Computer Science and Engineering
University of Nevada, Reno

NRDC Quality Assurance Application

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

Instructors:
Dr. Sergiu Dascalu
Devrin Lee

External Advisors:
Connor Scully-Allison
Vinh Le

12/13/18

1. Table of Contents

2. Abstract

The NRDC QA Application is a cross-platform mobile and web application. It is being 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. The current process has NEXUS site technicians physically recording data in notebooks before transferring it into a central database. This document will breakdown the current working prototype and developments made since the design report. It will also contain the feedback given from the demonstration with project stakeholders and outline the future developments to be made. Additionally, snapshots from the working prototype and a current code line count will be included.

3. Introduction

The 2018-2019 Senior Project's Team 9 developed a prototype for a web-based Quality Assurance (QA) application for NEXUS affiliated technicians who visit remote research sites throughout Nevada. Currently. The only option available to technicians for recording QA information is using physical notebooks. Technicians are then required to manually enter data collected into a computer upon their return. This can lead to inaccurate data entry due to factors such as human or documentation error. The finished application will allow a technician to enter information regarding quality assurance into given data fields, regardless of a connection to the internet. When the application is connected it will synchronize the data recorded to a central server. This will make the entire process of recording data more efficient. Currently, the prototype does not have a working database connected.

The application uses an ontology which acts as a graph data structure that tells the application how to format its data fields. When the technician enters information and the application is not connected, the data is planned to be stored locally in future iterations. If the user is connected, the application tests for a network connection and is designed to provide a notification that it is ready to send information in the future.

The application uses the Ionic framework which is designed for creating mobile applications. Ionic uses multiple programming languages including TypeScript, HTML, and CSS. The *Ragnarok* server uses a parser to load a given ontology, and the application connects to a Flask Microservice which provides it with the hierarchy necessary to generate the interface.

Progress made since the design report includes front and back end communication. More specifically, the front end has the ability to request a JSON from the back end to dynamically populate data and navigate using a hierarchy. This can be toggled off and on, using the offline and online switch found on the Home view. Basic UI templating for future implementation has

been made, as well as multiple views that are apart of the working prototype. An Example page has also been added for better visualization of the finished product.

4. Prototype Objectives & Functionality

4.1 Basic UI template

The prototype uses the ionic menu template, or “hamburger” menu. A color scheme that would allow both light and dark themes was chosen. As of right now, the UI template design is subject to change.

4.2 Multiple View Design

For the prototype, we implemented five different pages that can be visited through a side menu. They include: Home, About, Settings, Hierarchy Navigation, and Example pages.

- Home contains a simple welcome message and includes a setting for offline/online functionality.
- About has the details of each team member, as well as a template to add information about the instructors and advisors.
- Settings for the purpose of the demonstration had no functionality, and was only present to show future possibilities.
- Hierarchy Navigation is a “proof of concept” for the main functionality of the application, which is reading and parsing an ontology to populate data dynamically.
- Example was used for the demonstration to show the potential layout of a site page.

4.3 Front and Back End Communication

The prototype is capable of communicating between the front and back end. This means the front end makes a request for the ontology, the back end parses the XML RDF ontology and returns a JSON. It is then dynamically populated in a view on the front end. Currently, there is no ability to write and store data. Figure 1 demonstrates the process for better clarity.

