Advancing Quality Assurance Through Metadata Management: Design and Development of a Mobile Application for the NRDC

Connor Scully-Allison[*], Hannah Muñoz[*], Vinh Le[*], Scotty Strachan[*],
Eric Fritzinger[*], Frederick C. Harris, Jr.[*], and Sergiu Dascalu[*]
University of Nevada, Reno, Reno, NV, 89509, USA

Abstract

In this paper we present the design, implementation, and impacts of a cross-platform mobile application that facilitates the collection of metadata for in-situ sensor networks and provides tools assisting Quality Assurance processes on remote deployment sites. Created in close conjunction with scientists and data managers working on environmental sensor networks, this paper details the software requirements, specifications, and implementation details enabling the recreation of such an application. In a discussion on how this software improves on existing techniques of logging contextual metadata and quality assurance information, we show that this application represents a significant improvement over-existing methods. Specifically, the proposed application allows for the near-real time update and centralized storage of contextual metadata. Compared to prior methods of logging, often physical notebooks with pen and paper or program comments on embedded field sensors, the method proposed in this paper allows for contextual information to be more tightly bound to existing data sets, ensuring use of collected data past the lifetime of a specific research project.

Key Words: Data management, data science, mobile application, sensor networks, software engineering, cross platform mobile development.

1 Introduction

Individual researchers and one-off projects dominate the model of data collection in traditional climate/environmental research [9]. In traditional research, single use data proves extremely effective at answering a singular project's research questions and fulfilling research requirements attached to funding streams. However, despite the short-term success of such a model, a clear problem arises when another research team wishes to use this previously collected data, or when data need to be integrated into larger syntheses [3]. This problem drives the need for complete, accurate, and usable metadata.

Metadata is considered a major part of the data lifecycle. Creation of metadata include information surrounding the data set, such as format, file names, and measurement units, and information about the experiment producing the dataset, such as documenting data processing steps and contextual information to the data [13]. Its purpose is to make the datasets quick and easy to understand [20]. Unfortunately, that is not always the case. Traditionally, scientific metadata is kept in notebooks and papers, a holdover from days before computers. This creates fractured data, where it's hard to relate the electronic data sheets to hand written documents [18].

Due to the narrow focus of typical projects, only the original researchers intimately know how the data was generated. Metadata is often non-standardized, incomplete, and stored in temporary formats. Some communities and organizations have developed their own metadata standards; however, several different metadata standards can exist within any given discipline [7]. Eventually, over time -- or given enough distance -- the value of these data sets is diminished to other researchers and the public. It becomes harder to recover and ascertain contextual information that is essential to decoding it and metadata standards rarely address long-term preservation [18]. Methods for uniform quality assurance and metadata collection are being recognized as the next major challenge for data intensive science as collection becomes increasingly automated and results globally disseminated [21].

This paper proposes a mobile application that manages and maintains quality assurance metadata about data collected from remote sensor networks. The Quality Assurance (QA) Application described in this paper represents a positive step forward into modern data collection models by centralizing, modernizing, and standardizing contextual metadata for environmental sensor systems. The QA App gives technicians and researchers a tool for dynamic modification and creation of contextual information relating to hundreds of live data streams in a statewide sensor network.

Continuing from here, this paper is structured as follows: Section 2 presents a survey of scholarly works related to the app developed; Section 3 details the software specifications and use cases of the application; Section 4 discusses the architectural and user interface design of this application; Section 5 discusses how the application was implemented; Section 6 evaluates the success of implementation and Section 7 explores ideas for future developments.

2 Related Work

At the broadest level, Quality Assurance refers to the preventive maintenance and management process employed to reduce inaccuracies in data automatically logged by sensors [5]. Although many works touch on the idea of Quality Assurance,

* Department of Computer Science. Email: cscully-allison, hannahmunoz, vle@nevada.unr.edu, scotty@dayhike.net, ericf@unr.edu, fred.harris@cse.unr.edu, dascalus@cse.unr.edu.

ISCA Copyright© 2018

the most seminal motivating work published on the subject comes from a 2013 paper "Quantity is Nothing without Quality: Automated QA/QC for Streaming Environmental Sensor Data" [4]. In this paper, the authors put forth a comprehensive, generalized set of practices to optimize QA on environmental field sensors. They suggest that QA procedures be automated, well documented, and complete metadata maintained alongside data. This work presents Quality Assurance as a "process oriented" approach to data management, this strongly implies that no single software solution can provide effective QA but can only rather aid the QA process performed by humans. Accordingly, a significant number of background works on this subject study the analysis and creation of software that best facilitates good QA practices.

A further example of a motivating work indicating the need for Quality Assurance practices in the paper "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis" [1]. This paper outlines the development of a software environment that "addresses technical challenges related to accessing and using heterogeneous sensor data from within the Kepler scientific workflow system." Within the bounds of their end-to-end examination of this existing sensor network workflow, the authors indicate on several occasions that a clear need exists for quality assurance practices with in-situ sensor networks. They also acknowledge that existing software solutions used to facilitate these practices are not well developed. Our QA application intends to fill this proposed gap.

