Bluetooth enabled accelerometer sensor enables autogenic data collection. An interface was needed for the user to select the connection action and the ability to select a specific Bluetooth tag to be referenced to each spreading implement (see Figure 22). When the app connects to a Bluetooth tag for the first time the app will not recognize the sensor tag's unique address; when this happens, the app displays the list of spreader names that the user has input into the app. Once the user selects the appropriate spreader the tag has been installed upon, the app stores the address within the selected spreader object. The app will now associate the sensor tag address to a specific spreading implement. When the user connects to the sensor tag in the future, the app will automatically select the spreader with the same stored Bluetooth address as the spreading implement being used by the operator to conduct hauling events.

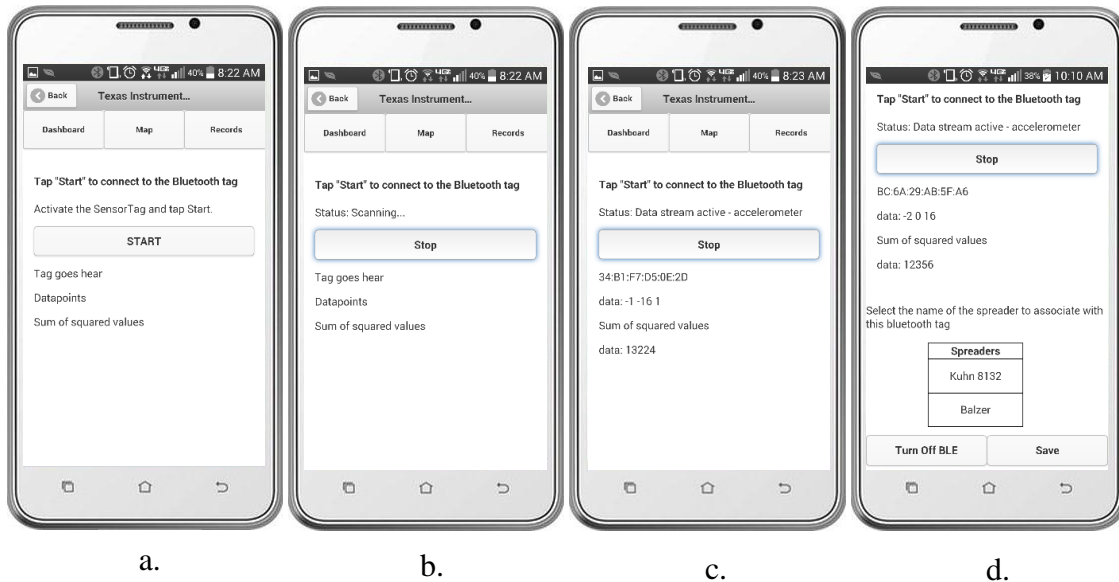


Figure 22: UI of connection process to the Bluetooth sensor tag. The main page not connected (a.), the app is scanning for a sensor tag after the start button has been pressed (b.), connection to the Bluetooth tag achieved (c.), reading data from the sensor tag's accelerometer (d.).

4.3 Test Results

Testing was conducted on the sensor output using a lathe for rotation. Rotational acceleration was measured and stored using a test app created using example code from Evothings Studio. Values were collected from all three axis every 0.2 second while the sensor was secured to a rotating shaft (lathe). Five sets of data were collected; one each for the speeds of 49, 138, 296, 567, and 1000 RPM. Data was collected in the test app and then exported as a CSV file (see Figure 23).

	A	B	C	D	E	F
1	Data07:41:03.0					
2	Sample 1	RPM	49			
3						
4	Start_Time	Date	Time	Accel_X	Accel_Y	Accel_Z
5	41:03.0	4/10/2015	41:04.1	-0.8125	-0.1875	0.5625
6	41:03.0	4/10/2015	41:05.2	-0.625	-0.125	1.375
7	41:03.0	4/10/2015	41:05.4	0.1875	-0.1875	0.6875
8	41:03.0	4/10/2015	41:05.6	1.375	-0.125	0.1875
9	41:03.0	4/10/2015	41:05.8	0.5	0	-0.75
10	41:03.0	4/10/2015	41:05.1	-0.3125	0.0625	-0.5625
11	41:03.0	4/10/2015	41:06.2	-1.0625	-0.0625	0.0625
12	41:03.0	4/10/2015	41:06.4	-0.875	-0.1875	1.25
13	41:03.0	4/10/2015	41:06.6	0.125	-0.25	0.375
14	41:03.0	4/10/2015	41:06.8	1	0.0625	1.3125
15	41:03.0	4/10/2015	41:06.1	0.5	0.0625	-0.625
16	41:03.0	4/10/2015	41:07.2	-0.4375	0.0625	-0.875
17	41:03.0	4/10/2015	41:07.4	-1.3125	-0.1875	1.125
18	41:03.0	4/10/2015	41:07.6	-1.0625	-0.1875	1.3125
19	41:03.0	4/10/2015	41:07.8	0.0625	-0.125	1.25
20	41:03.0	4/10/2015	41:07.1	0.875	-0.1875	-0.25
21	41:03.0	4/10/2015	41:08.2	0.125	0.125	-0.6875
22	41:03.0	4/10/2015	41:08.4	-0.6875	0.125	-0.375
23	41:03.0	4/10/2015	41:08.6	-0.875	-0.125	0.4375
24	41:03.0	4/10/2015	41:08.8	-1.1875	-0.25	1.0625
25	41:03.0	4/10/2015	41:08.1	0.4375	-0.0625	1.4375
26	41:03.0	4/10/2015	41:09.2	0	0.125	0.125
27	41:03.0	4/10/2015	41:09.4	-0.25	-0.125	-1.25
28	41:03.0	4/10/2015	41:09.6	-0.5625	0	-0.875
29	41:03.0	4/10/2015	41:09.8	-1.1875	-0.1875	0.75
30	41:03.0	4/10/2015	41:10.0	-0.75	-0.1875	2
31	41:03.0	4/10/2015	41:10.2	0.25	-0.125	0.3125

Figure 23: Sample data from the rotation of the sensor at varying speeds.

By calculating the resultant of the average value for each axis value (x, y, and z) and selecting a threshold value, the status of spreading operations can be known reliably. Analyzing the data collected, the minimum, average, and maximum resultant value was calculated for each RPM sample (see Table 4).