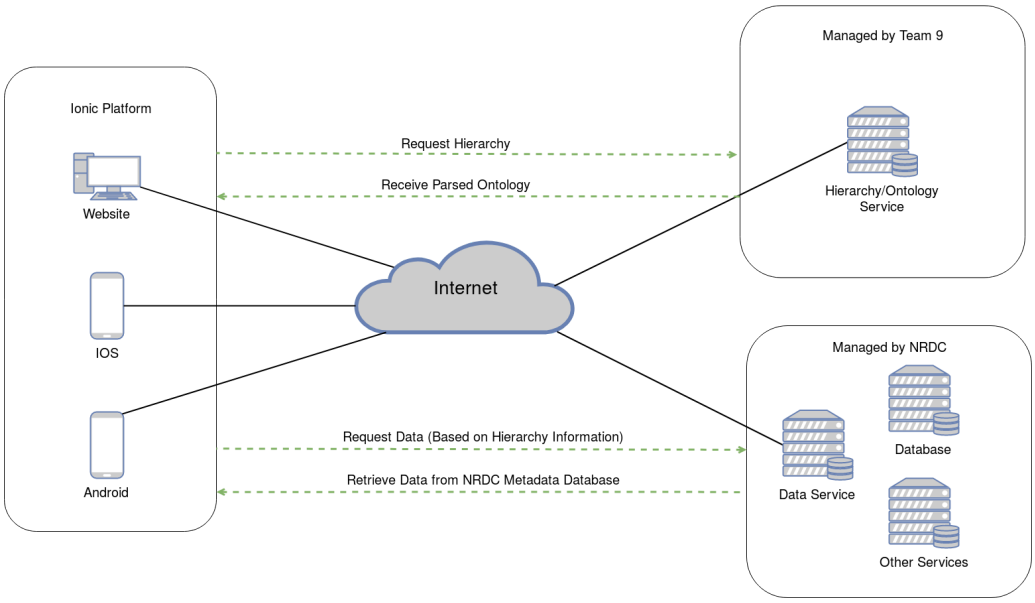

Fig 1. A diagram describing the communication between the front and back ends of the prototype.

4.4 Dynamic Population of Assets

The Hierarchy Navigation is able to create data fields dynamically from a parsed ontology given to the application, or read from a server. This makes it efficient as the prototype only requires one page instead of making individual pages for each part. The ontology is contained inside of a JSON file that is sent from the back end.

4.5 Hierarchy Navigation of Ontology

The ontology used by the prototype is set up in a hierarchical manner. The prototype is able to traverse the hierarchy given in the proper order, from top to bottom.

4.6 Parsing JSON for Data

The front end of the prototype is able to read in and parse a given JSON file from the back end to create the hierarchy navigation and populate data dynamically.

4.7 Example of a Site Page

The example was designed to help better illustrate what the dynamically populated site pages would eventually become. Currently, it is statically implemented and is subject to change with further development.

4.8 Online/Offline Functionality

The Home page of the prototype includes a toggle switch to simulate offline/online functionality. When offline, the prototype can create the hierarchy, but cannot populate the data. When online, it can populate the data as well.

5. Develop Prototype

The prototype for the NRDC QA application was developed using multiple programming languages. The front end user interface was built using the Ionic framework, which uses TypeScript for its functionality, and uses HTML and CSS for its visuals. The back end was developed using Python, specifically using Flask, a web micro service. There was extensive amount of research done to properly prepare for these new languages and the ontologies that were handled. The team was previously only fluent in HTML, CSS, C, C++ and some C#. It should also be noted that both TypeScript and Python are efficient languages that focus on using the least amount of code lines to complete multiple tasks. The application also uses an ontology. The purpose of using an ontology in the application is to make it dynamic, which inherently reduces the lines of code necessary to create working pages. This is in comparison to a static page which require a lot more hard coding and redundant code to achieve a similar level of functionality.

Table 1 is a breakdown of the the code used in the front end development of the application. The total number of lines of code was 655, excluding comments and blank lines. Table 2 is a breakdown of the total number of lines of code used by the back end, which was 649, excluding comments and blank lines.

Table 1. The breakdown for the lines of code of the Front End of the prototype.

Languages	Files	Blank Lines	Comment Lines	Working Code Lines
TypeScript	17	77	33	380
HTML	9	70	26	188
CSS	9	42	45	56
JavaScript	1	6	8	17
JSON	2	0	0	14
Total	38	195	112	655

Table 2. The breakdown for the lines of code of the Back End of the prototype.

Languages	Files	Blank Lines	Comment Lines	Working Code Lines
XML	1	110	45	464
Python	2	39	93	181
Markdown	1	1	0	4
Total	4	150	138	649

Figures 2-9 are snapshots of the current prototype User Interface. Figures 2 and 3, the Login and Home views, are are shown to show the initial views that will be seen by a user. Figures 4 and 5 are the “hamburger” menu and Settings views which show the straightforward navigation through the main sections of the app, and the settings which can influence how the app functions.

