Open Water Journal

Volume 4
Issue 1 *Open Water Science and Software*

Article 2

2017

HydroShare GIS: Visualizing Spatial Data in the Cloud

Shawn Crawley
Brigham Young University - Provo, scrawley@byu.edu

Daniel Ames
dan.ames@byu.edu

Zhiyu Li
Brigham Young University, zyli2004@gmail.com

David Tarboton
dtarb@usu.edu

OPEN WATER

HydroShare GIS: Visualizing Spatial Data in the Cloud

Case Study

Shawn Crawley[1*], Daniel P. Ames[2], Zhiyu Li[3], David Tarboton[4]

[1,2,3]*Brigham Young University*
[4]*Utah State University*
[*]*Corresponding Author: scrawley@byu.edu*

ABSTRACT

Cloud-based data management systems are more conducive to collaborative efforts when they are integrated with cloud-computing tools that interact with their stored data. HydroShare, a web based data management system for climate and water data, has implemented an Application Programming Interface and a web application platform deployed using Tethys Platform to encourage the development of apps that interact with its data. HydroShare GIS is the result of one such development effort to provide cloud-based visualization of spatial data stored in HydroShare. It functions by accessing the spatial metadata contained within the HydroShare resource data model and overlaying datasets as layers within the OpenLayers JavaScript library. Data are passed from the app's server to a GeoServer data server and shared as web mapping service layers. Thus, users can easily build map projects from data sources registered in HydroShare and save them back to HydroShare as map project resources, which can both be shared with others and re-opened in HydroShare GIS. This paper will describe the design of the HydroShare GIS app and the cyber-infrastructure that supports it, and evaluate its efficacy as a web based mapping tool.

Keywords
HydroShare, web app, spatial data

1. Introduction

Cloud-based applications are programs that run on both hardware and software that are distributed between one or more remote servers and made accessible to end users through an Internet connection. These applications are desirable to end users for many reasons. They help free up storage space on personal computers, are accessible from anywhere with an Internet connection, eliminate the hassle of installing and updating software, are operating system independent, and facilitate collaboration with others. Due to the many benefits of cloud-based applications, they are being developed at a rapid rate and are transforming the way of life around the world [*Miller,* 2008]. Though many fulfill common needs and are interacted with on a daily basis (i.e. Gmail, Facebook, LinkedIn, Google Drive, Dropbox, Microsoft OneDrive), others are being developed to fill niche needs and desires in various fields, industries, and markets, as is the case with HydroShare.

HydroShare (https://www.hydroshare.org) is a cloud based system for sharing hydrologic data and models (often referred to in HydroShare as "Resources") aimed

