A Mobile Quality Assurance Application for the NRDC

Hannah Muñoz, Connor Scully-Allison, Vinh Le
Scotty Strachan, Frederick C. Harris, Jr. and Sergiu Dascalu
Department of Computer Science, University of Nevada, Reno
Reno, NV, 89509, USA
hannahmunoz, cscully-allison, vle@nevada.unr.edu
scotty@dayhike.net
fred.harris, dascalus@cse.unr.edu

Abstract

In this paper we present the design, implementation, and impacts of a cross-platform mobile application that facilitates the collection of metadata for grassroots-level sensor networks and provides tools for Quality Assurance processes on remote deployment sites. Created in close conjunction with scientists working on environmental sensor networks and data management experts, this paper details the software requirements, specifications, and implementation details required to construct such an application. In a discussion on how this software improves on existing techniques of logging contextual metadata and quality assurance information, it is shown 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.

keywords: 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 [6]. 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. This is the need for complete, accurate, and usable metadata.

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. Eventually, over time (or given enough distance) the value of these data are diminished to other researchers and the public. It becomes harder to recover and ascertain contextual information that is essential to decoding it. 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 [10].

This paper proposes a mobile application that manages and maintains quality assurance information 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.

In this paper, we describe the development and utility of a cross platform quality assurance application. This paper is structured as follows: Section 2 presents some related works to the app developed; Section 3 details the software specifications given for implementation; Section 4 discusses the software design and the use cases of the app; Section 5 discusses the implementation; Section 6 gives some discussion on the success of implementation and Section 7 contains our 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 [2]. Motivating work published on the subject comes from a 2013 paper "Quantity is Nothing without Quality: Automated QA/QC for Streaming Environmental Sensor Data [1]. 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. The QA platform implemented for the Nevada Research Data Center (NRDC) was designed using these fundamental requirements to fulfill these needs and the needs of a larger Quality Assurance/Quality Control system.

The NRDC is an Nevada-based organization dedicated to the, "storage, retrieval, and analysis of research data that is relevant to the needs and interests of the state of Nevada" [8]. Conceived as part of a NSF Track 1 project, the NRDC represents the collaborative efforts of top Nevada-based research institutions, including the University of Nevada Reno, the University of Nevada Las Vegas and the Desert Research Institute [7, 9]. 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 [6]. 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 our data management expert. The functional requirements detailed in Table 1, informed the core functional elements of the QA application. Using an agile

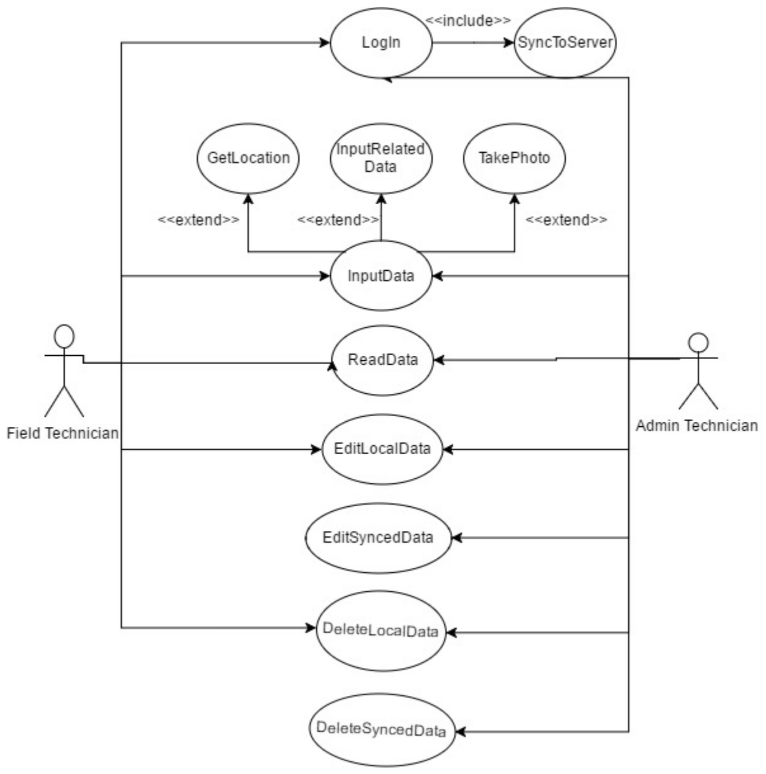


Figure 1: The Use Case diagram of the QA App

development method, these requirements went through several iterations before settling into the current list.

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

The nonfunctional requirements detail the needs of the system to be multiplatform 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.