Outside of motivating works driving research in QA, there also exists several works that explore the practical or theoretical implementation of Quality Assurance processes and software. One, "Meta-information concepts for ecological data management" represents an early survey of many data management needs for ecological sensor data collection [12]. This paper suggests exactly at which stage in the data collection pipeline to place QA processes and software. Another paper, "Anomaly detection in streaming environmental sensor data: A data-driven modeling approach" shows a more practical approach to Quality Assurance by suggesting that expected shifts in the quality of data can be anticipated by using data-driven modeling techniques. Although the approach taken by the authors of this paper is not representative of the approach we undertook, it shows that there exists a strong interest in applying modern software engineering techniques to the problem of Quality Assurance.

Along similar lines, in the paper, "Automatic processing, quality assurance and serving of real-time weather data" the authors demonstrate that there exists a strong interest in developing software to automate and streamline the process of Quality Assurance on environmental sensor data [22]. The authors of this paper propose a software to manage and utilize statistical metadata that can indicate the quality of data through uncertainty values. The concept of collecting metadata in a standardized format for quality assurance strongly reflects the goal of the Quality Assurance software developed and detailed in this paper for the NRDC. However, our approach is unique in that it is oriented towards metadata collection as an end goal rather than as a supplement to Quality Control computations. Taken together, the above papers generally indicate a strong research interest external to the Nevada Research Data Center (NRDC) in the development of Quality Assurance software, however there also exists a well-documented interest in developing software within this organization as well.

The NRDC is a data management organization dedicated to the, "storage, retrieval, and analysis of research data that is relevant to the needs and interests of the state of Nevada" [15]. Conceived apart of a NSF Track 1 project, the NRDC represents the collaborative efforts of top research institutions, including the University of Nevada Reno, the University of Nevada Las Vegas and the Desert Research Institute [14, 16]. It presently supports the data sets of five projects and works actively with external research networks to disseminate and preserve data for continued research.

References to the considerations of a Quality Assurance system for the NRDC appear in early literature proposing practices and architecture for its predecessor project NCCP [11]. These works present Quality Assurance and Quality Control (QC) as crucial elements of any large scale environmental research project. They also impress upon the reader a need for a standardized and centralized set of tools which enable universal comprehension of data being collected. From this specific need to improve on existing QA practices, a quality assurance application was conceived.

3 Software Specifications

The QA App was developed with many functional and nonfunctional requirements in mind. These requirements were decided on after extensive talks with several data management experts and stakeholders. Detailed in Table 1, these requirements guided design and development of core functionalities for the QA application. Using an agile development method, these requirements went through several iterations before settling into the current list.

The nonfunctional requirements set many constraints on development, but most importantly dictated that the system should be multiplatform, upload and download data at only one point and perform logins with an SSH certificate. This set of requirements informed development by cementing the software and architecture used to implement the app and indicate how it should interface with the backend server.

Using these functional requirements, a series of use cases were constructed and mapped in a use case diagram, found in Figure 1. This process informed the principal design phase of this applications construction and was frequently referenced or tweaked alongside the software specifications through the implementation phase. The description of each use case follows:

- LogIn

Field technicians must log into the app. Once logged in, technicians are given a list of projects they are associated with. This helps reduce the amount of unnecessary data downloaded. Technicians can also add new entries and upload them to the

Table 1: Functional requirements

Functional Requirements	Description
Input Data	The user shall be able to enter new data into the QA app.
Upload Data	The user shall be able to upload data to a secure database.
Read Data	The user shall be able to download entries from the database to their mobile device and view previous data entries.
Edit Data	The user shall be able to edit previous entries and upload the change data to the database.
Navigate Data	The user shall be able to move between different screens of the app, and input data.
User Authentication	The user will be able to authenticate themselves to access secure functionalities.
Delete Server Data	The user, with proper authorization levels, shall be able to delete data stored on the database.
Upload Photos	The user will be able to launch the device's camera and upload a photo from their photo gallery on their device.
Save Unsynced Data	The user shall be able to save data locally on their device to upload it to the server at a later point

server. Administrative technicians, once verified through the log in, are given the ability to edit or delete entries already synced to the server.

- SyncToServer
  Connects to the server and uploads new data entries found on the phone. Then, downloads new data found on the server. The app can manually be synced by pressing the synchronization icon on the header bar on the front page.

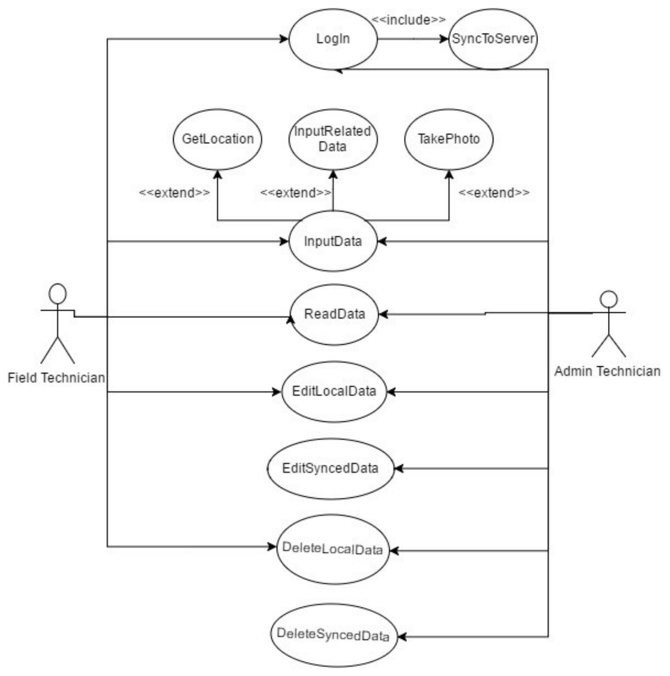


Figure 1: Diagram showing use cases and their actor interactions for the quality assurance application

- InputData
  Allows the user to input new data. Opens a blank template for whichever dataset they are choosing to input. Once finished, it is saved to local memory until synced to the server.

