Computer Science and Engineering
University of Nevada, Reno

NRDC Quality Assurance Application

Team 9:
Brianna Blain-Castelli
Christopher Eichstedt
Matthew Johnson
Nicholas Jordy

Instructors:
Dr. Sergiu Dascalu
Devrin Lee

External Advisors:
Connor Scully-Allison
Vinh Le

2/22/19

1. Table of Contents

2. Abstract

The NRDC QA Application is a cross-platform mobile and web application. It is designed as a generalized, user-facing tool for recording metadata, but with a more specific goal of aiding NEXUS site technicians for better efficiency in the field. After a technician has stored their findings, a real-time representation of the data is recorded by saving the information to an offline device until an online connection is made. All information is stored using a database connection to send and receive metadata, that is then displayed using dynamic hierarchical navigation. The current process has NEXUS site technicians physically recording data in notebooks before transferring it to a central database. This document outlines the changes and updates to the specifications and design.

3. Recent Project Changes

There are no recent changes that have been made.

4. Updated Specification

4.1 Summary of changes in project specification

Some functional requirements have been reorganized and have had their tier changed according to the updated technical requirements specification. This is due in part to time restrictions, as well as coordination of core features that are necessary for deployment.

4.2 Updated technical requirements specification

Table 1. Updated Functional Requirements for the NRDC QA Application.

Functional Requirements		
Requirement ID	**Level**	**Description**
FR1	1	The application shall use an ontology to populate the data fields.
FR2	1	The application shall allow the user to interact with the populated data fields.
FR3	1	The application shall allow the user to edit site information.
FR4	1	The application shall type check data entered into the data fields.
FR6	1	The application shall allow the user to view site information without editing
FR7	1	The application shall allow the user to navigate through nested hierarchical pages.
FR8	1	The application shall allow the user to store data for later submission when an internet connection is established.
FR9	1	The application shall allow the user to retrieve GPS information from the mobile device for use in data entry.
FR10	1	The application shall allow the user to upload images to the site.
FR11	1	The application shall query data from a flask service.
FR5	2	The application shall allow the user to login with a username and password.
FR13	2	The application shall require validation from the user before they upload data.
FR20	2	The application shall allow the user to manage conflicts of information
FR22	2	The application shall allow the user to upload and store document files.
FR12	3	The application shall notify the user when they can synchronize their data.
FR14	3	The application shall allow administrators to change administrator information.
FR15	3	The application shall allow the user to navigate using a search function.
FR16	3	The application will have multiple steps for data verification.
FR17	3	The application will highlight incorrect information for the user after checking that the data is correct.
FR18	3	The application shall be able to handle network timeouts.
FR19	3	The application shall display loading progress as it synchronizes with the database.
FR21	3	The application shall email the user when data is synchronized.

Table 2. Non-functional requirements for the NRDC QA Application

Non-Functional Requirements	
Requirement ID	Description
NFR1	The application shall be created using the Ionic Framework.
NFR2	The application's front-end user interface shall be coded in TypeScript.
NFR3	The application's back-end parsing of the ontology shall be coded in Python.
NFR4	The application shall use a simple, intuitive user interface.
NFR5	The application shall be compatible with both Android and iOS.
NFR6	The application shall have an alternative web page interface.
NFR7	The application shall be deployed using Cordova built into the. Ionic Framework.
NFR8	The application shall utilize RESTful APIs via FLASK services.
NFR9	The application shall read an ontology in RDFS/XML format.

Table 1 describes the functional requirements of the NRDC QA Application. Level 1 requirements are must have. Level 2 are should have, and Level 3 are could have or want to have. Table 2 describes the non-functional requirements of the application.

4.3 Updated Use Case Modelling

4.3.1 Updated Use Case Model

The following section outlines the high level requirements as seen from a user perspective. Figure 1 displays the requirements that an outside entity (the Sensor Technician) will interact with.