Table 4: Results of resultant acceleration

RPM	Resultant Value		
	Minimum	Average	Maximum
49	-0.9	-0.8	-0.7
138	-0.8	-0.7	-0.6
296	1.8	2.1	2.2
567	6.9	7.0	7.1
1000	8.8	9.1	9.4

Using the data output displayed in Table 4 to define a threshold value of 3.0 for average resultant acceleration, enables the Manure App to assess when spreading is occurring. A timer function was created that calls a function every five seconds. Each time the function is called, it calculates the average resultant value for the data collected during the previous five seconds. When the average resultant value is returned, an “if statement” handles the starting of the manure record generation process.

```
function isSpreadingOccuring(){
    if(resultant value >= 3){
        begin recording data;
    }else{
        do nothing;
    };
};
```

The above function is called each time the timer function is called. The returned average value is then compared to the threshold value of 3.0 to determine the spreading status of the implement. If the resultant calculated from the five second average is greater than or equal to three, spreading is occurring. If the value is less than three, spreading is not occurring. By installing a sensor on a spinning shaft or cog on the spreading implement that generates the same centripetal acceleration or greater, spreading status is determined.

Recognizing the variability in mechanical spreader implement design, the location of installation for the sensor tag may vary. For this reason, Table 5 was created to compare shaft rotational velocity with shaft radius to compute acceptable acceleration for proper communication of spreading status.

Table 5: Acceptable acceleration calculated from speed (rpm) & radius (in) highlighted in green. Cells colored in red generate excessive acceleration and could lead to sensor unreliability.

	Speed (rpm)									
	100	150	200	250	300	350	400	450	500	
1	0.3	0.6	1.1	1.8	2.6	3.5	4.5	5.7	7.1	
2	0.6	1.3	2.3	3.5	5.1	7.0	9.1	11.5	14.2	
3	0.9	1.9	3.4	5.3	7.7	10.4	13.6	17.2	21.3	
4	1.1	2.6	4.5	7.1	10.2	13.9	18.2	23.0	28.4	
5	1.4	3.2	5.7	8.9	12.8	17.4	22.7	28.7	35.5	
6	1.7	3.8	6.8	10.6	15.3	20.9	27.2	34.5	42.6	
7	2.0	4.5	7.9	12.4	17.9	24.3	31.8	40.2	49.7	
8	2.3	5.1	9.1	14.2	20.4	27.8	36.3	46.0	56.8	
9	2.6	5.7	10.2	16.0	23.0	31.3	40.9	51.7	63.9	
10	2.8	6.4	11.4	17.7	25.5	34.8	45.4	57.5	71.0	

Using Table 5 as a guide, the user can find a suitable location on their spreading implement for sensor installation based on shaft speed and distance from the center of the shaft to generate the acceleration on the sensor to correctly define status of manure application.

To assess the functionality of the Manure App with the connected Texas Instruments Bluetooth sensor tag, the tag was installed on a tractor (New Holland 3415) PTO shaft, with an approximate radius of one inch, and manure application was simulated. A field boundary was created around a gravel parking lot on the Purdue campus where simulated manure application would be occurring (see Figure 24).

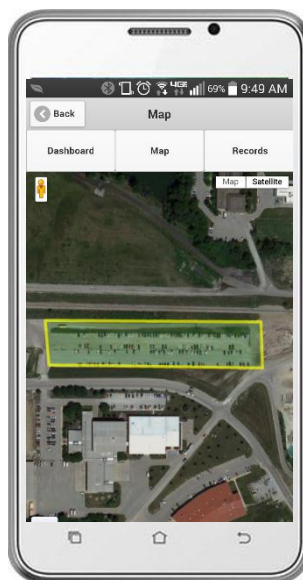


Figure 24: Test location for simulated manure application.

The Manure app was started on the mobile device (Verizon LG G3) and connected to the sensor tag installed on the PTO shaft. Once the tractor entered the field, the app correctly identified the field as it compared the current location to the field boundary. The PTO shaft was engaged and rotation speed was increased to an approximate operating speed of 350 RPM. This velocity was chosen due to the sensor being installed on the PTO, lacking an optional slower rotating shaft available on mechanical spreaders. If the PTO would have been spun up to operating RPM, the acceleration imposed on the sensor would have been excessive (see Table 5). The Manure App, running the algorithm for assessing the accelerometer data, read a value over that of the predetermined threshold value causing the Manure App to trigger the unloading function in the app collecting application data. Multiple loads of simulated manure were applied to the test field (parking lot) creating a polyline on the Google Map for each

unloading event. The map generated can be seen in Figure 25 and generated manure application records exported in Figure 26.



Figure 25: Spread paths displayed on map from simulated spreading operations.

Record													
Date	Time	Operator	Source Name	N	P	K	Nutrient Measure	Spreader	Spreader Capacity	Fill Level	Field	Rate applied	
4/15/2015	7:37	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	64.69	tons/ac
4/15/2015	7:40	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	61.15	tons/ac
4/15/2015	7:43	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	56.78	tons/ac
4/15/2015	7:45	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	54.12	tons/ac
4/15/2015	7:50	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	57.49	tons/ac
4/15/2015	7:53	Charlie	Pit2	18	14	12	lbs/ton	Kuhn1	20(Tons)	100%	Adm	55.91	tons/ac

Figure 26: Exported manure application records displayed as a CSV file.

Although the speed used to determine the spreading status was 350 RPM, it is not a “standard” speed on agricultural equipment. The two most common speeds used on tractor implements are 540 and 1000 RPM (ASABE, Standards, 2004). This may create some challenges for selecting a proper installation location for the sensor. However, most implement drivelines have several speed reductions. The use of the external

Bluetooth sensor served the purpose of proving the concept of autogenic data collecting apps. While this configuration was functional, CAN bus messaging technology may be more suitable for use in the future. Even so, the user could use an external Bluetooth sensor with the Manure App to communicate spreading status. Extreme caution, however, should be used anytime one is securing something to a shaft spinning at high speeds. These objects may come loose during operation and cause damage or injury.

CHAPTER 5. CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project consisted of the development of a platform independent application capable of recording data pertaining to manure hauling events. Development occurred in three main stages leading to the final app created in version 3.0. Final app testing was conducted simulating manure hauling operations to create application maps of each field coverage.