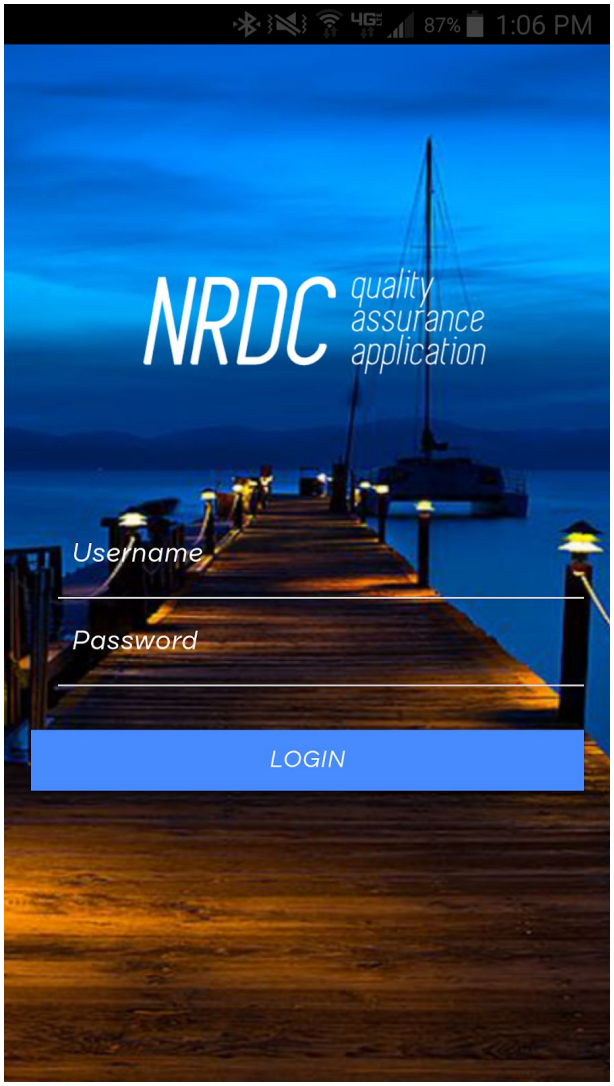

Fig 2. The login view the user will encounter when they first start the application.

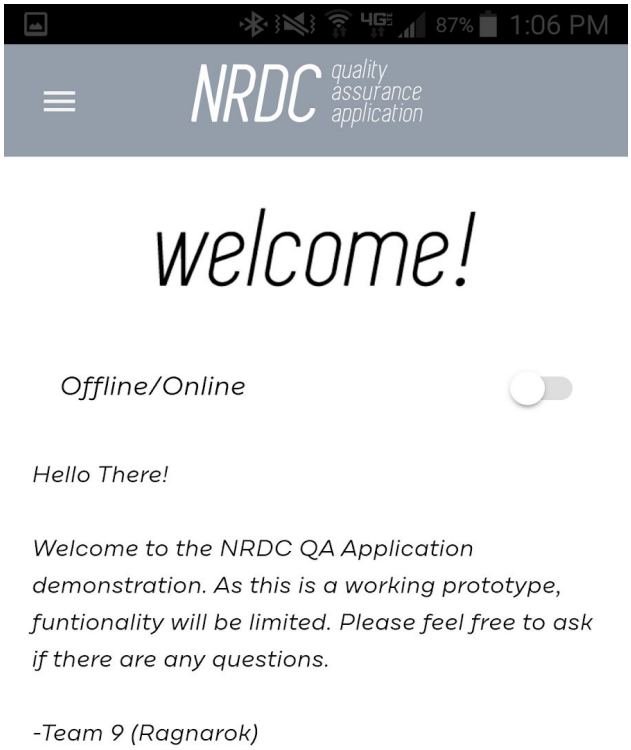

Fig 3. The home view with a message from the development team. The offline/online functionality toggle is also included.