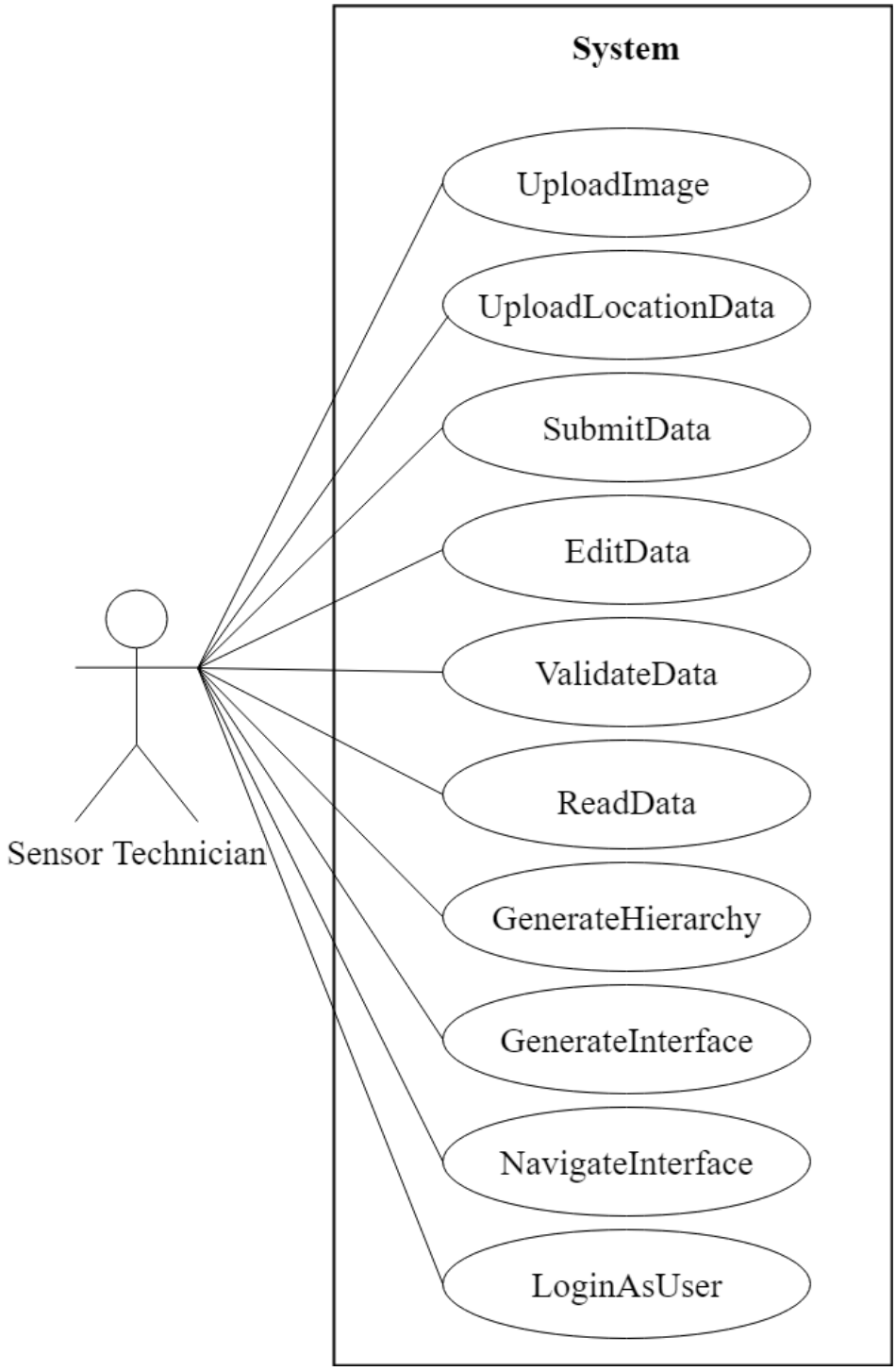

Fig1. The Use Case Diagram for the NRDC QA Application

Table ?. Use Cases for the NRDC QA Application

4.3.2 Updated Use Case Table

Table 3: The use cases for the NRDC QA application

6.1 Use Cases		
ID	Use Case	Description
UC01	UploadImage	The user shall be able to upload an image that will be displayed on the individual site screen.
UC02	UploadLocationData	The user shall be able to pull their GPS location and have this information immediately added to the necessary data fields.
UC03	SubmitData	The user can save data without internet connection and submit data when connection is made, by user request, following a verification process for the data and submission.
UC04	EditData	The user shall be able to edit data based upon user privileges.
UC05	ValidateData	The user shall enter data into the system which will then be checked that it meets specifications.
UC06	ReadData	The user shall be allowed to read site and sensor information.
UC07	GenerateHierarchy	The user will select an ontology to parse into an appropriate hierarchy that will be used by the application.
UC08	GenerateInterface	When a user opens a page the application will get the appropriate section of the hierarchy and dynamically generate and interface based upon the hierarchy.
UC09	NavigateInterface	The user shall be able to navigate an interface via a search for by navigating through a hierarchy from the ontology.
UC10	LoginAsUser	The user is required to enter their username and password upon starting that application before being granted permissions to view the rest of the application.

Table 3 describes the use cases for the NRDC QA application. Each case is an interaction that a user or that a part of the system has with the application.

4.3.3 Updated Traceability Matrix

Figure 5 displays the relationships between the use cases and functional requirements. The rows are the functional requirements and the columns are the use cases

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10
FR1										
FR2										
FR3										
FR4										
FR5										
FR6										
FR7										
FR8										
FR9										
FR10										
FR11										
FR12										
FR13										
FR14										
FR15										
FR16										
FR17										
FR18										
FR19										
FR20										
FR21										
FR22										

Fig 2. The Requirement Traceability Matrix

5. Updated Design

5.1 Summary of Changes in Project Design

User interface has seen a considerable number of changes as its design and logic continues to evolve. All decisions were at either advisement of the stakeholders for readability, or creative design choices. The included design is subject to change before final release.

5.2 Updated High-Level and Medium-Level Design

5.2.1 Updated High-Level Design

Figure ? details interaction that occurs between client and server. Indicated below is the detailed functionality regarding sending and receiving data from both the database and ontology service.

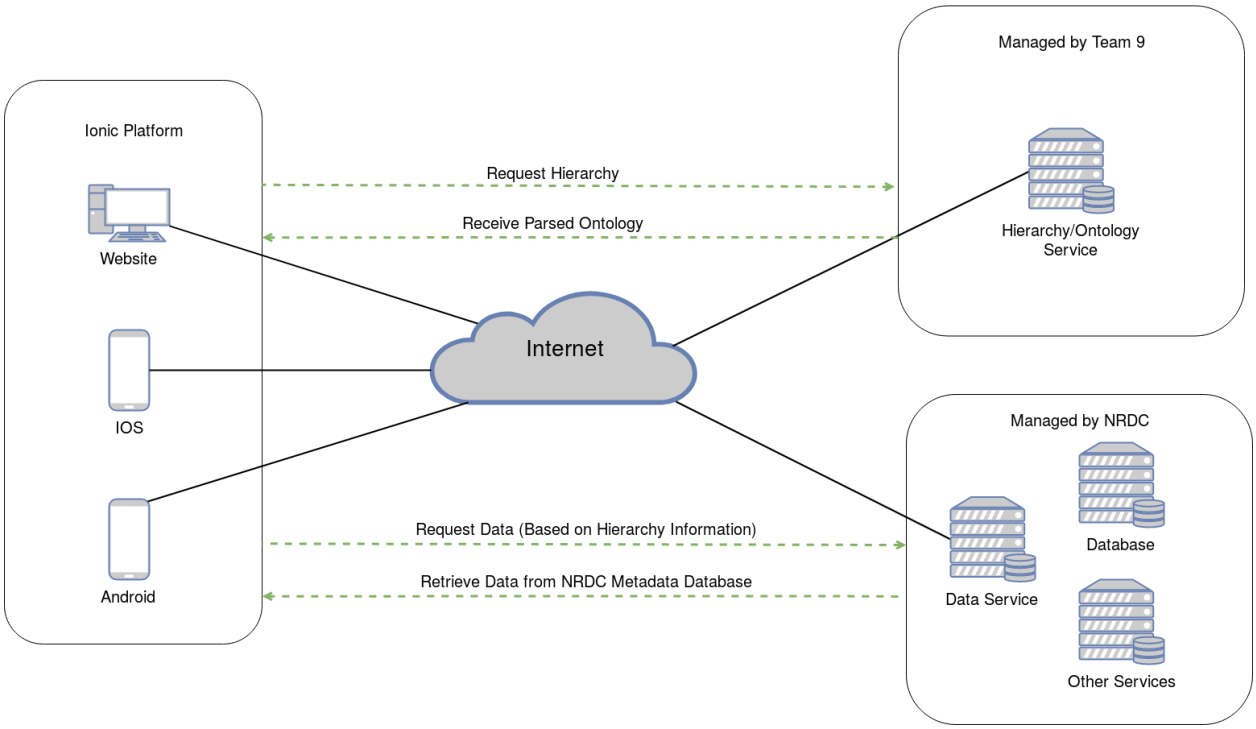

Fig 3: Diagram of of the architectural design of the application

5.2.2 Class Descriptions

The tables below contain the classes found within the NRDC QA Application. Each individual table contains functions that are described in greater detail as they pertain to the design.

Table 4. A description of the class "About"

Class: About	**Description:** An about page. It populates the necessary information about the development team and loaded project
getTeam()	This function populates the information regarding the development team.
getProject()	This function will populate information regarding the current project attached to the verified user.

Table 5. A description of the class “Home”

Class: Home	**Description:** The main page for the application. It displays notifications that updates have taken place and also displays the last site updated. It displays online status.
getNotifications()	This function will populate information on recent updates to the home page. It will display the number of updates that have taken place since the user last logged in.
getOnlineStatus()	This function will populate information on the current connection status. It will be denoted using a symbol for either “connected” or “disconnected.”
syncService()	A function called by a user pressing a synchronization button on the home screen. This function allows for the application to synchronize with the database
displayLoadingStatus()	This function visualizes the loading process on the home screen for the user by providing updates to the user on the loading process in the form of a progress bar with captions.
getOT()	Calls the flask service for a list of all organizational tiers and their parent-child relationships
getDataFields()	Calls the flask service for the characteristics of the tiers that will take the form of data fields in the application. This data is then stored in Characteristics.

Table 6. A description of the class “HierarchyManager”

Class: HierarchyManager	**Description:** Displays available hierarchical tiers according to the ontology assigned via login.
getUserNavigation()	Gathers hierarchical navigation data and populates the hierarchy manager page to display the appropriate level of navigation per user.
getLowerTiers()	This function gathers lower organizational tier information in order to structure pages following the broad location page.

Table 7. A description of the class “Search”

Class: Search	**Description:** Queries the application for appropriate search items defined by the user that can be keywords, locations or objects.
keywordSearch()	Takes the user submitted information and checks against database to populate interface with similar results.
locationSearch()	Similar to keywordSearch() but checks against the locations available to the user.

Table 7. A description of the class “SearchResults”

Class: SearchResults	**Description:** Populates a page based on items defined by the user that can be keywords, locations or objects.
returnKeywordResults()	Returns a populated list regarding information checked for when using the function keywordSearch() in the Search class.
returnLocationResults()	Returns a populated list regarding information checked for when using the function locationSearch() in the Search class.