5.1.1 Version 1.0

Version 1.0 was developed creating a basic dashboard showing the number of loads applied to each field and the number of loads taken from each source. The load numbers were updated after a cycle of the user pressing “Start Unloading” and then “Load Complete” buttons. Separate pages were created for listing each type of object containing specific characteristics created by the user for each field, spreader, operator, and source. Each object was then stored in arrays created for each object type (spreader, field, source, and operator). Data taken from the current field undergoing application, current manure spreader in use, current source being emptied, and the operator conducting the task; this data combined created a new manure application record. Generated manure application records were then available for saving to the mobile device.

5.1.2 Version 2.0

Manure App version 2.0 involved an updated UI, the implementation of the Google Maps API, and the ability to use the mobile device's GPS sensor for recording and then storing the spread path's GPS points. Version 2.0 used the same data structure for each stored load data that was created in Version 1.0. During a spreading event, the app records the devices location as they travel through the field applying manure. The GPS data points are then stored within each load record for later parsing onto the Google map that is located inside the Manure App. The app was converted to a native app using Cordova to enable offline functionality.

5.1.3 Version 3.0

Version 3.0, a public releasable version of the Manure App records date, time, source, spreader, operator, field, spread path and application rate in autogenic mode. Autogenic capabilities were successfully implemented for hands free starting and stopping of data collected during manure spreading operations. Implementing a Google Maps API containsLocation() functionality, enabled the app to select the field in which spreading was occurring when the operator enters the field. Connecting the app to the external sensor tag reading accelerometer measurements, when combined with the developed algorithm, enabled the app to know when application has been initiated, is occurring, and has ended. The user has the option to use that app without the added automatic field selector and connected Bluetooth sensor. The user can select options to turn off certain functionality if the desire is to manually select fields, spreader, and spreading status.

The combined capabilities of the Manure App make data collection of manure hauling events accurate and nearly effortless to the user. Additionally, developing the Manure App has served as a proof of concept for further development of autogenic data collecting apps in agriculture.

5.2 Recommendations

5.2.1 Future Work

The objectives of the Manure App were achieved, yet further improvement could be realized through user feedback, testing, and added functionality. The next stage of development could commence through deployment to specific focus groups. Focus groups representing animal operations varying in size and type would bring valuable user perspective to implement and achieve desired functionality.

A tool that assists the user in creating nutrient application records for reporting to regulatory bodies can be met with hesitation. The Manure App generates data that is stored on the user's own account database and would not be accessible by the public. The data is secure, Operations may want to have the ability to quickly display their manure records in an easily viewed format (spreadsheet, pivot table, or PDF). They may also want a method for quick and easy aggregated reporting or display during an inspection. Focus groups are needed for further development of reporting and inspection capabilities that could be implemented into the app or external to the app..

Further testing would be beneficial to conclude the usefulness for a greater number of end users. Specific testing of the accuracy of the GPS location data and the calculation of the distance of the spread path to deduce an average rate of application on

various platforms should be completed to assure the approach is robust. Additionally, GPS accuracies vary with some devices, and could plague some users with difficulties, further testing would be needed to determine the variability due the users' device. The app should also be deployed on iOS devices due to their prevalence among users; this will involve more testing and development to ensure proper functionality for users running iOS.

The current app can deduce the spreading implement being used, and the field in which spreading operations have occurred. Implementing the ability for the user to specify the location of each manure source would enable later versions of the app to know which source is being used when the operator returns to the source's location. The app would no longer need to have the user specify the source with a tap event.

Using the Cloudant remote database, the Manure App could be further developed with added internal data editing capabilities. Currently, the Manure App syncs all data in the local database to the remote database. This can create data complications in the future when the user wants to change databases, because all data stored locally ends up in the new database. In some cases this would lead to an undesirable outcome if the user was wanting to start creating manure records in an empty database. Future versions of the app could allow the user to select which (if any) of the existing data to be synced to a new remote database. Data editing and exportability is important in agricultural records so convenience of these operations are needed. Future development could bring native/web apps that can export all data directly from the database outside of the Manure App.

Additional usefulness could be realized by implementing Open Ag Data Alliance (OADA) compliance. OADA is a database protocol that allows data to be collected from

previously incompatible sources, making data useful in ways not previously possible while the data owner security and flexibility (OADA, 2015). This could result in the ability to import field boundaries from users' field boundaries that have already been created in other software. This would eliminate the need for drawing field boundaries, which can be tedious in a mobile device. Additionally further use of manure management software could be populated from the OADA database being populated by the Manure App.

Drawing field boundaries could also be eliminated by utilizing state owned Common Land Unit (CLU) data. This data layer places boundaries around historically accurate field and ownership boundaries. By utilizing this data layer, the user could look at the map already displaying field boundaries generated from property lines. The user would only have to tap which boundaries they would like to use. In some cases the CLU data layer is outdated, so the user may still need to edit the boundaries.

Additional data layers for increasing the data collection and overall manure record keeping system. The addition of a weather layer could help to record the amount of rain 24 hours preceding and 24 hours after application has occurred. This is a required piece of data recorded in the operation's manure application records. Using Open Weather Map's API can return current and historical rainfall events based on geographical location (Open Weather Map, 2015). Other data layers are available for indicating the location of waterways in the United States. By adding a data layer that showed all the locations of waterways that are near sites of manure application, it would be possible to develop a way to add the visualization of manure application set-backs. Available

elevation data could also be beneficial to aiding in sensitive areas of land that may require caution with the application of manure.

Apps developed specifically for farm management could utilize the installation of the CC2541 sensor tag on each implement would make it possible to determine who is doing each activity on a farm at any given time. Installing a sensor tag on the each implement on an operation would create a network of production recording. The mobile device is able to wirelessly connect to the implement in use, retrieving user-stored information attributed to the implement. With this information, the app will be able to know what operation is being conducted, who is conducting the operation, where the operation is occurring, and when this operation is taking place. This information put together would be the basis of autogenic data collection.

Combined with the ability to sync data over multiple devices, a farm management app could collect operational data without the need for user input. Management apps with autogenic capabilities could keep FMIS information tools in real-time with current field operations without the need for users to update their next task or tap an icon each time their task status has changed.