  - TakePhoto
    Opens the phone's camera app to take a picture that can be uploaded alongside the data set, like the System in Figure 2. Not every data set can have a photo.

  - GetLocation
    Uses the phone's GPS to fill out latitude and longitude coordinates. Only two types of data sets need GPS location.

- ReadData
  All users must be able to view the data, regardless of whether or not they are logged in. This is so users who are not a part of the project, but are interested in the data, can view it. To read the data, users need only to navigate to their desired object and click on the name.

- EditLocalData
  Users are allowed to edit entries that have not yet been synced to the server. Users can navigate to unsynced data entries and select the edit button to change them.

- EditSyncedData
  Administrative technicians are allowed to edit data already synced to the server. If an admin is logged in, they can edit entries by navigating to it and clicking the "Edit" button. If the user is not an admin, this button will be greyed out. The changes will be uploaded the next time the app is synced to the server.

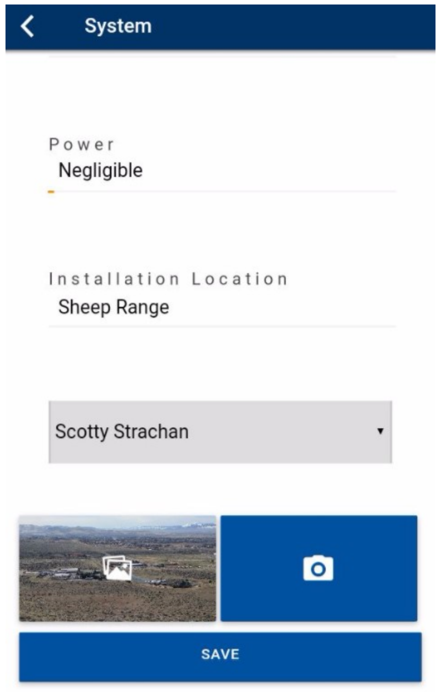


Figure 2: An example of inputting data with a picture from the phone's camera

- DeleteLocalData
  Users can delete entries that have not yet been synced to the server. Users can navigate to unsynced data entries and select the delete button to remove them.

- DeleteSyncedData
  Administrative technicians are allowed to delete data entries already on the server. If an admin is logged in, they can delete entries by navigating to it and clicking the "Delete" button. If the user is not an admin, this button will be greyed out. The changes will be uploaded the next time the app is synced to the server.

4 Software Design

4.1 Architectural Design

When designing the Quality Assurance Application, there were several key requirements that shaped the development process. First, the QA application must be able to structure the enormous amounts of service entries and access them in a timely manner. To achieve this, the application utilizes a data access hierarchy that narrows down the amount of data queried. Second, the application must handle the situation that there is no available internet or cell signal in the immediate area. The application manages this problem by storing the changes locally and allows users to commit these changes when they reach an appropriate area. Finally, and most importantly, the application cannot function as a singular client-side application without any support. The QA application consists of a client and server with the client utilizing a Model-View-Controller(MVC) architectural pattern and the server utilizing a microservice architecture.

The client-side application utilizes a MVC architectural pattern as shown in Figure 3. This architectural pattern was executed while utilizing the Google Angular 3 framework alongside the C# and Javascript programming languages. The model section of the client-side application is the central nexus of interaction with the main NRDC System. This contains routines that would handle the HTTP communication with the server end, as well as the manipulation of locally stored data. The view section is tasked with the primary task of satisfying the first requirement and showcasing the data in a hierarchal format. In this section, the functionality to navigate through the hierarchy, view imagery, and handle conflicts syncing with the database is handled. Finally, the controller section serves the primary master and control portion of that application that dictates the behavioral actions that result from the interactions made by the user. This section is where the application would issue the command to shift the page views, create the transitional effects in-between views, and initiate the interaction with the model section.