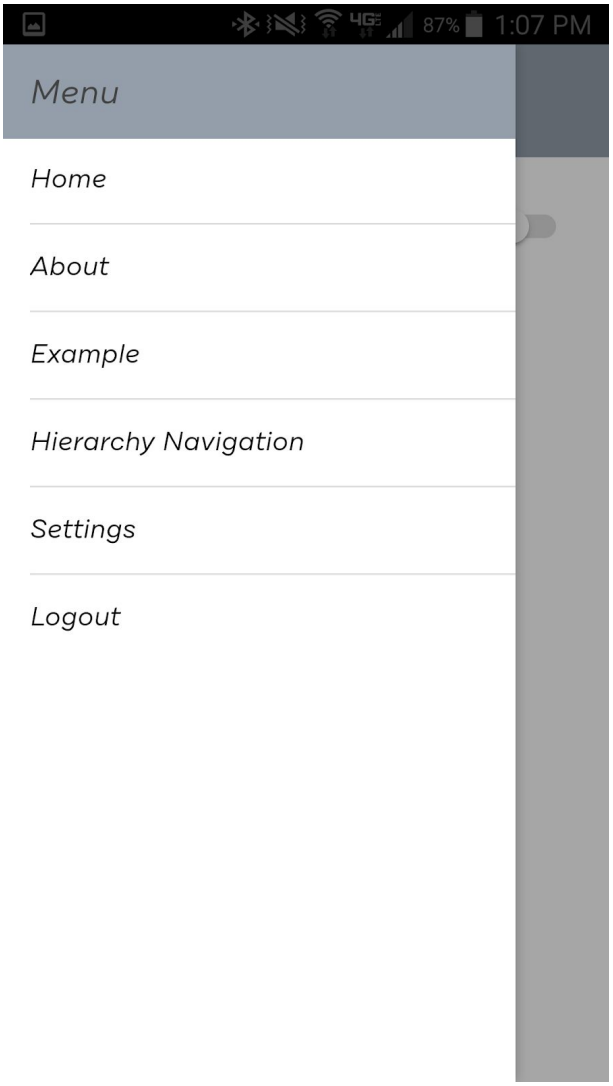

Fig 4. The “hamburger” menu that opens when the user clicks the icon at the top left of the application.