REFERENCES

LIST OF REFERENCES

- Agent. 2015. Available at: <http://tryagent.com/>.
- Apple, Inc. Location and Maps. 2014. Available at:
<https://support.apple.com/en-us/HT202335>. Accessed February 2015.
- ASABE Standards. 2004. S203.14: Front and Rear Power Take-Off for Agricultural Tractors. St. Joseph, Mich.: ASABE.
- Automatic Labs. 2015. Available at: <https://www.automatic.com/>.
- Balsamiq Mockups. 2013. Available at: <http://www.balsamiq.com/products/mockups>.
Accessed February 2014.
- Beegle, D. 2003. Manure Spreader Calibration. Penn State University Extension.
Available at:<http://pubs.cas.psu.edu/freepubs/pdfs/uc200.pdf>
- Cloudant. 2014. Available at: <http://cloudant.com/>.
- Code, Indiana Administrative. 2012. "Title 327: Water Pollution Control Board." *Article 19: Confined Feeding Operations: Rule 7: Application Requirements*.
- Code, Indiana Administrative. 2012. "Title 327: Water Pollution Control Board." *Article 19: Confined Feeding Operations: Rule 14: Land Application of Manure*.
- Cordova. 2013. Available at: <https://cordova.apache.org/>. Accessed February 2015.
- CouchDB Web Database. 2014. Available at: <http://couchdb.apache.org>. Accessed January 2015.
- CSS. 2015. Available at: <http://www.webstandards.org/learn/external/css/>.
Accessed April 2015.
- Evthings Studio. 2014. Available at: <http://evthings.com/>. Accessed February 2015.

- Engineering Toolbox. 2014. Available at:
http://www.engineeringtoolbox.com/centripetal-acceleration-d_1285.html.
Accessed April 2015.
- Ess, D. R., Joern, B. C., & Hawkins, S. E. 1996. Development of a precision application system for liquid animal manures. *Precision Agriculture* pp 863-870.
- Field, H. L., and J. B. Solie. 2007. *Introduction to agricultural engineering technology: a problem solving approach*, pp. 124 - 126. Springer Science & Business Media.
- Geobells. 2015. Available at: <http://geobells.com/>.
- Geofencer. 2015. Available at:
<https://play.google.com/store/apps/details?id=com.arpacell.fencer&hl=en>.
- Gillespie, A. W., A. Diochon, B. L. Ma, M. J. Morrison, L. Kellman, F. L. Walley, T. Z. Regier, D. Chevrier, J. J. Dynes, and E. G. Gregorich. 2014. Nitrogen input quality changes the biochemical composition of soil organic matter stabilized in the fine fraction: a long-term study: *Biogeochemistry*, v. 117, p. 337-350.
- GitHub, Inc. 2014. GitHub. Available at: <https://github.com/>. Accessed November 2014.
- Google. 2014. Google Maps Javascript API v3. Available at:
<https://developers.google.com/maps/documentation/javascript/>. Accessed January 2014.
- Haynes, R. J., and R. Naidu. 1998. Influence of lime, fertilizer and manure applications on soil organic matter content and soil physical conditions: a review: *Nutrient Cycling in Agroecosystems*, v. 51, p. 123-137.
- HTML. 2015. Available at: http://www.w3schools.com/html/html_intro.asp.
Accessed August 2014.
- IDEM. Indiana Department of Environmental Management. YEAR. *Indiana confined feeding regulation program*, site. <http://www.in.gov/idem/files/cfomanual.pdf>. Indianapolis, IN, Accessed February 2015.
- ISOBlue. 2013. Available at: <http://isoblue.org/>.
- JSON. 2015. Available at: <https://www.json.com/>. Accessed September 2014.
- JavaScript. 2015. Available at: <http://javascript.info/tutorial/overview#what-is-javascript>.
Accessed April 2015.

- jQuery. 2015. Available at: <http://jQuery.com/>. Accessed August 2014.
- jQuery Mobile. 2015. Available at: <https://jQuerymobile.com/>. Accessed August 2014.
- Lemmermann, O. and Behrens, W. 1935. On the influence of manuring on the permeability of soils. *Z Pflanzenernähr Düng Bodenk* 37:174–192
- PhoneGap. 2015. Available at: <http://phonegap.com/>.
- PouchDB. 2014. PouchDB, JavaScript Database That Syncs. Available at: <http://pouchdb.com>. Accessed January 2015.
- Massey, Raymond E. and Haluk Gedikoglu. 2011. "Manure Application Rules and Environmental Considerations." In *Selected Paper prepared for presentation at the Southern Agricultural Economics Association Annual Meeting, Corpus Christi, TX, February 5*, vol. 8.
- Meyer, D., and D. D. Mullinax. 1999. Livestock nutrient management concerns: Regulatory and legislative overview: *Journal of Animal Science*, v. 77, p. 51-62.
- MMP. 2014. Purdue University Manure Management Planner. Available at: <http://www.purdue.edu/agsoftware/mmp/>.
- Moore, J. A., and M. J. Gamroth, 1993. Calculating the Fertilizer Value of Manure from Livestock Operations Oregon State University Extension Service, Corvallis, OR.
- Nowak, P., Cabot, P.E., Karthikeyan, K.G., Pierce F.J. 2005. Manure distribution patterns, operator decisions, and nutrient management plans. Symposium State of the Science Animal Manure and Waste Management, January 5–7, San Antonio, Texas, Conference Proceeding, L.19. Available from: http://www.cals.ncsu.edu/waste_mgt/natlcenter/sanantonio/Nowak.pdf. Accessed January 2015.
- OADA. 2015. Available at: <http://openag.io/>.
- Ohio State University Extension. 2009. Manure Nutrient Rate and Value Calculator. Available at: <http://agcrops.osu.edu/specialists/fertility/fertility-fact-sheets-and-bulletins>
- Open Ag Toolkit. 2013. Available at: <http://openagtoolkit.com/>.
- Open Weather Map. 2015. Available at: <http://openweathermap.org/>.