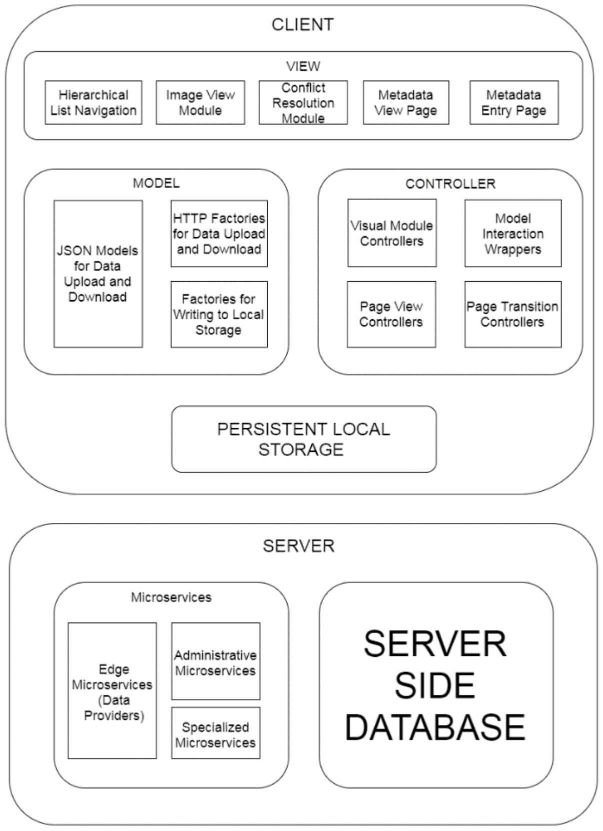

CLIENT
VIEW
Hierarchical List Navigation
Image View Module
Conflict Resolution Module
Metadata View Page
Metadata Entry Page
MODEL
JSON Models for Data Upload and Download
HTTP Factories for Data Upload and Download
Factories for Writing to Local Storage
CONTROLLER
Visual Module Controllers
Model Interaction Wrappers
Page View Controllers
Page Transition Controllers
PERSISTENT LOCAL STORAGE
SERVER
Microservices
Edge Microservices (Data Providers)
Administrative Microservices
Specialized Microservices
SERVER SIDE DATABASE

Figure 3: High-level block diagram detailing the architecture of the QA app. Client and server are connected via http calls from the mobile application to the Edge Microservices

The server side of the application utilizes a microservice architecture where each web service is independent of each other and are combined to create greater functionality. These services can each be classified into three main groups: Edge Microservices, Administrative Microservices, and Specialized Microservices. The Edge Microservices are infrastructure services that perform the role of data provider from the data base to the client application. The Administrative Microservices perform the role of security and provides varying level of access to the hierarchy, based on the individual's status within the project. The Specialized Microservices provide complex functionality to the application that are outside that of the Edge Microservices. These functionalities can include anywhere from photo compression and searching to file conversions.

4.2 UI Design

Throughout development, the User Interface design of the Quality Assurance application transformed drastically. Initial designs for a prototype implementation of this application, visible in Figure 4, were relatively simplistic, utilitarian and focused on the highest level of sensor network metadata that would be managed: a "project". A project described a specific cluster of in-situ sensors networked together that collected data associated with answering the question of a single research project. Given the broad scope of "a project," the metadata collected and stored was general in scope, requiring only one page and a few form fields. As development on this application expanded so too did the UI design to better accommodate the technical and personal needs of user stakeholders.

The first major design shift was a visual one. The application transitioned out of the simple Flat Design of the initial prototype into a Material Design variant with the addition of more primary color contrasts, subtler form fields and floating action buttons. This design choice gave the application a more modern and commercial feel that people have come to expect from a high quality mobile application.

A second design shift, more critical to the proper functioning of the application itself, was the inclusion of a hierarchical navigation structure. While the details of this structure are explored in Section 5, from a design standpoint it was crucial to enable the fast traversal of this hierarchy by making each item on every navigation level a large, clickable button that only leads to a sub list of items contained by the item clicked. Figure 5 contains an example of one of these sub lists. By placing a link to the information about the previously clicked item in the top right corner, the design of this list navigation negates the possibility of users unintentionally opening information pages that will slow their navigation. This streamlined design enables users to find the lowest level component they want in seconds, and then open information pages for the editing and viewing of related metadata.

Finally, at a late stage in development stakeholders expressed a strong need for the display of saved pictures showing the makeup of a sensor system or the structure of a specific component. This need drove the development of a dedicated image viewing component which retrieved images from a remote or local source and displayed them on appropriate pages. The addition of this component required significant planning as each image had many management functions associated with it: saving the image to the database, saving the image locally, delete locally, delete on the database, open an enlarged view of

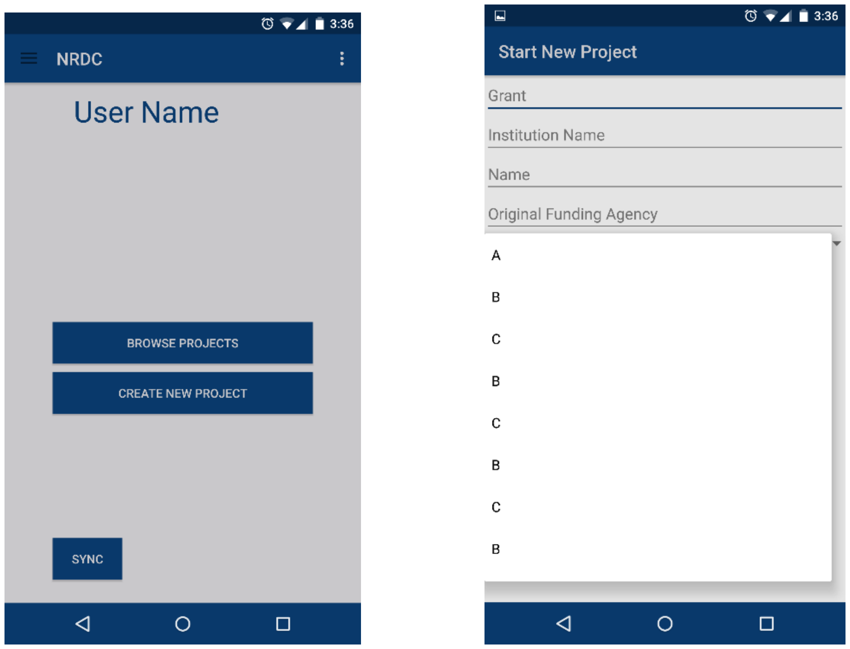


Figure 4: Screenshots of the initial UI design for the NRDC QA application. Left, the main screen shows the basic functionality implemented in the prototype. From here users can browse projects or create a new project. Right, the "Start New Project" interface was used to input relevant metadata about the sensor research project being documented