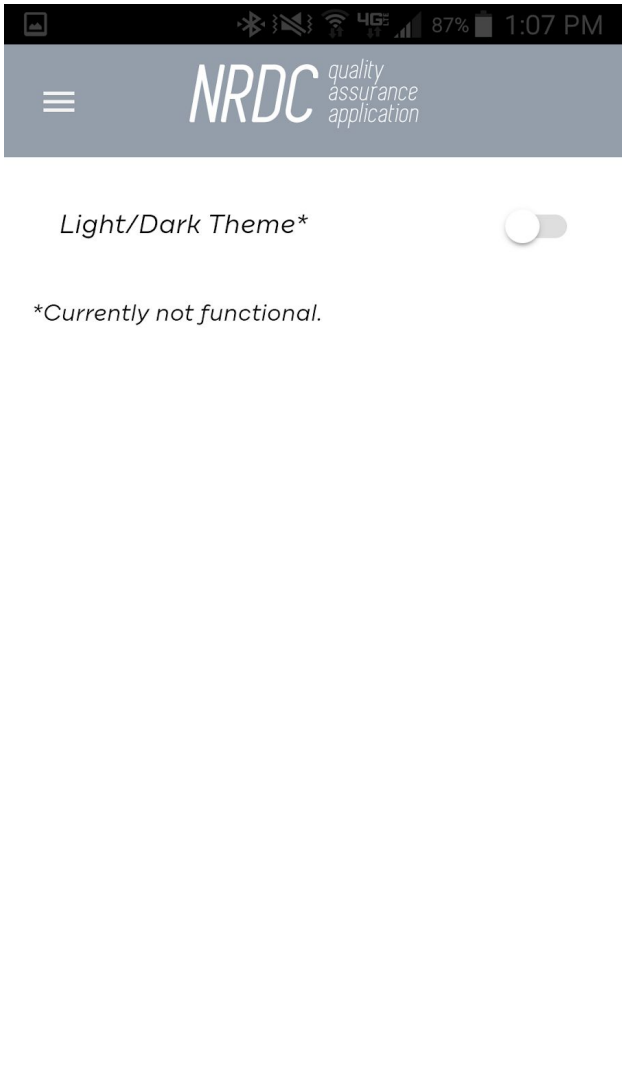

Fig 5. The settings view that currently has no real functionality,

Figures 6 and 7 are the About page, which highlights some stylistic choices in the app, and Hierarchy Navigation views, which shows how the hierarchy is navigated. Figure 8 is the demo example page which shows what the user will expect to see from a read only data page and Figure 9 gives a good view of how fields are being dynamically created and data is being dynamically collected. The current user interface is subject to change.

Fig 6. The About view where information regarding the team, project and advisors are included.

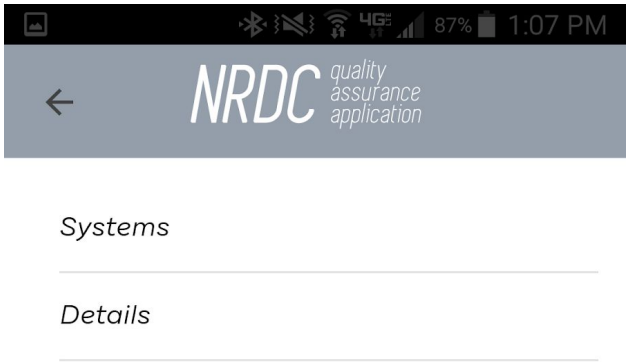

Fig 7. The Hierarchy Navigation view that is dynamically populated using the JSON.

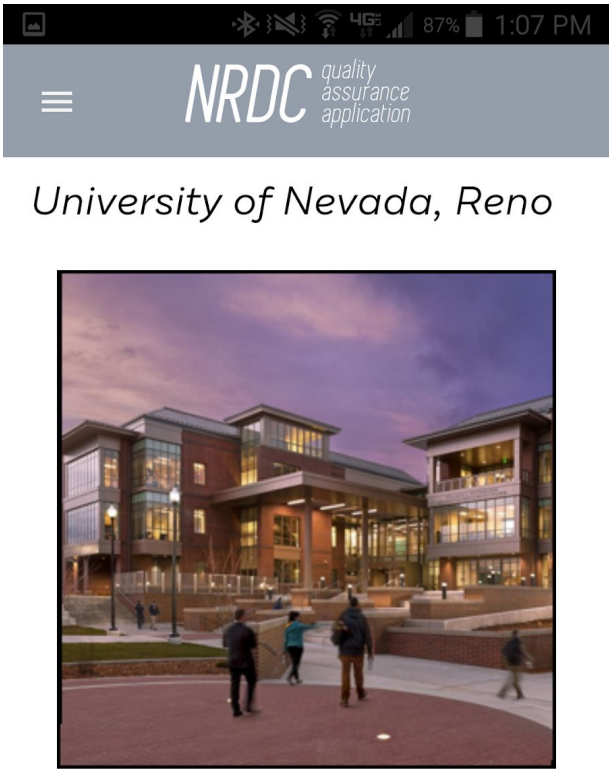

Fig 8. The Example view that will help illustrate what will populate when selecting a site.