- Spolsky, J. 2000a. Painless Functional Specifications - Part 2: What's a Spec? Joel on Software. Available at: <http://www.joelonsoftware.com/articles/fog0000000035.html>. Accessed February 2014.
- Spolsky, J. 2000b. Painless Functional Specifications - Part 4: Tips. Joel on Software. Available at: <http://www.joelonsoftware.com/articles/fog0000000033.html>. Accessed February 2014.
- Sensor Tag User Guide. 2015. Available at: http://processors.wiki.ti.com/index.php?title=SensorTag_User_Guide&keyMatch=gyro+calculation&tisearch=Search-EN#Gyroscope_2
- Texas Instruments. 2015. CC2541 SensorTag Development Kit. Available at: <http://www.ti.com/tool/cc2541dk-sensor>. Accessed February 2015.
- UNL Water. 2014. Manure Management Software. Available at: <http://water.unl.edu/manure/software>.
- University of Michigan Extension. 2014. What's Manure Worth? Calculator. Available at: <http://www.extension.umn.edu/agriculture/manure-management-and-air-quality/manure-application/calculator/>.
- Welte, J., A. Ault, C. Bowman, A. Layton, S. Noel, J. Krogmeier, D. Buckmaster. 2013. Autogenic Mobile Computing Technologies in Agriculture: Applications and Sensor Networking for Smart Phones and Tablets. Paper No: C0340. Paris, France: EFITA
- Welte, J. May 2014. A Farm Management Information System with Task-Specific, Collaborative Mobile Apps and Cloud Storage Services. *MS Thesis*. West Lafayette, Indiana: Purdue University, Agricultural & Biological Engineering Dept.
- W3C. Geolocation API. 2014. Available at: <http://dev.w3.org/geo/api/spec-source.html> Accessed August 2014.

APPENDICES

Appendix A Functional Specification

Manure App Functional Spec

The purpose of the Manure app is to assist livestock operations with the task of manure management, as well as, creating and keeping complete records of manure hauling events for the sake of inventory and regulatory compliance. The Manure App is meant to be a tool that goes with the operator who is tasked with the hauling of manure. The goal of the app is to turn the operator's smartphone into a sensor/logger collecting data (for reporting compliance, improving management, etc.) while eliminating as much human input as possible; by reducing the need for manual entry, the record keeping process should be less tedious for the operator and farm manager and lead to more accurate and complete records.

Objectives

Management

The Manure App focuses initially on improving manure management by tracking (date, time, location, source, etc.) each load of manure. By outlining the spread path similarly to other precision ag operations, e.g., planting, tillage, harvesting...etc., as applied maps, by the load, will be generated which should improve the match between nutrient demand and supply.

The Manure App utilizes a database as the cloud storage. Data from each load of manure will be automatically uploaded and will be exportable as a comma separated values file for import into spreadsheet software. Most operations are very familiar with the use of spreadsheets and would require almost no learning curve to navigate

information. In this way, a manager will be able to easily track each load and know how many loads were applied to each field for record keeping and future manure placement.

Autogenic

A final version of the Manure app will have the ability to continuously (and “behind the scenes”) take information from the operator’s environment, information like: location, speed, machine, and task to enable the Manure App tool to calculate: load spread location, coverage rate, and map the field coverage. By the Manure App into a native application, capable of running in the background, be able to upload information to the user’s cloud service. Additionally, the app will be able to take in the previously stated information and be able to deduce when the operator is spreading and what field the application is taking place enabling the app to require minimum input from the operator to ensure easy and complete generation of manure hauling records.

Target Users and Use Scenarios

Mark Smith is a fourth generation farmer from Southern Indiana. He and his two brothers operate a 300 head dairy farm and a grain farm totaling nearly 5000 acres. Mark is the primary caretaker of the cows and with that, has the responsibility of planning and accomplishing the manure hauling tasks. Mark is concerned about the possibilities of more stringent regulations of smaller dairy farms. Having only 300 Holsteins, he does not need a commercial permit but thinks that regulations will begin to become stricter for smaller dairy farms as well. Currently there has not been any efforts to keep good records of manure hauling events on the Smith farm. This has not been a problem so far but Mark feels that it is only a matter of time before inspectors will be testing the streams around his farm.

Mark feels that he needs to begin to keep thorough manure records but really doesn't want to spend a lot of time and effort to write down each load and its destination; he doesn't have autosteer on the tractor used to spread, so cannot multitask and write these records while actually spreading. For him, finding the time to spend in front of the computer entering the load information would be impossible. Mark needs a system that would reduce the amount of time required to create and maintain these records. In addition to the record keeping benefits, Mark also believes that when he is able to more accurately control the application and management of his manure, he will then be able to save money by accurately reducing the amount of fertilizer that will need to be purchased. In order to do this, he needs some way of communicating the manure records conveniently to his agronomist who provides the seed and fertilizer recommendations.

The completed Manure App has been downloaded to Mark's mobile device and has been used for a couple of weeks. At first Mark didn't mind tapping the start and stop icons for each load of manure applied to the field, but after forgetting to tap for a couple of loads and needing to go into the data to edit the applied amounts, Mark is ready for some autogenic capabilities. Mark decides to order the recommended Bluetooth sensor and installs it on one of the spinning shafts on his mechanical manure spreader. Once Mark has the sensor installed and in standby, Mark powers up the Manure App. The app is automatically scanning for the Texas Instruments CC 2541 Sensor Tag when running. The app connects to the Bluetooth tag without a problem. Since the app has never connected to a sensor tag before, it does not recognize the sensor. The app knows that Mark is trying to pair the app to a specific spreading implement, so it displays a list of spreaders that have been stored in the app. Once Mark selects the spreader he is using, the

app stores the sensor tag address with the spreader chosen for future connection and spreader selection.

Charlie Williams owns and operates a custom harvesting and manure hauling business. When harvest is over Charlie gets his fleet of six tractors and spreaders, three semis, and manure pumps together and makes the journey across the country to all of his customer's dairy and livestock operations to spread manure. Charlie is responsible for spreading the manure of 110 customers over nearly 12,000 acres. Like most custom farming, custom manure hauling is a fast-pace business and Charlie Williams' custom spreading business is no different. His crew is spreading at any moment that weather and conditions permit and if they are not spreading, they are travelling to the next customer's location. Charlie would like to have a more stream-lined way of keeping track of his equipment in the field and to have a reduction of errors in the generation of the spreading records. In addition, He would also like for the reports to be in the simplest form possible so that they can be easily use these records to move into the billing information. Charlie likes the idea of using spreadsheets for keeping these records as this is his current documentation method.