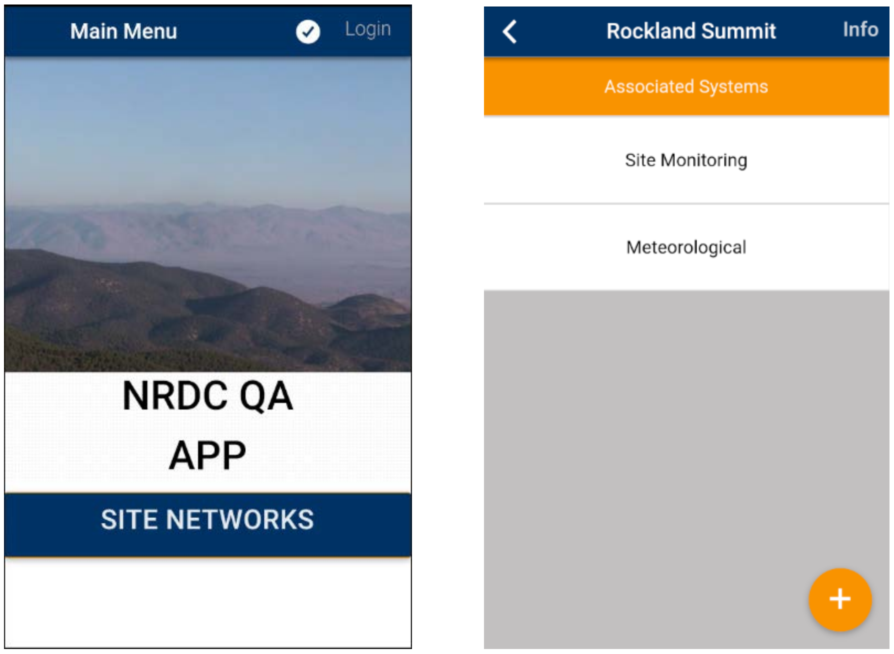


Figure 5: Screenshots indicating the final UI design of the Quality Assurance Application. On the left is an updated and ascetically pleasing main menu screen. On the right, we see an example of a sub-list in the hierarchal navigation structure. Here we see that "Site Monitoring" and "Meteorological" are systems associated with the "Rockland Summit Site". Clicking on either item will populate a new list of "Deployments" that are in the chosen system

the image, and more. This suite of functionality had to be added without cluttering the limited real-estate of a mobile screen. We overcame this problem with the inclusion of a floating action button which expanded into an array of buttons that each perform one of the above stated functions.

Finally, the interface provides users with a simple straightforward medium to quickly access to the forms required by technicians. Visually reminiscent of Google's material design, this application takes cues from public facing software to provide a refined interface to encourage smoother adoption of this app among unskilled mobile technology users. Large buttons and clickable lists simplify use for technicians wearing gloves when performing maintenance on sites in high elevations or in colder months.

5 Implementation

Currently, the NRDC QA application is comprised of a front-end system developed in the Ionic Framework, and a back end comprised of essential and independent web services communicating through a centralized hub [10]. The data transferred between the two are stored inside a Microsoft Virtual Environment with Microsoft SQL Server 2012 as the primary database management. These two main components together allow for a seamless interface between the main databases and the client application.

For the front-end system, the QA application was built with the Ionic Framework [8]. This framework utilizes HTML and CSS as a wrapper to manipulate the interface, as well as Javascript to apply functionality. Once completed, the HTML, CSS, and Javascript are then compiled into the appropriate codebase: either Apple or Android.

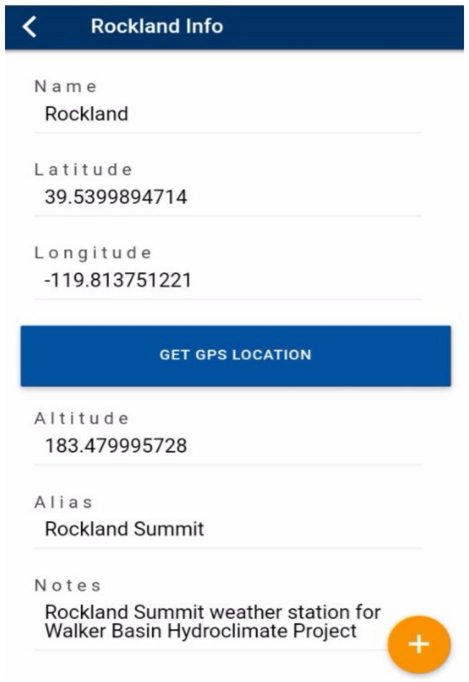
A wide collection of libraries and modules were used to simplify development at various stages of implementation. Primarily, Google's AngularJS was used as a structural framework for the Javascript codebase and allowed for a more object-oriented approach to manipulation of the HTML and interaction with microservice APIs [6]. Additionally, Node Packet Manager (NPM) and Bower were used as the main package managers for this application. They ensured libraries, assets, and utilities were regularly updated and organized.

Organization of the QA app follows a pattern representative of the hierarchical organization of existing sensor networks managed by the NRDC and associated institutions. At the topmost level, the application presents the user with a selection of Site Networks: a representation of several data collection sites connected by their similarity of purpose or project associations. From there the user selects a site associated with that network, a system associated with that site, and a deployment associated with that system, ending with a hardware component associated with that deployment. This workflow of "tunneling" down into atomic components gives data scientists a logical means of quickly locating the exact sensor network element they seek by leveraging their knowledge of existing infrastructure.