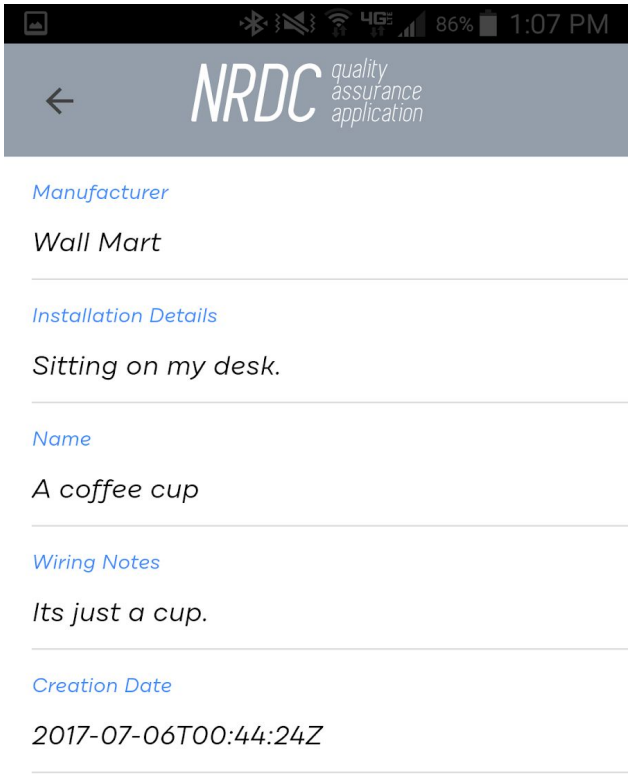

Fig 9. The data view that populates using the current JSON.

6. Evaluate Prototype

For the evaluation segment, Team 9 met with stakeholders Vinh Le and Connor Scully-Allison. Dr. Scotty Strachan was unavailable for evaluation.

6.1 Stakeholder Meeting with *Vinh Le* (Thursday, Dec 6. 4:30pm)

During the stakeholder meeting with Vinh Le's general reception to the progress on the demo was positive. With this in mind, he provided mostly feedback on features that still need to be implemented as well as features that could use some improvements. In the following section, his feedback is broken down by individual sections of the application.

Home Page:

- The offline/online toggle button should be more significantly separated from the welcome message.
    - Since this feature is critical to the application, it would be better suited in the header with a strong visible indication of the online/offline state.
- The home page could be made more visually appealing with a nice graphic.

About Page:

- Permission was given to use pictures of advisors.
- Adding team member aspirations could help improve the section.

View/Navigation Pages:

- Currently there is no indication that a view can be scrolled up or down to view more data, so adding one is an important addition for usability.
- Each section needs to populate the full (and correct) list of headers, fields, and titles. (e.g. Site-Networks would contain Nevcan or Walker Basin, and each of those would contain their own details).
- Currently, the menu disappears when viewing the hierarchy pages. This could be ok, but the team needs to make a decision on if it wants to lock the user out of going directly back to home while navigating the hierarchy.
- Certain fields need special considerations.
    - Need to include the altitude in GPS location.
    - GPS needs to indicate a clear distinction between longitude and latitude.
    - GPS needs to be have conversion functions for transferring to/from the back-end.
    - Date and time needs special conversions for both data transfer and ensuring the times in the app are based on the local time of the device.
    - Photos will need conversion functions for data transfer.

Settings Page:

- Settings page should have more settings (at minimum three).
- A change password setting would need to be a link to a site outside of the application.

6.2 Stakeholder Meeting with *Connor Scully-Allison* (Thursday, Dec 6. 5:00pm)

Connor told us several items in particular that he was happy about seeing in the application. He loved the graphical designs such as the logo in the header of the app and the graphics in the about section. He loved the breakdown of the data with its dynamic labels and dynamic data entry fields. He also loved that the data was populating the fields dynamically. Overall he was very excited to see that the app was being built from the ground up as a dynamic app rather than statically and was pleased that it was done as quickly as it was. He also had more comments about future improvements, which are expanded upon in the following section.

Home:

- On the online/offline page should be clearer. (Sectioned module rather than spaced from text).

About:

- Recommended to add a section about the app or about the NRDC and its history.
- Potentially add information in the ontology to get the about link to NRDC.

Example:

- Image conversion is very important. As such, the team should look into saving locally, remotely, etc.
- Example looks good for read only. Very clean looking.