Table 8. A description of the class “Login”

Class: Login	**Description:** Authenticates user login information and password to properly display information based on user credentials.
checkUser()	This function will check against the user’s login for validity.
checkPassword()	This function will check against the user’s password for validity.
encryptData()	This function will take in a string and encrypt it to using SHA256 encryption to ensure secure data transmission.

Table 9. A description of the class “Updates”

Class: Updates	**Description:** A page that displays most recent update logs and allows the user to check for updates.
checkOntologyUpdates()	Performs a checksum against the ontology to check for changes and modify information stored within the application accordingly.
checkVersionUpdates()	Performs a version check to search for new iterations of the application.
displayUpdates()	Display a list of information below the update option with changes made following the updates that took place. In the event no updates took place, an option stating no updates available is displayed instead.

Table 10. A description of the class “Settings”

Class: Settings	**Description:** A page that displays, holds and allows for the configuration of user settings, including accessibility, display, and more.
getUserSettings()	This function populates the user settings page with the current user settings.
updatePreferences()	This function updates the interface with user defined settings.

Table 11. A description of the class “DataValidation”

Class: DataValidation	**Description:** A class to hold validated data on the application. This class allows for validated user input to be saved if offline, and allows the user to send data when online.
validataData()	Checks the user entered information against the expected data types in the class Characteristics. This function copies over the data and deletes the information saved in Characteristics in order to allow the user to save validated data.
displayValidationErrors()	Returns which data was copied over successfully and which was not validated due to errors in the data. This information is populated on the page by highlighting successful fields in green and clearing them while highlighting unsuccessfully validated fields in red and keeping their information present in the field.
sendData()	If the user is online, sends the data to JsonConversion to send data to the database. If the user is offline, keeps the user entered data local.

Table 12. A description of the class “Parser”

Class: Parser	**Description:** Parses the ontology in order to determine data fields and hierarchical navigation. This class converts the parser results to JSON from XML and organizes this data to send to the interface.
convertOntology()	Converts the ontology XML file to JSON while parsing the data. The data is then stored in a graph to be organized.
sortOntology()	Sorts the ontology information into fields for easier reading. This includes gathering all organizational tiers, and linking characteristics to objects in the ontology such as sites or sensors.

Table 13. A description of the class “JsonConversion”

Class: JsonConversion	**Description:** A class to handle conversions of data types.
convertGPS()	Converts GPS coordinates to a format accepted by Json. This may include strings or integers.
convertImage()	Converts images to a format accepted by Json. This may include strings or integers.
convertTimeSeriesFromJson()	Converts a string provided in a query to date/time form for the application’s use
convertTimeSeriesToJson()	Converts time/date to a string accepted by Json.

Table 14. A description of the class “Characteristics”

Class: Characteristics	**Description:** A class to hold data fields as wells as data entries. This class organizes ontology information for the front end. It works in tandem with the DataValidation class.
gatherInput()	This function will accept user input and organize the data for later validation according to the corresponding data field.
generateFields()	This function will generate form fields based on the ontology assigned characteristics.

Figure 4 is a class diagram and describes the relationship that each class has with one another. In addition to the functions, the class diagram also displays the variables used by each class. The various diamonds and arrows notate the relationships the classes have with one another. The Parser class is not related to any other class because it derives from the back end while the rest of the classes derive from the front end.

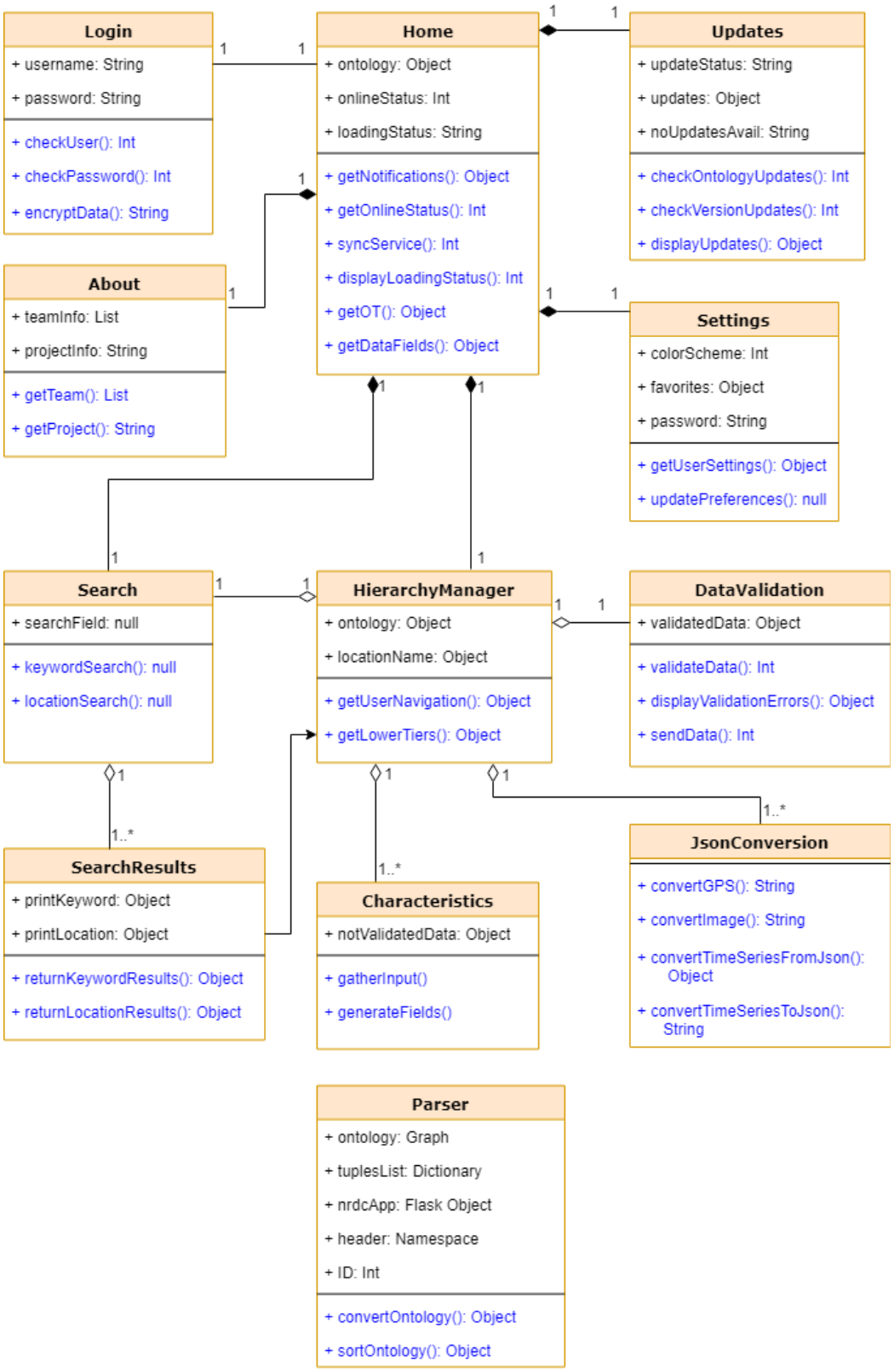

Fig 4. Class Diagram showing the relationship between critical classes in the application.

5.2.3 Data Structures

The main data structures utilized in the application's architecture include graphs, trees and a database. After parsing the ontology, the data gathered is stored within a graph. This graph is organized into a tree based upon relationships between the members of the graph and sent to the front end of the application through a series of queries. Metadata collected by the application is then stored within a database. A database table detailing this data type is displayed in Figure ?.