At any point in the navigation of the sensor network hierarchy, a user can add a new entry to the list of displayed entries or view the details of existing ones. Whether choosing to display or create, the user will be greeted with the same page. If displaying data about an existing item, the page will be populated with data about the selected element. If the user chooses to create a new item, the form fields on this page are blank and ready for input.

On the element creation screen, visible in Figure 2, the user inputs information into blank form fields that expand or contract to fit the size of the input data. The user can also choose to upload a related picture or get their location via their phone's GPS. This functionality enables field technicians to upload accurate location data about sites they are working on with the touch of a button. Once all necessary information is entered, the new metadata entry can be saved locally. And, once the user is done adding new entries, they can upload them en-masse to the server for storage in the database.

On the view screen the user is presented with a few different options compared to the creation screen. Principally she can no longer save an entry, only edit or delete with proper permissions, and there appears a floating orange icon in the bottom right corner visible in Figure 6. From the submenu which this button populates, users can add two different types of metadata about an entry, a document and a service entry. Documents allow users to add related files to a metadata entry. Service Entries are entered when scientific equipment is repaired or replaced.


Rockland Info
Name
Rockland
Latitude
39.5399894714
Longitude
-119.813751221

Altitude
183.479995728
Alias
Rockland Summit
Notes
Rockland Summit weather station for Walker Basin Hydroclimate Project

Figure 6: An example of retrieving latitude and longitude coordinates from the phone's GPS

6 Discussion

6.1 NRDC Impact

The successful implementation of the QA application changes the face of quality assurance in the field significantly for existing projects in the NRDC. The inclusion of a dedicated application impacts the workflow of sensor technicians and researchers by substantially augmenting current data management capabilities. Data stewards performing QA on sensor networks benefit from this application in several ways over traditional methods: uniform data entry, centralized QA data storage with synchronization and a usable interface facilitating the utility of the above benefits.

The problem of uniform an accurate data entry naturally occurs in any system reliant upon human interaction as the primary interface between a means of measurement and the means of logging. This problem is further exacerbated when technicians are deployed to remote areas, often equipped with only a notebook. It can be very hard to meaningfully restrict metadata and service logging, as different people are going to include different data that they find relevant to a QA expedition. The use of form fields significantly normalizes data input by restricting users to only give information deemed necessary and sufficient to detail the quality assurance practices performed. An example of these forms can be seen in Figure 7. In the case of intrinsically non-structured data, the option to attach documents is provided. This enables a diversity of data input methods.

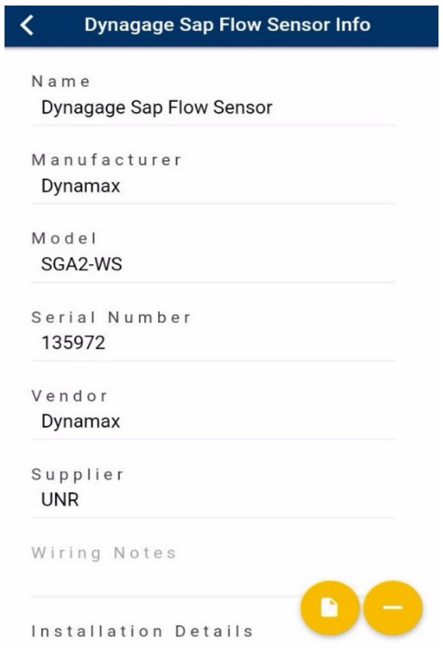

Dynagage Sap Flow Sensor Info
Name
Dynagage Sap Flow Sensor
Manufacturer
Dynamax
Model
SGA2-WS
Serial Number
135972
Vendor
Dynamax
Supplier
UNR
Wiring Notes
Installation Details

Figure 7: An example data entry page containing info about a single data sensor. The floating action button in the lower right-hand corner provides the option to add a service entry when clicked.

Previously, QA-relevant metadata collected and maintained by technicians was held in a decentralized heterogeneous collection of notebooks, spreadsheets and program comments. This proved problematic internally, as audits and reviews of quality assurance processes could not be effectively performed in a timely manner. Externally, this lack of centralized provenance-tracking metadata damages the integrity of collected data, as logs are not comprehensively tied to data streams. This limits the re-usability of collected data. With a central repository and dedicated backend infrastructure, the QA app significantly improves the maintenance of QA and metadata logs by providing a centralized, organized database to store this information and bind it to existing data streams. With the frontend mobile component automatically syncing with remote servers, users need not worry about any logistics of storing and formatting QA data for future use. They now only need to perform their normal maintenance and installation practices and fill out the form fields detailing their work.

6.2 Broader Impact

It has been well documented that an issue of paramount importance to the greater scientific community is the need for collected data to be accessible, findable, inter-operable and reusable. [9, 20-21] As scientific research has become increasingly collaborative with the growth of internet technologies, the free and easy distribution of data across the internet has not kept pace. [2, 17, 19] While there are many factors that contribute to this data need/accessibility gap, one prominent problem is a lack of identifying metadata accompanying datasets. [17, 21] That is the goal of this application and we believe that a need still exists in the wider scientific community for the easy creation and upload of identifying metadata, so a survey was created to demonstrate this.

To provide more significant validation and show the broader applicability of this software to the national scientific community a survey was conceived and solicited to Environmental Scientists. The survey was comprised of several questions that sought to evaluate how findable, accessible, interoperable, and reusable respondent's datasets were. Most questions were bound by a Likert scale of 1 to 5.