View/Navigation:

- A very important next step is to list out the items in an organizational tier rather than the current functionality of listing the tier itself and the data for the first item in it.
- It is important to list out all items in an organization tier of the hierarchy (e.g. need to list out all “site networks”.)
- Instead of the term “Hierarchy Navigation”, use a term more suited to the function of the application as seen by users. (e.g. “Metadata Entry”)
- Views need better indicators to show how move up or down the hierarchy.

- Clicking a button should take you to what that button says instead of the current functionality where it is more of a header that clicks to go to the next tier in the hierarchy.
- The current themed colors potentially generic.
- Fields in the views need an indicator for if it's editable or read-only.
- Warning: We will have to query data from hierarchy navigation page in the application.
- Need to be sure to have local storage for data on the device: JSON flat files is ok, but a database a great option. This is a super high priority.
- Connor would like to see what buttons look like in the future.
- Mark down a bug to fix: the app can navigate past components when it shouldn't.
- Since fields are rendered empty when no data is in the field, it possibly needs placeholder item to indicate no data is in the database.

Login:

- Login needs to keep white text when logging in instead of turning black.
- Backgrounds could be super low opacity (like 30%)

General:

- The app will need some checks for IOS/Android for compatibility.

7. Demo Prototype

The NRDC Quality Assurance Application demo was held on Thursday, December 6, 2018 at 10:00 am.

8. Changes Needed to Software Design

Based on stakeholder feedback, changes will be made to the software requirements and design specifications. These changes include moving current requirements to different tiers. For example the search functionality will be made a tier three requirement instead of a tier two requirement. Notifying the user when they are able to synchronize data as a lock screen notification will also be moved from a tier one to a tier two functionality due to the way the online and offline functionality is currently implemented.

New requirements will also be added based on feedback and the current status of the prototype. Implementing a thesaurus in the backend's Flask service will be added as a tier two requirement instead so as to allow users to upload ontologies with less strict specifications. For example, instead of using the term label, which is required to create and label data fields in the application,

the administrator can upload an ontology that uses the term field. Another new tier two requirement that will be added is accepting multiple file types for the ontology. In its current iteration, the application only accepts XML files. It is planned to later include either Turtle, JSON-LD, SPARQL, or some combination of these. This makes the design of the ontology more user friendly. A new tier one requirement has also been added, including online and offline functionality. This requirement has already been partially implemented; however, the storing of data while offline and keeping a local copy of navigation data and details sections has not yet been started. Additionally, the current option to switch between online and offline modes is not currently clear enough and needs to be modified to make this feature more user friendly.

The design specifications will also be updated. The Settings class will be combined with the Updates class to make one Options page. This new class will contain both the settings information for the user and an update log. An option to send a password change request to the administrator will also be added for users. The classes *Parser* and *HierarchyManager* will also be combined into a single class, *MetadataEntry*, to have a more friendly page name. As search has been moved to a tier three requirement due to time constraints, the Search and SearchResults classes may not be implemented. An updated version of the class diagram can be found in Figure 10.