Charlie sees value in implementing the Autogenic portion of this app. Charlie understands that in this fast-pace industry it is very inefficient to have to go back and correct an error in one's records whether it is someone forgetting to record a load of manure or recorded a load going to the wrong field. This becomes more of a problem when you multiply employees and rapid movement from one customer to another. Charlie feels that it is beneficial to remove human input when possible and also likes the

prospect of minimal man hours required to simply input information required to generate records, it simply takes place on its own.

Charlie also sees potential benefit to his customers. If they can have as applied manure records, they can more easily optimize other nutrient inputs. These same records can help Charlie with his “honest billing” philosophy. Additionally, some of his spreading situations actually have two customers: the livestock farm with the manure and the crop farm which is getting the manure applied. As the third party service provider, Charlie can help both of these farmers with their records.

Charlie would most interested in the autogenic capabilities of the Manure App. Charlie would immediately purchase and install the Bluetooth sensor tag on each of his spreaders. Having a network of spreaders all connected to his operator’s mobile devices would greatly reduce the complexity of who spread each load of material and in what field. This would improve is book keeping as well as generating accurate as applied nutrient maps without the need for expensive monitors installed in the cab of each tractor. The Manure App is free and the sensors are \$25 each. Charlie likes these numbers. Additionally, Charlie finds great value in the ability to keep all the operators data stored to the same database and synced continually during field operations. Charlie exports the generated data each evening after field operations have ceased. Sending the data to the office computer, he now has the ability to view the day’s progress and check the quality of application.

Considerations for Improving Functional Specifications

Additional information would be beneficial for the development of the Manure App. In addition to the basic functionality of the app, the views of the potential users on

regulatory requirements would be beneficial for creating a tool that can assist in the reporting process. Some questions that could be answered by future users could aid in developing desired functionality.

- What is the size of the operation?
- What is the level of interest for data collection and operation record generation on the operation?
- At this time, are manure application records being generated on the operation?
- How are the manure records generated?
- Are you satisfied with the accuracy of those records?
- Are there currently steps being made to improve record reporting?
- Do manure application records need to remain confidential?

User Interface

Mockups designed using mybalsamiq.com



Figure 1 - App Dashboard showing first time opened (left) and after use (right)

Dashboard

The dashboard is the main screen of the app that will most likely be used the most by the user. For that reason it is important that all relevant information is displayed in a simple and well organized fashion. The dashboard should display information that the operator would need access quickly while being able to continue spreading without added distraction or the navigation of long lists within the app. The dashboard houses other operations of the app as well.

The navigation bar is at the top of the screen and serves several purposes. The “view” icons is where the user can view a Google map of his fields showing manure coverage of each field in a similar way that a yield map appears. This “List” icon is where the user can look up current and past spreading events in a similar format to a spreadsheet. Most of the spreadsheet functionality will be saved for using on a pc, but

having this capability will help for quick look-up needs of the user. The list navigation is located at the center left of the dashboard. These icons take the user to spreader list and field lists for the purpose of record keeping and background calculations. The “Op” icon will likely change, but the purpose is to have a planning tool built in so that the user will be able to plan manure hauling events for future times. This would help with managing personnel and improving logistical efficiency. Additionally, this may help with the complexity of the autogenic functionality of Manure App Version 3.

The center of the dashboard will be the location of the most essential data that the user would need to find with no more effort than a glance. The bottom of the dashboard will be the location of the unloading functionality of the app (Non-Autogenic version).



Figure 2 - Spreader List

Spreader list

The Spreader list is the portion of the app where the user can save multiple spreaders and select them based off of which one is being used. After pressing the spreader icon from the dashboard, the user can either, select an existing spreader, or add new spreader.

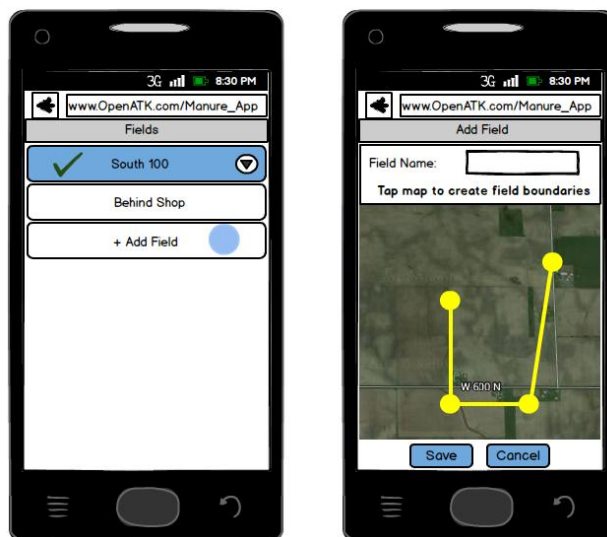


Figure 3 - Field list and add field function

Field List

The Field list is the location of the app where the user will be able to select which field that he is planning to spread, or add a new field. By selecting the “Add Field” icon, the user needs to type in the name of the field and then tap on the map to create the polygon.

Source

If the user has a large operation, it is highly likely that there are multiple sources of manure on site. If so; the Source List is the location in the manure app where the user can add in source data for later use. For each source added to the list, the user will be able to add the nutrient values for accurate tracking of each unit of N, P, K applied to their fields. This will make reporting much easier and accurate. It will also aid in the reduction of

artificial fertilizer needed for crops. All of this translating to added savings for the operation.

Unload Function

There are only two buttons in the unload portion of the app. The large button contains all of the unload functionality, tap it once to start, and tap again to stop. Once the operator has pressed the button the app will begin to track their progress across the field by recording Lat/Long position continually during travel. This path will then be displayed on a Google map showing exactly where this load of manure was hauled.

By knowing the spread width of the machine and the capacity, the app can then calculate the rate of application and save this information to the Google Spreadsheet. When the app is unloading the unload button will change to show “Load Complete” and a “Pause” button is displayed. The “Pause” button is displayed so the user can pause the recording of GPS points when turning around at the end of the field or any other reason that would need the spreading operation halted. Once that spreader is empty, the user can press the “Load Complete” button that will trigger the upload of this load information to the web based database for syncing with other devices.