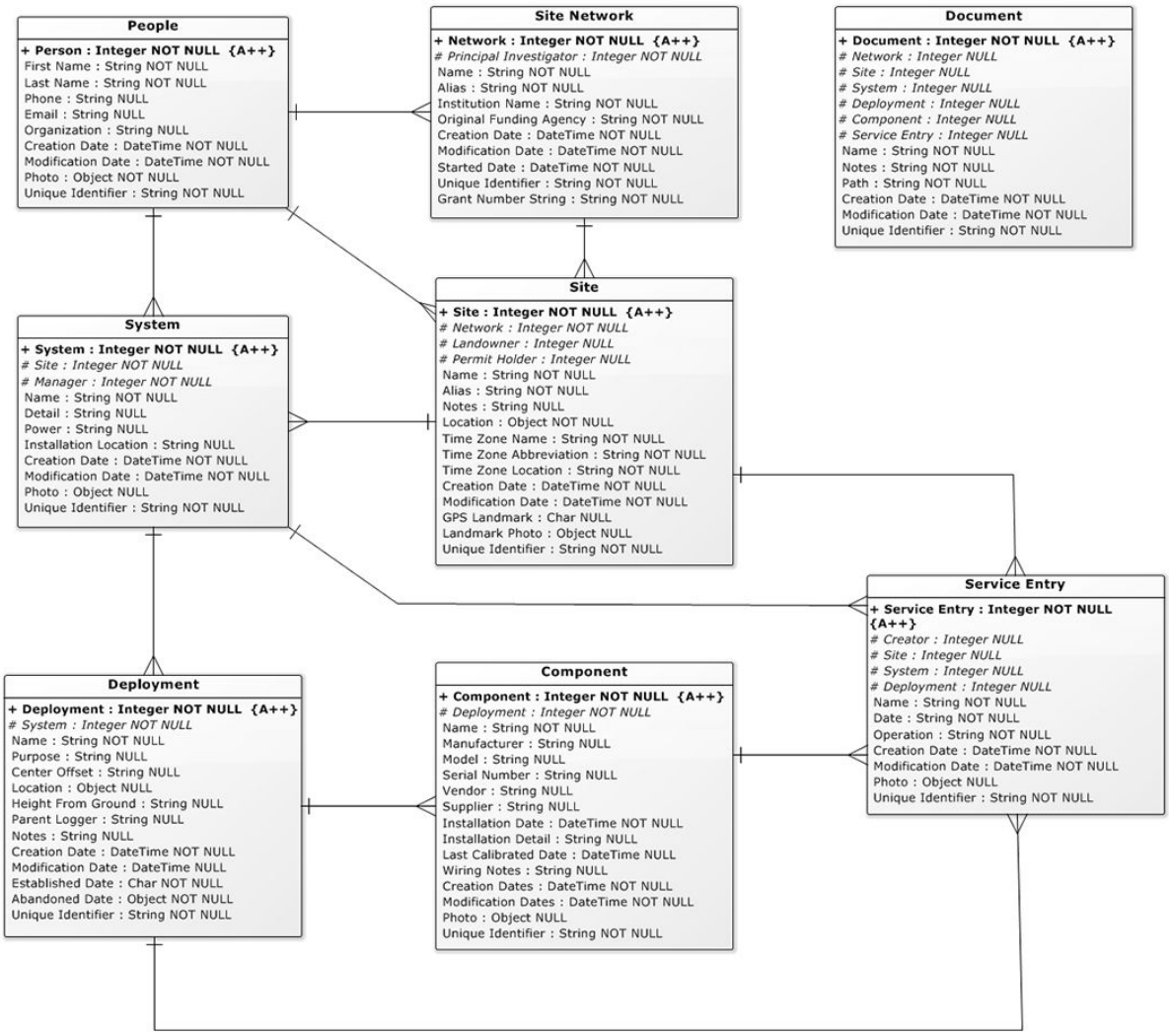

Fig 5. Database Table for the NRDC QA Application. Provided by Vinh Le and Connor Scully-Allison.

5.3 Updated Hardware Design

As there is no hardware component, the hardware design has not been changed.

5.4 Updated User Interface Design

Minor changes have been made to the NRDC QA application during the course of its development. Although it is currently presentable, these screenshots do not reflect the final design, as it is undergoing further changes.

Figures 6 and 7 show the login screen and home screen, which are the most prominent screens in the application. They are the first indication of the application's visual theme.

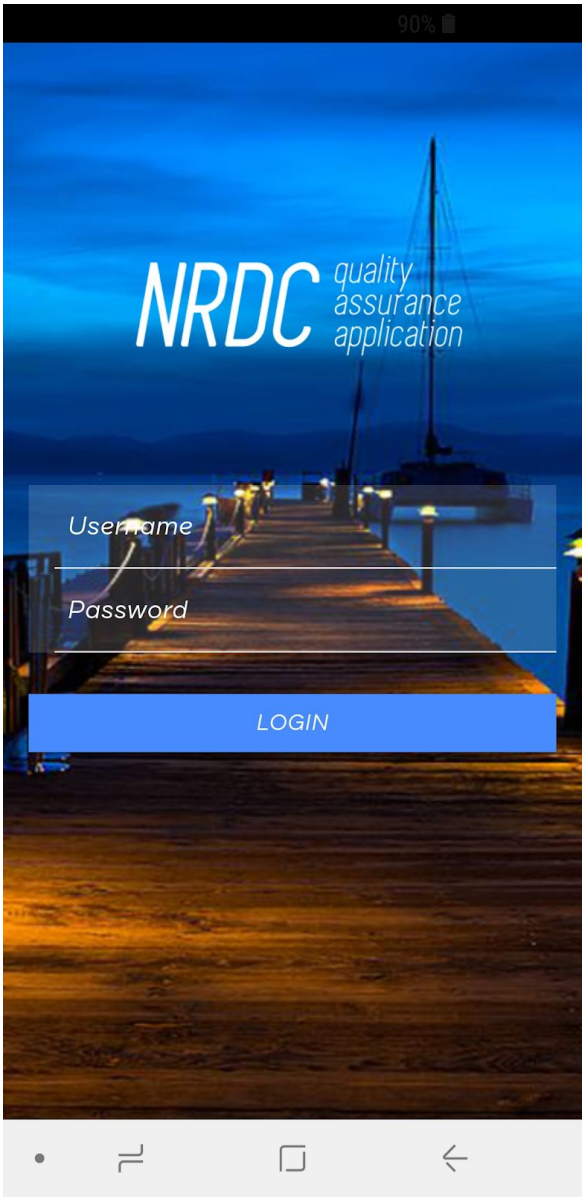

Fig 6. The login screen the user sees when they open the application.

Fig 7. The home screen the user sees after they login.

Figure 8 shows the hamburger menu which is the primary method of navigation in the application. Figure 9 presents a small screen capture from the about page linked from the hamburger menu.