The pool of potential respondents came from two working groups – called "Clusters" -- for the collaborative organization Environmental Science Information Partners (ESIP). Specifically, members of the Envirosensing Cluster and the Documentation Cluster were emailed with the survey. After hosting the survey for one week 12 responses were acquired.

Using the responses collected from these domain experts we made some preliminary conclusions about how data scientists are managing their metadata. Primarily, we observed that there still exists a strong need for software like the Quality Assurance app to automate and speedup metadata documentation processes. Secondarily, we observed that although metadata was being digitized, it was not being digitized effectively.

Concerning the first conclusion, a few questions demonstrated the current gap in technology that limits effective metadata collection. When asked how managers document their metadata, the majority of respondents replied with either "Field Notebook/ Pen and Paper" and "Comments on Data Logger Scripts". When we juxtapose these responses against a later question that asks people to estimate the length of time it takes to get metadata into a machine-readable format, a picture emerges showing that scientists are collecting and digitizing this data but not in the most effective way. Upwards of six respondents indicated that it takes them "hours" and up to "weeks" to digitize their metadata.

When scientists are extensively using program comments and field journals to collect their data, they are not leveraging the advantages of mobile technology has to offer in the internet age. By comparison, the Quality Assurance app enables the fast and easy input of formatted data which is digital from the start and immediately uploaded to a central database. This application could save many data scientists and technicians hundreds of hours over the course of a year which are now wasted on basic data input.

With this survey and the extensive background works that clearly demonstrate a need to this sort of automated workflow, we have demonstrated that a strong need still exists for software like the QA app presented in this paper.

7 Future Work

Work on the Quality Assurance application will continue in the interest of expanding the present functionality detailed. First and foremost, further work will be done to help audits and administration of QA practices. Presently actions performed on the application are primarily user agnostic. They are performed with no considerations or limitations based upon the present user of the app. This can be problematic when it is necessary to track down the specific user who performed a given preventative maintenance on a given piece of equipment. A paper trail can prove immensely useful to any project on the scale of those which the NRDC helps maintain.

Outside of internal growth of the application itself, the data collected and maintained by the Quality Assurance application will be used to provide increased functionality to a companion quality control web application. QA and QC are often referred as nearly the same entity in discussions of data management. Where QA is concerned with ensuring that data streams have little opportunity to fail in their logging through constant maintenance and monitoring, QC is concerned with handling data that has been logged in error and attempting to correct those mistakes. The data stored via this QA app can be used to give context to any errors that might be discovered by an automated quality control service.

Acknowledgement

This material is based in part upon work supported by The National Science Foundation under grant number IIA-131726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Derik Barseghian, Ilkay Altintas, Matthew B. Jones, Daniel Crawl, Nathan Potter, James Gallagher, Peter Cornillon, Mark Schildhauer, Elizabeth T. Borer, Eric W. Seabloom, and Parviez R. Hosseini, "Workflows and

Extensions to the Kepler Scientific Workflow System to Support Environmental Sensor Data Access and Analysis," *Ecological Informatics* 5, DOI: http://dx.doi.org/10.1016/ j.ecoinf.2009.08.008, 1:42-50, 2010.

[2] Sean Bechhofer, David De Roure, Matthew Gamble, Carole Goble, and Iain Buchan, "Research Objects: Towards Exchange and Reuse of Digital Knowledge," *Nature Poceedings*, DOI: http://dx.doi.org/10.1038/ npre.2010.4626, February 2010.

[3] W. Bishop and T. H. Grubesic, "Metadata," *Geographic Information*, Springer Geography, Springer, Cham, 2016.

[4] John L. Campbell, Lindsey E. Rustand, John H. Porter, Jeffery R. Taylor, Ethan W. Dereszynski, James B. Shanley, Corinna Gries, Donald L. Henshaw, Mary E. Martin, and Wade M. Sheldon, "Quantity is Nothing Without Quality," *BioScience*, 63(7):574-585, 2013.

[5] ESIP, "Federation of Earth Science Information Partners," http://wiki.esipfed.org/index.php/Sensor_Data_Quality# Quality_Control_ .28QC.29_on_data_streams.

[6] Google, "Angularjs," https://angularjs.org/, Last accessed June 13, 2017.

[7] Sean Gordon, and Ted Habermann, "The Influence of Community Recommendations on Metadata Completeness," *Ecological Informatics* 43:38-51, 2018.

[8] Ionic, "Ionic," https://ionicframework.com/, Last accessed June 13, 2017.

[9] John Kratz and Carly Strasser, "Data Publication Consensus and Controversies," *F1000 Research*, DOI: http://dx.doi.org/10.12688/f1000research,3979.2, pp.1-21, October 2014.

[10] V. D. Le, M. M. Neff, R. V. Stewart, R. Kelley, E. Fritzinger, S. M. Dascalu, and F. C. Harris, "Microservice-Based Architecture for the NRDC," *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 1659-1664, July 2015.

[11] Michael J. McMahon, Frederick C. Harris, Sergiu M. Dascalu, and Scotty Strachan, "S.E.N.S.O.R. Applying Modern Software and Data Management Practices to Climate Research," 2011.

[12] William K. Michener, "Meta-Information Concepts for Ecological Data Management," *Ecological Informatics* 1, DOI: http://dx.doi.org/10.1016/ j.ecoinf.2005.08.004, 1:3-7, January 2006.