Records

The records will be generated in the app and then uploaded to the Cloud, in this case, Cloudant. The user will create their own database for use with the Manure App. The app will then be able to upload information to the user’s Cloudant database syncing and later export.

Developmental Stages

Version 1

- Save information about each load of manure (e.g. date, operator, spreader, field, and source)
- Save the input data in browser local storage
- Display manure records in the app
- Export CSV file capability

Version 2

- Finalize and implement user interface
- Utilize HTML5 Geolocation API
- Implement Google Maps API for drawing and viewing field boundaries
- Convert Manure App from a web application to a native application for continuous operation on a mobile device

Version 3

- Implement Bluetooth connectivity to external sensor tag for communicating spreading status to the app
- Utilize Google Maps API for detecting the field that the user is inside during operation
- Implement web based database for multi-device use and synchronization

Testing

- Collect data from sensor tag during simulated manure application for analysis
- Develop an algorithm for determining the status of application from sensor output
- Install on implement for generating spread path and manure record

Appendix B Manure Records Documents

327 IAC 19-7-5 Manure management plan

327 IAC 19-7-5 Manure management plan

Authority: IC 13-14-8-7; IC 13-15-2-1; IC 13-18-10-4

Affected: IC 13-11-2; IC 13-14; IC 13-15; IC 13-18; IC 13-30

Sec. 5. (a) A manure management plan must be developed and submitted to the commissioner that contains the following:

- (1) Procedures for soil testing as described in subsection (c).
- (2) Procedures for manure testing as described in subsection (d).
- (3) Plot maps as described in section 2(a)(1) and 2(b) of this rule.
- (4) If applicable, the land application acreage requirements waiver, as described in 327 IAC 19-14-2(d).

(b) If applicable, the manure management plan must also contain a description of any:

- (1) alternate methods proposed by the applicant for managing of the manure; and
- (2) other practices to be used that assure the CFO meets the performance standards in this article.

(c) A soil test must be obtained that provides sufficient information about soil fertility to allow for nutrient recommendations for existing or planned crops. Soil tests may not represent more than twenty (20) acres per sample. The frequency of this testing must be:

- (1) specified in the manure management plan; and
- (2) conducted a minimum of once every four (4) years unless a different frequency is approved by the department in writing and is included in the manure management plan.

(d) A manure test must be obtained that provides sufficient information about the manure content to allow for nutrient recommendations for existing or planned crops and to minimize nutrient leaching. The frequency of this testing must be:

- (1) specified in the manure management plan; and
- (2) conducted a minimum of once every year.

(e) Manure samples must be representative of the manure that is land applied. If manure is mixed from separate manure storage facilities prior to land application, a composite sample may be taken. If manure is land applied from separate and distinct storage facilities, a sample must be taken from each unique production system.

(f) A manure management plan must be submitted to the department at least one (1) time every five (5) years and with any approval application and renewal application to maintain a valid approval for the CFO. A copy of the current manure management plan must be maintained in the operating record. (*Water Pollution Control Division; 327 IAC 19-7-5; filed Feb 6, 2012, 2:58 p.m.: 20120307-IR-327090615FRA, eff Jul 1, 2012*)

Manure application record (IDEM, 2015)

CONFINED FEEDING OPERATION (CFO) MANURE APPLICATION RECORD *(Required Information)*

As required by 327 IAC 19-14-3(f), you must document and maintain information regarding the acreage used for land application of manure. You may use this form to document compliance with the rule. It is suggested that you use a separate form for each tract of land you use for manure application in a calendar year.

Field ID: _____ **Expected Crop:** _____ **Expected Yield:** _____

Soil P Result(s) _____ **Past Crop:** _____
(BrayP1/Mehlich3 Test Method)

Date(s) of Application (MM/DD/YY)	Manure Source (Storage Facility ID)	Amount Applied (gal/ton/#)	Acres Covered	Available N and P From Manure Tests (#/ton or #/1000gal)	Method of Application S=Surface Injected SI=Surface/Incorporated	Pounds N and P Applied per Acre		Additional N Inputs	Precipitation Events			Field Tile Monitoring	Applicator Initials	Comments (Slope, Soil Condition, etc)
						N	P		24 Hours Prior	During	24 Hours After			

Portions of the data above are derived from sample analysis results, performing calculations, reviewing soil maps etc. All of these supporting materials listed below are required to be maintained in your operating record and will be reviewed during compliance inspections to check proper agronomic rates.

- An explanation of the basis for determining application rates which includes the calculations performed to arrive and nitrogen and phosphorus to be applied, and calculations showing the total amount of each nutrient actually applied to each field.
- Test methods and results from manure, litter, process wastewater, and soil sampling.
- The date or dates of manure, litter, and process wastewater application equipment inspection.
- USDA soil survey maps of currently available land application sites.
- For application to highly erodible land, as specified in 327 IAC 19-14-4(j), a written conservation plan with an explanation of conservation practices used. The conservation plan must be implemented prior to land application.

Calculations for manure application from Indiana IAC 327 - 19 - 14 - 4

CONFINED FEEDING OPERATIONS

FEBRUARY 13, 2003 AND CFOS APPROVED FOR INITIAL CONSTRUCTION AFTER THE EFFECTIVE DATE OF THIS ARTICLE	
Soil test level (ppm)	Application rate
0-50	N based
51-100	1.5 × P crop removal
101-200	1.0 × P crop removal
201+	0

(e) Beginning with the effective date of this article, CFOs and CAFOs not listed in subsection (d) must comply with the phosphorus application rates in Table 2:

Table 2.				
PHOSPHORUS APPLICATION RATES FOR ALL OTHER CFOS AND CAFOS				
Soil test level (ppm)	YEAR ¹			
	2012-2013	2014-2015	2016-2017	2018+
0-50	N based	N based	N based	N based
51-100	1.5 × P crop removal	1.5 × P crop removal	1.5 × P crop removal	1.5 × P crop removal
101-200	1.0 × P crop removal	1.0 × P crop removal	1.0 × P crop removal	1.0 × P crop removal
201-250	0.9 × P crop removal	0.75 × P crop removal	0.75 × P crop removal	0
251-275	0.9 × P crop removal	0.75 × P crop removal	0.5 × P crop removal	0
276-300	0.9 × P crop removal	0.75 × P crop removal	0.25 × P crop removal	0
301-350	0.7 × P crop removal	0.5 × P crop removal	0	0
351-400	0.7 × P crop removal	0.25 × P crop removal	0	0
401+	0	0	0	0

¹ Multiple years of phosphorus may be applied as long as the net average of phosphorus does not exceed the amounts indicated in Table 2.