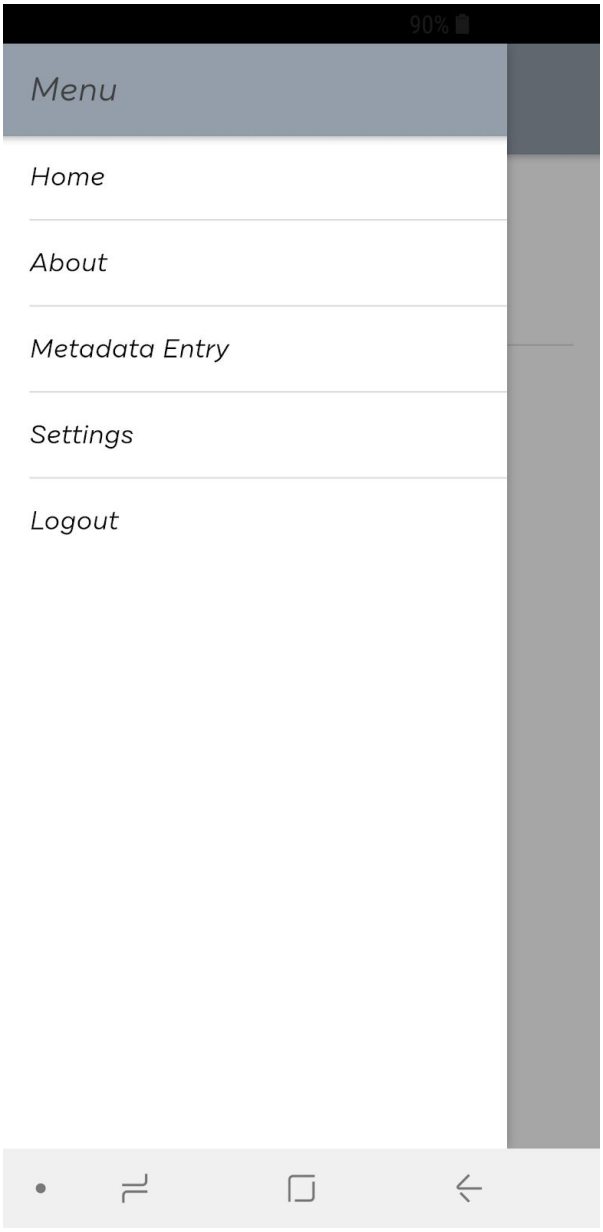

Fig 8. The “hamburger” menu that appears when the user presses the navigation menu button.

Fig 9. The about page that gives information about the people behind the application’s creation.

The main functionality of the application is seen in Figures 10 through 14. The screenshots show how the program displays the dynamic via the Metadata Entry page. As the user chooses items from the menus, the page then shows the sub-items of what was selected. More details for each selection are accessed easily by the edit link at the bottom of the list. Figure 15 is the current version of the settings screen, where the manually toggling between offline and online modes is possible.

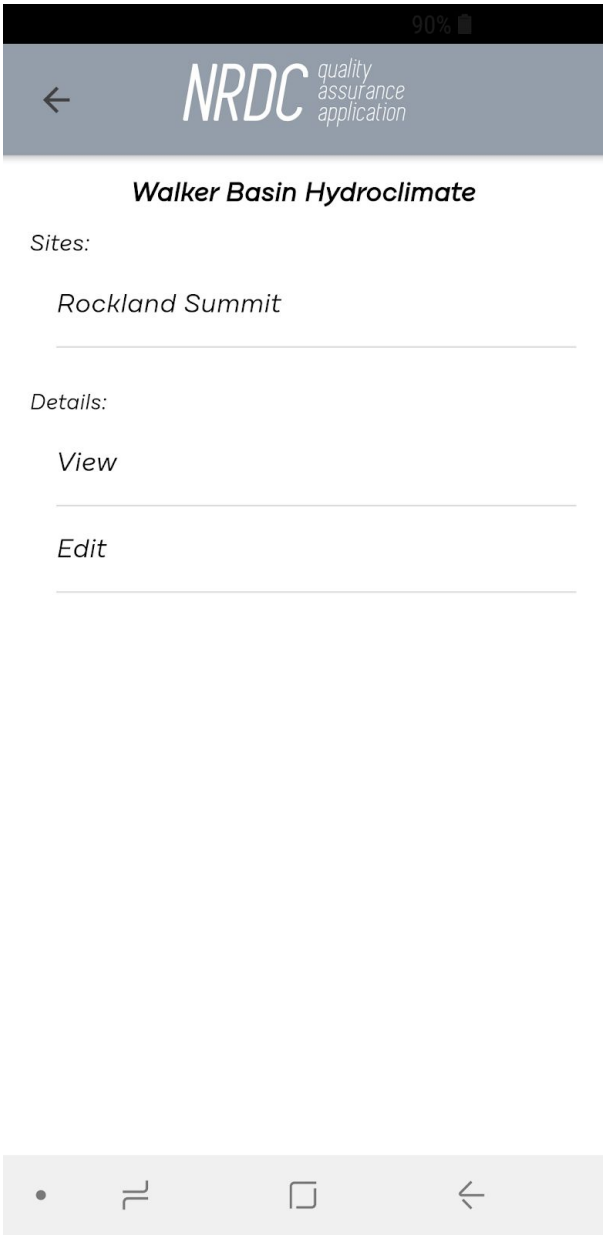

Fig 10. The hierarchical navigation display created from an ontology.

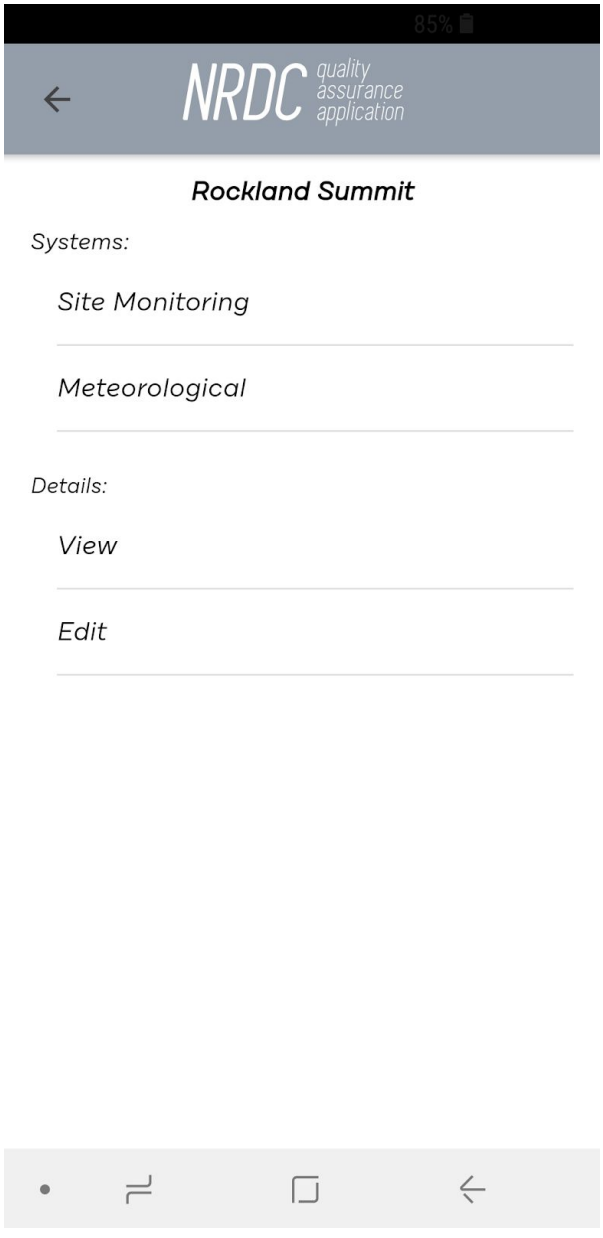

Fig 11. Another navigation display showing how application traverses and filters the data based on the ontology.