4 Software Design

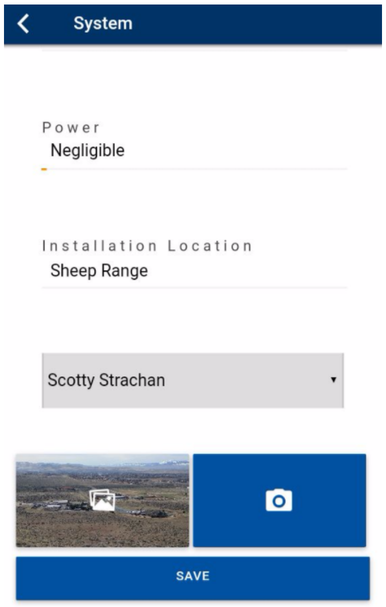
Using the functional requirements detailed in Section 3, a series of use cases were constructed and mapped in a use case diagram, found in Figure 1. This process informed the principal design phase of this applications construction and was frequently referenced or tweaked alongside the software specifications 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 unnecessary data downloaded. Technicians can also add new entries and upload them to the server. Admin 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.
- 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, such as the Site in Figure 3. 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
  Admin 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.

- DeleteLocalData
  Users are allowed to 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
  Admin 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.



Power
Negligible

Installation Location
Sheep Range

Scotty Strachan

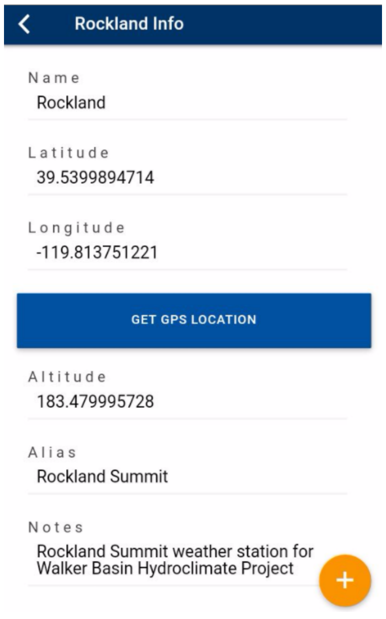
Figure 2: An example of inputting data with a picture from the phone's camera.

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 [5]. 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 completed with the Ionic Framework [4]. This framework utilized HTML and CSS as a wrapper to manipulate mobile elements of the interface, as well as Javascript to apply functionality. Once completed, the HTML, CSS, and Javascript are then compiled into the appropriate codebase. In addition to the actual languages, a wide collection of libraries and modules were used in various development stages of the application. Primarily, Googles 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[3]. 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 top 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



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 3: An example of retrieving latitude and longitude coordinates from the phone's GPS.

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 narrowing down 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.

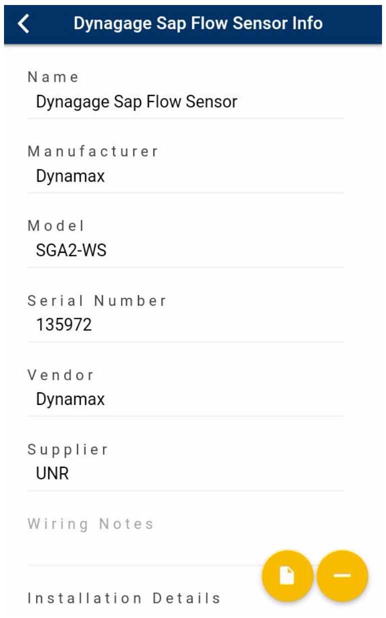
On the element creation screen, visible in Figure 2, the user inputs information into blank form fields that expand or contract to fit the size of the input data. The user can also choose to upload a related picture or get their location via their phone's GPS. This functionality enables field technicians to upload accurate location data about sites they are working on with the touch of a button. Once all necessary information in 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 3. 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.

6 Discussion

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 4. In the case of non-structured data, the option to attach documents is provided. This enables a diversity of data input methods.



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

Figure 4: 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.

Originally QA data 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 QA data 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. Users now only need to perform expected maintenance and installation and fill out the form fields detailing their work.

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.

7 Future Work

Work on the Quality Assurance Application will continue in the interest of expanding the present functionality detailed. First and foremost, with multiple technicians using this application at the same time its possible that different notes and modifications may be made to the same metadata entry. The user may not want work stored locally on their phone to be overwritten with a synchronization with the central database. A merge conflict option should be given to the user. Accordingly, it's of paramount importance that this functionality get added to the application as its adoption increases and the potential for such conflicts increases.

In addition to handling data conflicts, 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 a automated quality control service.

8 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] 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):574585, 2013.

[2] ESIP. Federation of earth science information partners. `http://wiki.esipfed.org/index.php/Sensor_Data_Quality#Quality_Control_.28QC.29_on_data_streams`.

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

[4] Ionic. Ionic. `https://ionicframework.com/`. Last accessed June 13, 2017.

[5] 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. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1659–1664, July 2015.

[6] 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.

[7] NEXUS. Solar energy water nexus. `https://solarnexus.epscorspo.nevada.edu/`. Last accessed June 13, 2017.

[8] NRDC. Nevada research data center. `http://sensor.nevada.edu/NRDC/`. Last accessed June 13, 2017.

[9] NSHE. Epscor nevada. `https://epscorspo.nevada.edu`. Last accessed June 13, 2017.

[10] Mark D. Wilkinson, Micheal Dumontier, IJsbrad Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, and et. al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 2016.