(f) The following land application information must be added to the operating record as needed in accordance with required time frames established in this article and IC 13-18-10 and must be maintained and updated in the operating record:

- (1) Expected crop yields.
- (2) The date or dates manure, litter, or process wastewater is applied to each field.
- (3) Precipitation events at the time of application and for twenty-four (24) hours prior to and following application.
- (4) Test methods used to sample and analyze manure, litter, process wastewater, and soil.
- (5) Results from manure, litter, process wastewater, and soil sampling.
- (6) An explanation of the basis for determining manure, litter, and process wastewater application rates.
- (7) Calculations showing the manure nitrogen and phosphorus to be applied to each field.
- (8) Total amount of nitrogen and phosphorus actually applied to each field, including documentation of calculations for the total amount applied.
- (9) The method used to apply the manure, litter, or process wastewater.
- (10) The date or dates of manure, litter, and process wastewater application equipment inspection.
- (11) USDA soil survey maps of currently available land application sites.
- (12) The type of manure applied.
- (13) A written conservation plan with an explanation of conservation practices used must be completed and implemented prior to land application on highly erodible land, if required in section 4(j) of this rule. CAFOs with a NPDES permit must have a nutrient management plan prior to land application on highly erodible land.

(Water Pollution Control Division; 327 IAC 19-14-3; filed Feb 6, 2012, 2:58 p.m.; 20120307-IR-327090615FRA, eff Jul 1, 2012; errata filed Nov 9, 2012, 11:09 a.m.: 20121128-IR-327120607ACA)

Appendix C Creating User's Remote Database

Step-by-step guide to creating one's own Cloudant database for storing manure hauling records generated using the Manure App.

Step 1 – create an account

Navigate to [Cloudant.com](https://cloudant.com)

Click “Sign Up” at the top right of the page in orange

You should see a page like this:

Complete the form on your right to create your Cloudant DBaaS account. Cloudant is free to start, always on, easy to use, and **feature rich**.

Already have an account? [Sign in](#)

Username .cloudant.com
Lowercase, alphanumeric, and at least 3 characters

Password
At least 8 characters

First name

Last name

Company

Email

Data Location Moonshine - SoftLayer, US E

By clicking the button below, you agree to Cloudant's [terms of service](#).

I agree, sign me up

Input your information:

Username – this will be the name of your database and will be used for logging into your account. Save this information because it will be input into the Manure App for syncing.

Password – Used for logging into your account. This will also be input to the Manure App for syncing data.

First name

Last name

Company - does not have to be actual company name... Purdue University was used during development.

Email

Location – This allows you to select the location that you would like to have your data stored.

Click “I agree, sign me up”

Once this is accomplished, you will get an email telling you that your account has been created and you can log in to see you data stored. (There will be no data until you sync the Manure App - next step).

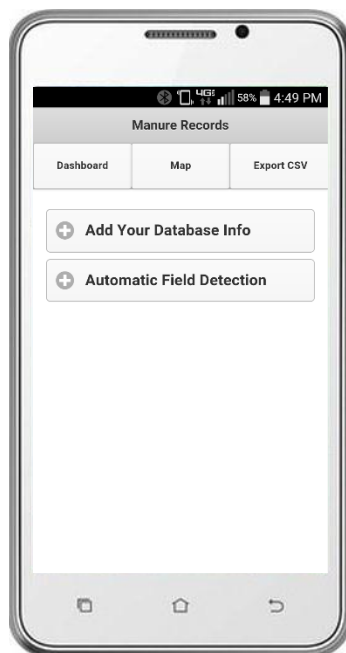
Step 2 – Syncing the app to your database

Now that you have a web based database, you can sync you data generated with the Manure App for safe storage and multiple device syncing of records. (See below figures for visuals for adding database id and password).

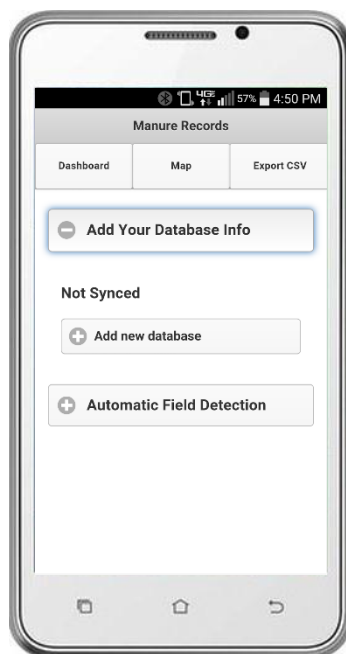
Open up the Manure App on your device.

From the Dashboard, tap “options” in the top right corner of the screen. This navigates to the options page of the app.

Tap “Add database info” (this screen will display the name of the database once the app is synced)

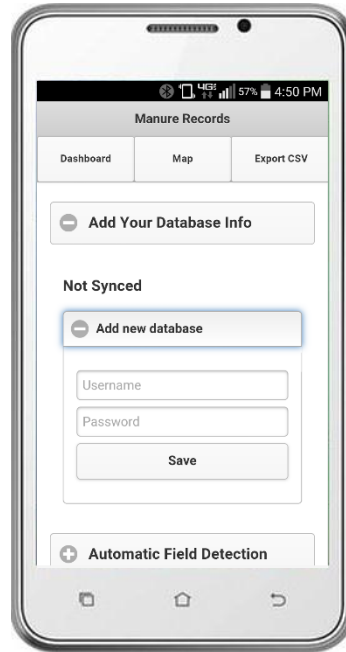


Tap “Add new database”



Add in the Id and password created when you completed the creation of your account (see step one).

Tap “save”



Once you have saved the information, the app will call your database to sync. This can take a couple of moments depending on your internet connection.

Once you have synced the database to the app, you can now generate data in the app and it will be synced to your Cloudant database. Additional devices can be added to the database by following the above steps. Data will be synced to the database and other connected devices as it is generated.