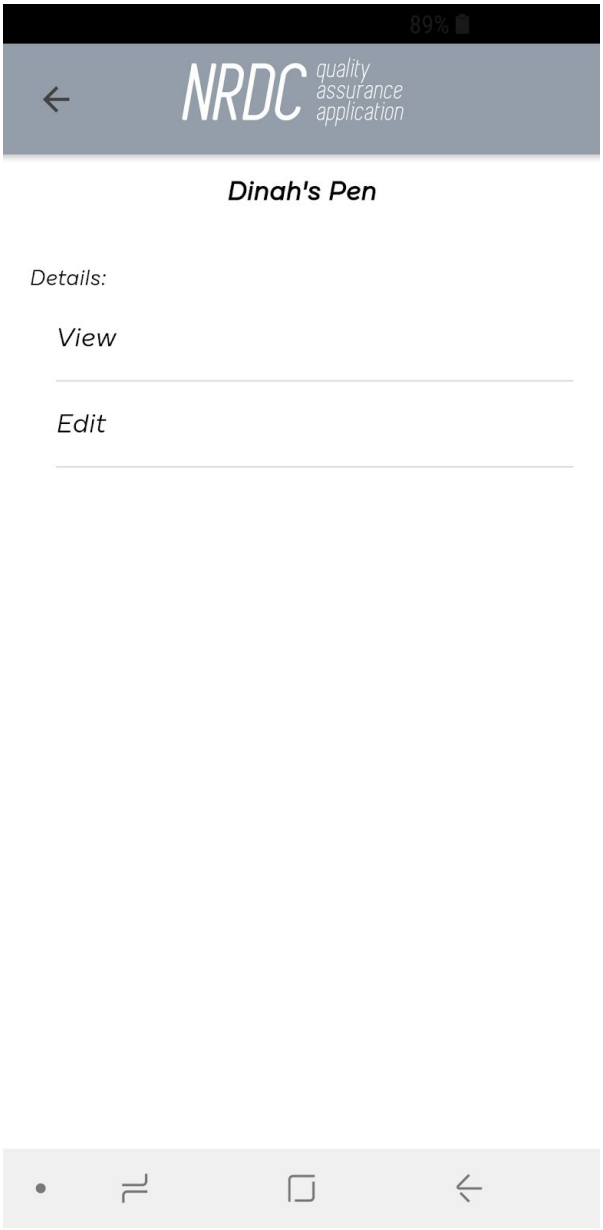

Fig 12. The view of the a bottom level item in the hierarchy (a “component”).

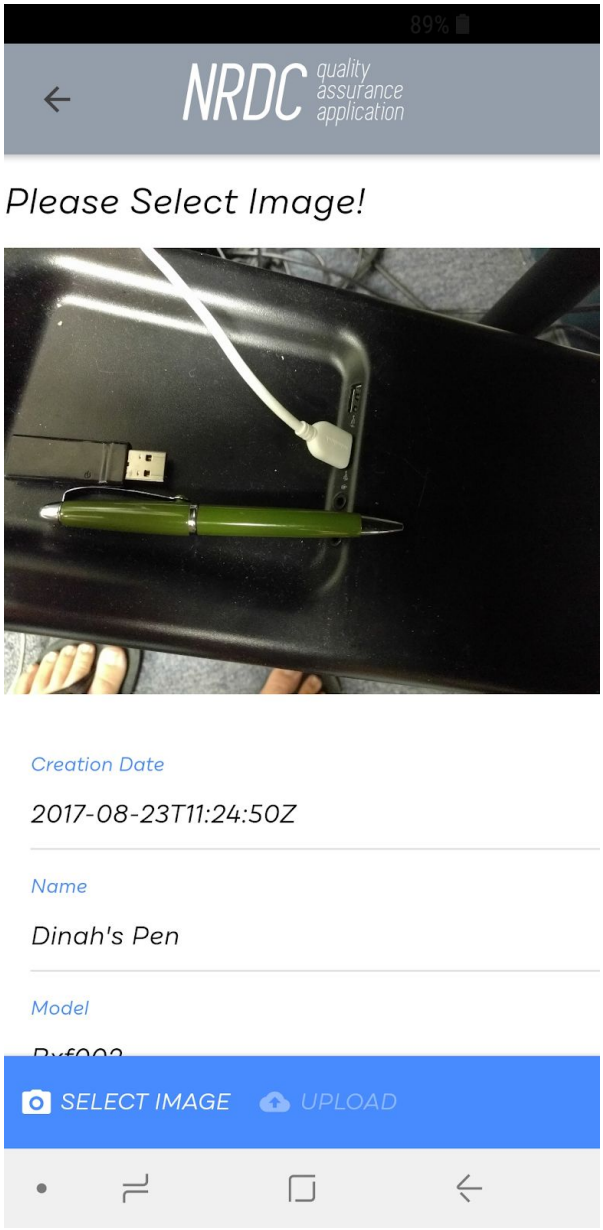

Fig 13. The edit view showing details of the component that is loaded from a database, including a photo.

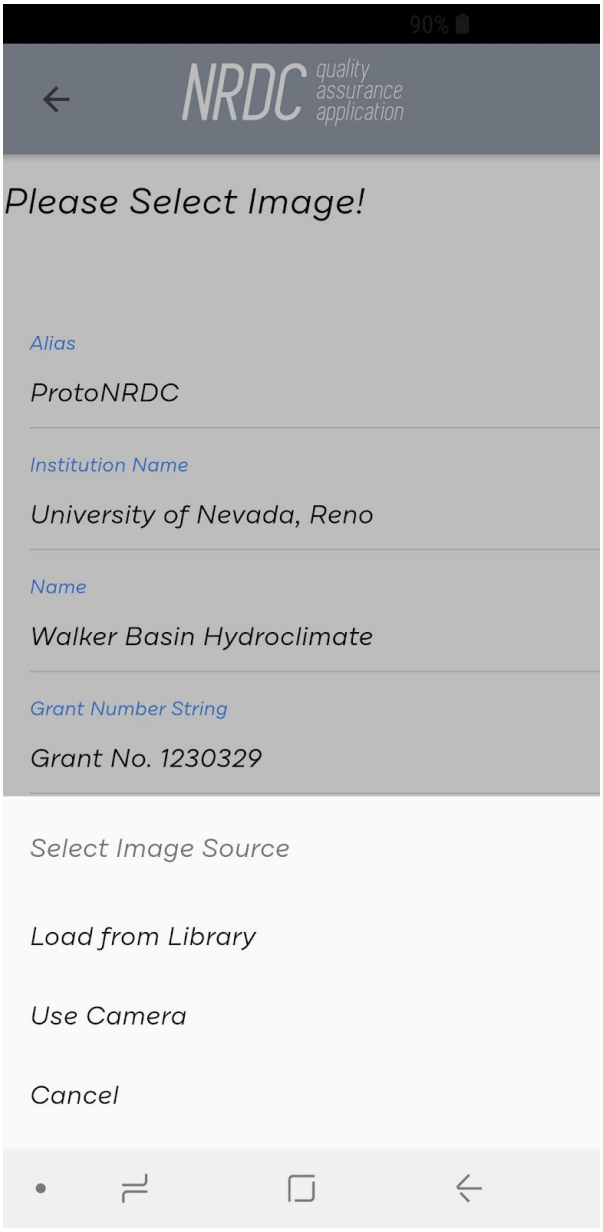

Fig 14. A menu showing interaction settings a user can use to add an image to the data.