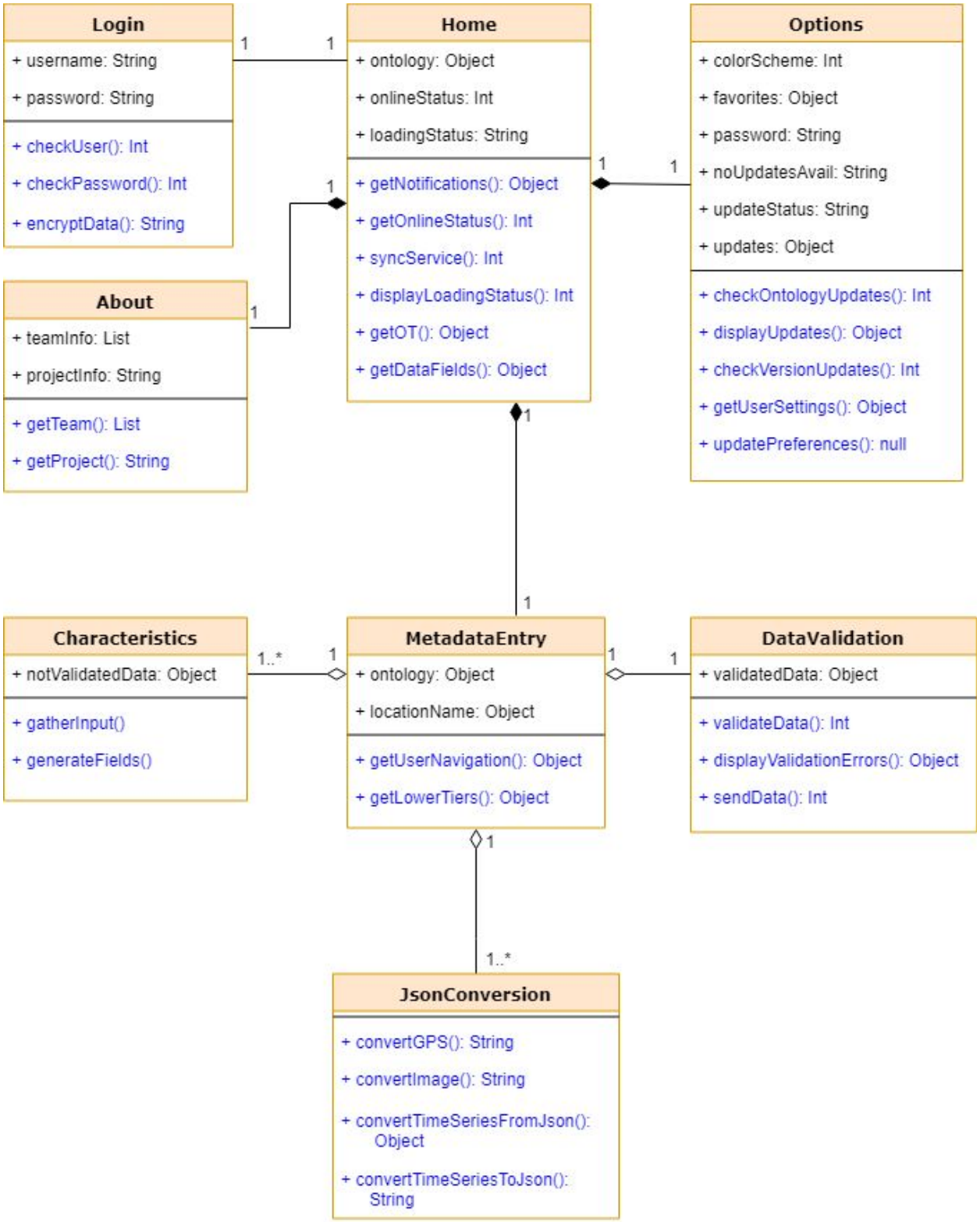

Fig 10. An updated class diagram for the application after combining Settings and Updated into the new Options class, removing Parser, Search and SearchResults, and changing HierarchyNavigation to MetadataEntry.

9. Contributions of Team Members

- Brianna Blain-Castelli
    - Contributions
        - Paper contributions - 4 hours
            - 3. Introduction
            - 5. Develop Prototype
            - 7. Demo Prototype
            - 8. Changes Needed to Software Design
            - 9. Contributions of Team Members
        - Prototype Implementation - 113 hours
    - Time Worked: 117 hours
- Christopher Eichstedt
    - Contributions
        - Paper contributions - 4 hours
            - 2. Abstract
            - 3. Introduction
            - 4. Prototype Objectives & Functionality
            - 5. Develop Prototype
        - Prototype Implementation - 123 hours
    - Time Worked: 127 hours
- Matthew Johnson
    - Contributions
        - Paper contributions - 4 hours
            - 5. Develop Prototype
            - 6. Evaluate Prototype
        - Prototype Implementation - 125 hours
    - Time Worked: 129 hours
- Nicholas Jordy
    - Contributions
        - Paper contributions - 4 hours
            - 2. Abstract
            - 3. Introduction
            - 4. Prototype Objectives & Functionality
            - 5. Develop Prototype
        - Prototype Implementation - 125 hours
    - Time Worked: 129 hours