[13] William K. Michener, "Quality Assurance and Quality Control (QA/QC)," *Ecological Informatics*, Springer, Cham, pp. 55-70, 2018.

[14] NEXUS, "Solar Energy Water Nexus," https://solar nexus.epscorspo.nevada.edu/, Last accessed June 13, 2017.

[15] NRDC, "Nevada Research Data Center," http:// sensor.nevada.edu/NRDC/, Last accessed June 13, 2017.

[16] NSHE, "Epscor Nevada," https://epscorspo. nevada.edu, Last accessed June 13, 2017.

[17] Dominique G. Roche, Loeske E.B. Kruuk, Robert Lanfear, and Sandra A. Binning, "Public Data Archiving in Ecology and Evolution: How Well Are We Doing?" *PLOS Biology* 13, DOI: http://dx.doi.org/ 10.1371/jour nal.pbio.1002295, 11:1-12, November 2015.

[18] Carly A. Strasser and Stephanie E. Hampton, "The Fractured Lab Notebook: Undergraduates and Ecological Data Management Training in the United States," *Ecosphere* 3, 12:1-18, 2012.

[19] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame, "Data Sharing by Scientists: Practices and Perceptions," *PLoS ONE*6, DOI: http://dx.doi.org/10.1371/journal.pone.0021101, 6:1-21, June 2011.

[20] Michael C. Whitlock, "Data Archiving in Ecology and Evolution: Best Practices," *Trends in Ecology & Evolution* 26, DOI: http://dx.doi.org/10.1016/ j.tree.2010.11.006, 2:61–65, November 2011.

[21] Mark D. Wilkinson, Micheal Dumontier, IJsbrad Jan Aalbersberg, Gabrielle Appleton, Myles Axton, and Arie Baak, "The Fair Guiding Principles for Scientific Data Management and Stewardship," *Scientific Data*, 2016.

[22] Matthew Williams, Dan Cornford, Lucy Bastin, Richard Jones, and Stephen Parker, "Automatic Processing, Quality Assurance and Serving of Real-Time Weather Data," *Computers & Geosciences 37*, DOI: http://dx.doi.org/10.1016/j.cageo.2010.05.010, 3:353-362, March 2011.

Connor Scully-Allison received his B.A. in Philosophy in 2012 from the University of Nevada, Reno (UNR). Accepted into the master's program at UNR for Computer Science and Engineering in 2015, he is currently working as a research assistant on the Track 1 Nexus Project for the Cyber-Infrastructure lab located in the College of Engineering. His research interests include Human Computer Interaction, High Performance Computing, and Software Engineering. He has published two conference papers since 2016. As of June 2018, Connor holds a position as a student fellow for the Earth Science Information Partners (ESIP) Organization.

Hannah Muñoz graduated in 2016 with a B.S. in Computer Science from the University of Nevada, Reno. She started on her journey to her M.S in Computer Science with her advisers Dr. Sergiu Dascalu and Dr. Frederick Harris, Jr. in 2017. She hopes to finish her degree in 2018. Currently, Hannah works in the Cyber-infrastructure Lab where she helps develop applications that enable earth scientists to further their research.

Hannah has written and published two conference papers during her time as a Master's student. Her interest lies in mobile development and developing software to help in scientific analyses.

Vinh Le graduated from the University, Reno with a B.S in Computer Science and Engineering in 2015. Vinh is a Graduate Research Assistant affiliated with the Cyber Infrastructure Lab at the University of Nevada, Reno. He currently aims to earn a Master of Science in Computer Science and Engineering by 2018 and his research interests consist primarily of Software Engineering, Internet Architecture, and Human-Computer Interaction.

Scotty Strachan is the Director of Cyberinfrastructure in the Office of Information Technology at the University of Nevada, Reno. Strachan graduated from the University of Nevada, Reno in 2001 with a bachelor's degree in geography and minor in economics. After spending some additional years as a geotechnical consultant and project manager, he returned to the University and completed a M.S. and Ph.D., both in geography, along with a graduate minor in business administration. Strachan's primary research interests lie in mountain ecosystems and observational networks, and he relies heavily on the integration of information technologies with research to accomplish his goals of producing useful, long-term science.

Eric R. Fritzinger received his B.S. (2003) and M.S. (2006) from the University of Nevada, Reno. After spending several years in the field of medical robotics, he returned to UNR to participate in a state-wide project studying the effects of climate change in the Great Basin. He has worked on model and data interoperability as well as management and organization of environmental sensor data. He is currently the lead developer for the Nevada Research Data Center (NRDC), based out of UNR's Computer Science and Engineering Department.

Frederick C. Harris Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994, respectively.

He is currently a Professor in the Department of Computer Science and Engineering and the Director of the High-Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He has published more than 200 peer-reviewed journal and conference papers along with several book chapters. His research interests are in parallel computation, computational neuroscience, computer graphics, and virtual reality.

He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).

Sergiu Dascalu is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno, USA, which he joined in 2002. In 1982 he received a Master's degree in Automatic Control and Computers from the Polytechnic University of Bucharest, Romania and in 2001 a Ph.D. in Computer Science from Dalhousie University, Halifax, NS, Canada. His main research interests are in the areas of software engineering and human–computer interaction. He has published over 180 peer-reviewed papers and has been involved in numerous projects funded by industrial companies as well as federal agencies such as NSF, NASA, and ONR.