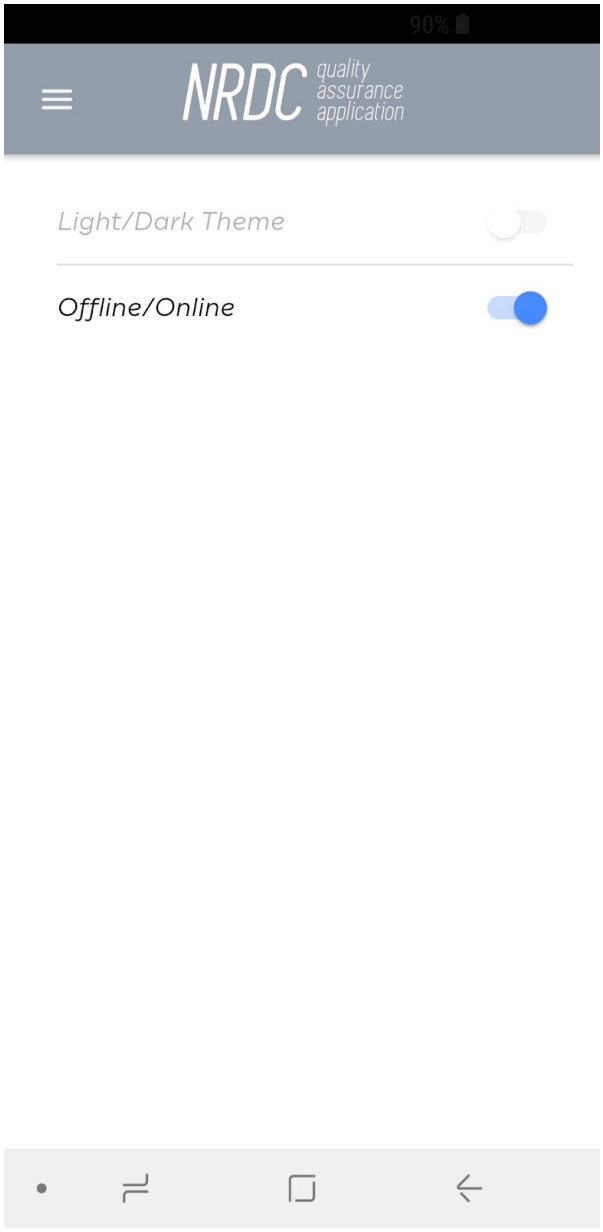

Fig 15. The settings screen that has an offline/online setting and will allow a variety of user defined settings.

Below, figures 16 and 17 help demonstrate the application's cross-platform capabilities by demonstrating a browser view of the application as deployed on a website.

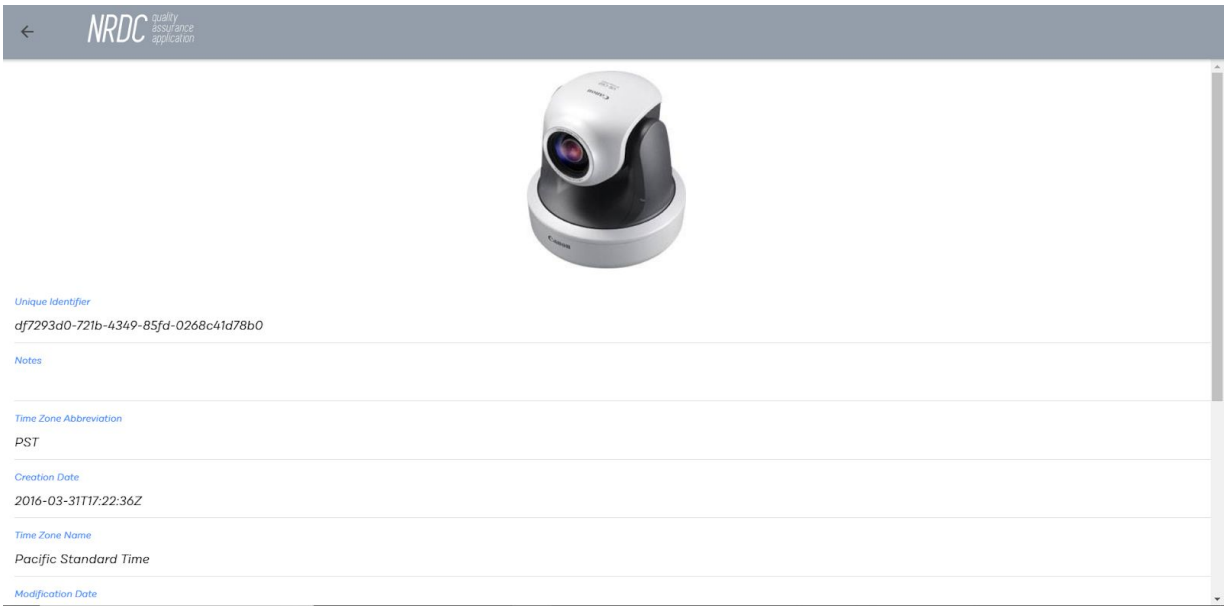

Fig 16. A website version of the details view from application shows its multi-platform capabilities.

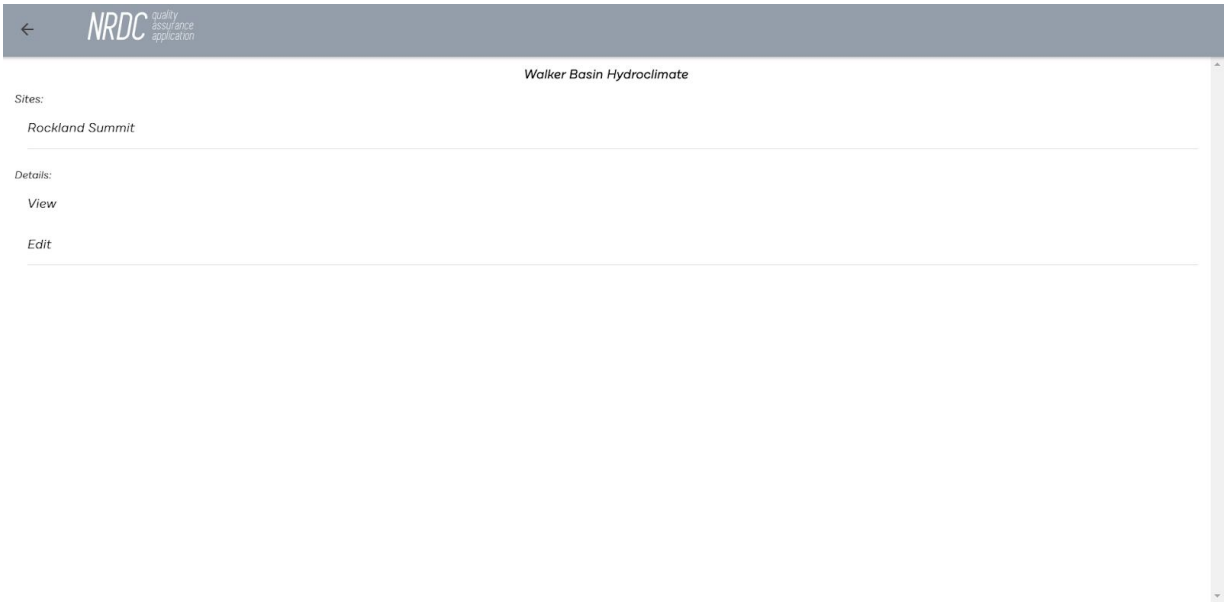

Fig 17. The hierarchy view as seen from the website build.

6. Glossary

1. Angular: An open-source framework for building web applications. It is based on Javascript.
2. API: Application program interface allows two software programs to communicate with one another.
3. Big Data: A large volume of structured or unstructured data that can be analyzed for patterns, trends, and associations.
4. Database: An organized collection of data designed to be easily accessed, managed and updated.
5. Discord: A chat room service and the main method of communication utilized during development.
6. Endpoint: A hardware device on a network designed to interconnect devices on the internet.
7. Environmental Science: The branch of biology concerned with the study of the natural world, including the conditions of the environment and the effects of these conditions on organisms.
8. Flask Service: A micro framework for Python used for managing website requests through HTTP with no data abstraction.
9. GPS: Global Positioning System used to find location data at any time of day in any weather conditions, using a radio navigation system.
10. Graph: An abstract data type used to define mathematical concepts and relations between objects through ordered pairs, nodes, and edges.
11. Hierarchical Navigation: The ability to navigate a website based on its organized structure that ranks items into levels of importance or relations.
12. In-situ Research: Researched performed on site, allowing data to be collected without altering test conditions and without isolating measurements taken from the effects of other systems. In-situ Research is also a service specializing in training and assisting in mixed-methods.
13. Ionic: A platform for building mobile, desktop and web applications.
14. Metadata: A dataset used to describe other data.
15. Microframework: A minimalistic web application framework that lacks most of the functionality expected from a web application.
16. Nevada EPSCoR: A part of the Nevada System of Higher Education concerned with multidisciplinary learning within science, research, education, and technology.
17. NEXUS: A project under Nevada EPSCoR concerned with the effects of Solar Energy on the environment of Nevada. This is the most recent project under Nevada EPSCoR and the affiliation of the main group of technicians that would be utilizing the application.
18. NRDC: A research center concerned with sensor-based data management, and improving research cyberinfrastructure.

19. Ontology: A standard for the abstraction of data models in order to define the categories, properties and relations between data and entities .
20. Parser: A program, usually part of a compiler or interpreter, that breaks data into smaller elements for data management.
21. RESTful: A web service based on stateless operation that allow the requesting device to access and manipulate web resources through a textual representation.
22. Server: A program designed to provide services to other computer programs and its users.
23. Trello: A project management service designed to allow managers to turn large iterations into tasks and mark them as not started, in progress, and completed.
24. Triple Query: A request for information or data composed of subject-predicate-object.
25. XML: A language to define rules to encode documents. The format is designed to be easily readable by both humans and machines.

7. Standards and Technologies

The table below describes the various technologies and standards used for development of the application. Included are their descriptions and their use in the project.

Table 15. A table of the standards and technologies used for development of the application

Name	Description	Type	Use in Project
Flask	Micro framework based in Python and others.	Technology	Web Service used to parse the ontology.
Ionic	Open source software development kit for cross platform applications.	Technology	Framework used to build application.
HTML5	Hypertext Markup Language for building websites.	Standard	Defines application pages.
CSS	Cascading Style Sheets used for presentation of a document in HTML.	Technology	Defines user interface of the application.
Typescript	Microsoft developed superset of JavaScript.	Technology	Defines the functionality of the application.
Python	General purpose programming language compiled using an interpreter.	Technology	Used in development of the back end Flask Web Service.
UML	Unified Modeling Language used to plan application development.	Standard	Used in past assignments to better visually develop system architecture.
NodeJS	JavaScript runtime environment allowing use without a browser.	Technology	Allows application to run outside of browser.

8. Updated List of References

8.1 Problem Domain Book

1. Singh, Indermohan, and Hoc Phan. *Ionic Cookbook: Recipes to Create Cutting-Edge, Real-Time Hybrid Mobile Apps with Ionic*. 3rd ed., Packt Publishing, 2018.

A book that has tutorials on how to create hybrid applications using Ionic, Angular, and Cordova. The tutorials include basic sections such as creating themes as well as more advanced sections such as using the REST API, lazy loading, and deep linking.

8.2 Project Reference Articles

1. Catenazzi, Nadia & Sommaruga, Lorenzo & Mazza, Riccardo. (2009). "User-Friendly Ontology Editing and Visualization Tools: The OWLeasyViz Approach." 283-288. 10.1109/IV.2009.34. www.researchgate.net/publication/221360225_User-Friendly_Ontology_Editing_and_Visualization_Tools_The_OWLeasyViz_Approach (Oct. 30, 2018).

A conference paper that explores different methods and tools used to visually display an ontology. Primarily, the paper explores the usability and functionality of several web browser plug-ins and a web-based tool before presenting its own conceptualization of a standalone application for visualizing ontologies.

2. McCarthy, J.L. "METADATA MANAGEMENT FOR LARGE STATISTICAL DATABASES." Lawrence Berkeley National Laboratory, 1 Mar. 1982, cloudfront.escholarship.org/dist/prd/content/qt5cc031cm/qt5cc031cm.pdf (Oct. 30, 2018).

A paper explaining the differences between data and metadata and detailing what metadata is, such as its types, uses, and characteristics. It also explains some of the requirements necessary to properly manage metadata. Specifically, it talks about the problems and proposed solutions to the metadata issues that have arisen with SEEDIS.

3. Scully-Allison, Connor, et. al. “Advancing Quality Assurance Through Metadata Management: Design and Development of a Mobile Application for the NRDC.” 1 Mar. 2018, www.cse.unr.edu/~fredh/papers/journal/61-aqatmmdadoamaftn/paper.pdf (Oct. 30, 2018).

A research paper that details the previous version of the application being developed. It details the previous versions specifications such as its use cases, its architectural design, and its UI design. It also describes the application’s general impact and impact for the NRDC.

4. McGuinness, Deborah L., and Frank Van Harmelen. "OWL web ontology language overview." *W3C recommendation* 10.10 (2004). www.researchgate.net/publication/200034408_OWL_Web_Ontology_Language---Overview (Feb. 21. 2019).

This paper outlines the design of the Web Ontology Language (OWL) and how one is built using the Resource Description Framework (RDF) Schema. Also in the paper is a description of what makes OWL useful as well as a detailed breakdown of all of the language features.

8.3 Project Related Websites

1. W3C. *OWL Web Ontology Language Parsing OWL in RDF/XML*, www.w3.org/TR/2004/NOTE-owl-parsing-20040121/ (Oct. 30, 2018).

A web standards organization’s web page outlining how one might construct an OWL ontology by parsing the triples in an RDF/XML file. It thoroughly details steps one needs to take to translate the structures presented in the RDF file as well as how to handle certain errors that are likely to arise.

2. Fredrich, Todd and Pearson Corp. “Learn REST: A RESTful Tutorial.” *REST API Tutorial*, www.restapitutorial.com/ (Oct. 30, 2018).

An educational website that explains what Representational State Transfer (REST) is and what is needed to know to design a REST API. It includes an introductory tutorial video, explanations of terms, and explains how to use RESTful web services using http.

3. Ionic. “Ionic Platform Documentation.” *Ionic Framework*, ionicframework.com/docs/ (Oct. 30, 2018).

The online documentation for the Ionic Framework that very thoroughly covers the many features of Ionic such as the API, themes, components, and its command line interface. It also gives code examples and provides links to outside resources that may be useful to developers.

4. NRDC “Nevada Research Data Center.” *NRDC,* sensor.nevada.edu/ (Nov. 2, 2018).

The Nevada Research Data Center website offers an overview of their data services and ongoing projects. The data services branch offers geospatial data, video streams, image archives and current weather conditions for listed locations. The ongoing projects branch redirects the user to the project’s respective website.

9. Contributions of Team Members

Brianna Blain-Castelli

- Cover Page
- Table of Contents
- Abstract
- Recent Project Changes
- Updated Specification
- Updated Design
- Glossary
- Standards of Technologies
- Updated Lists of References
- Total: 6.00 hours

Christopher Eichstedt:

- Cover Page
- Table of Contents
- Abstract
- Recent Project Changes
- Updated Specification
- Updated Design
- Glossary

- Standards of Technologies
- Updated Lists of References
- Total: 6.00 hours

Matthew Johnson:

- Cover Page
- Table of Contents
- Abstract
- Recent Project Changes
- Updated Specification
- Updated Design
- Glossary
- Standards of Technologies
- Updated Lists of References
- Total: 6.00 hours

Nicholas Jordy:

- Cover Page
- Table of Contents
- Abstract
- Recent Project Changes
- Updated Specification
- Updated Design
- Glossary
- Standards of Technologies
- Updated Lists of References
- Total: 6.00 hours