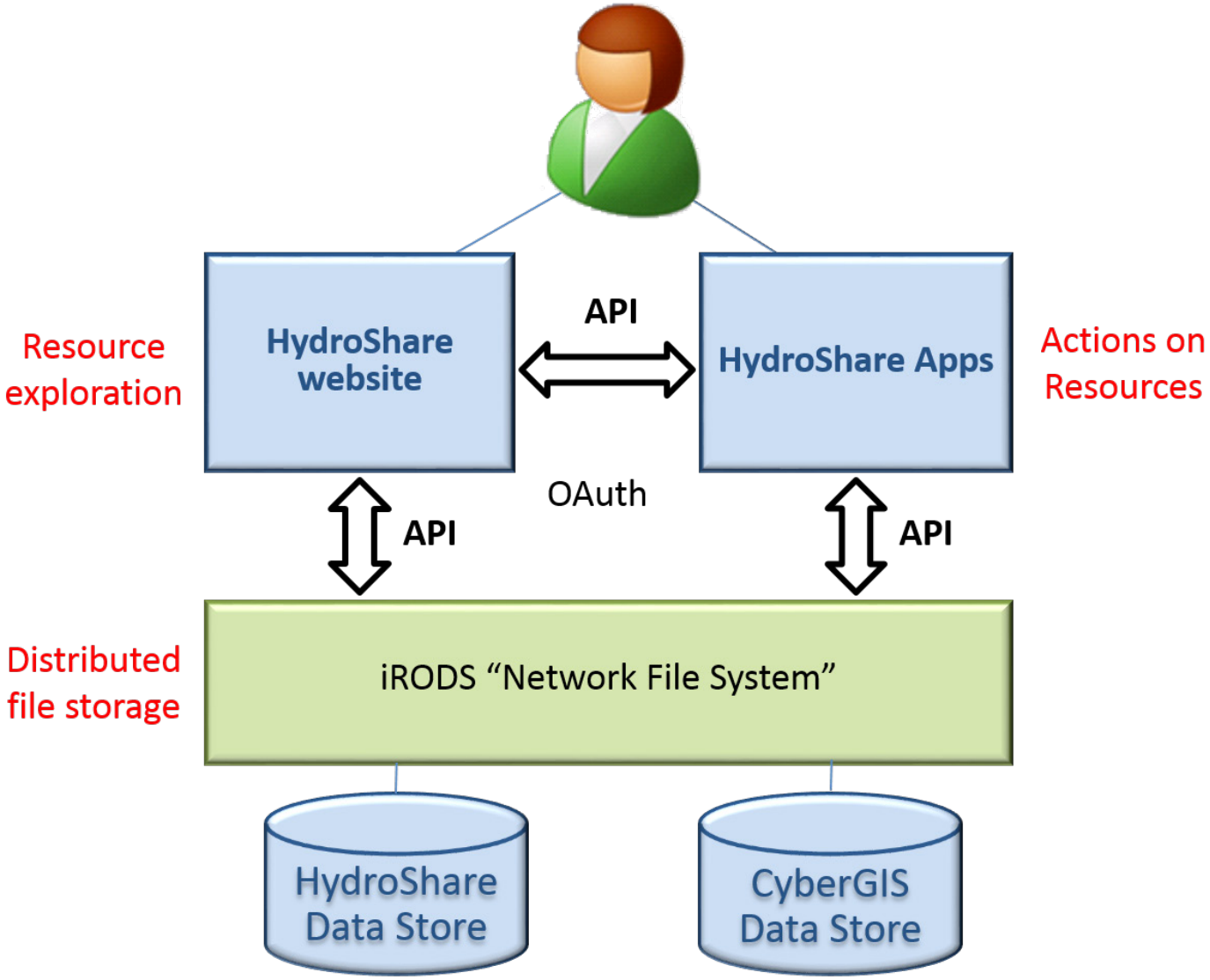

Figure 1: The basic architecture of HydroShare

at giving hydrologists and water scientists the cyber infrastructure needed to innovate and collaborate in research to solve water problems. With HydroShare, users can: (1) share data and models with colleagues; (2) manage who has access to shared content; (3) share, access, visualize and manipulate a broad set of hydrologic data types and models; (4) use the web services Applications Programming Interface (API) to program automated and client access; (5) publish data and models to meet the requirements of research project data management plans; (6) discover and access data and models published by others; and (7) use cloud-based applications to visualize, analyze, and run models on data in HydroShare [*Ames et al.,* 2015; *Horsburgh et al.,* 2016; *Idaszak et al.,* 2016]. HydroShare, at its core, is a cloud-based data management system.

The trend with most mainstream cloud-based data management systems (i.e. Google Drive, Box, Dropbox, Microsoft OneDrive) is to provide not only cloud storage for data, but also cloud-computing tools that interact with the data already being stored and/or generate new data to be stored. For example, Google Drive also provides users with a cloud-based word processor (Google Docs) to edit or create documents, a cloud-based spreadsheet (Google Sheets) to edit or create spreadsheets, and a cloud-based presentation program (Google Slides) to edit or create slideshows [*Covili*, 2016]. The integration of these cloud-computing tools with the basic cloud storage system is largely what encourages and facilitates the collaborative efforts with its data.

Much like Google Drive, the capability to have cloud-based applications that act on its data is a key part of HydroShare that advances its capability within the

general trend towards providing web-based software services. At a high level, the HydroShare architecture is organized into: (1) Resource storage, (2) Resource exploration, and (3) actions on resources (Figure 1). These are implemented using system components that are relatively loosely coupled and interact through APIs. The loose coupling is a variant on Services Oriented Architecture (SOA) that enhances robustness, as components can be upgraded and advanced relatively independently. This also supports extensibility.

The Web Applications (Web Apps) approach is configured so that anyone can establish a Web App server that can act on HydroShare resources. This configuration was accomplished by a number of important developments. First, an application programming interface (API) was developed that gives third party programmers the ability to interact with HydroShare's data resources. Second, the Web App resource type was developed to define launching parameters for third party web apps that act on HydroShare resources (Figure 2). This lets anyone set up a web app to act on HydroShare resources, a very general and powerful extensibility mechanism. Finally, the team has developed an official web apps portal (https://apps.hydroshare.org) as a platform to host web applications developed for HydroShare built using Tethys Platform [*N Swain et al.*, 2015].

Because HydroShare primarily manages data of hydrologists and water scientists, both of whom interact with a significant amount of spatial data, cloud-computing tools that interact with HydroShare's spatial data would be a significant addition and fill a niche need. It would also likely generally increase users' motivation to use HydroShare as a storage space for spatial data.

The problem is that though many cloud-based applications and cloud-computing tools that interact with spatial data already exist, none of them could be readily integrated with HydroShare. Many of them are either already filling a niche need and being tightly coupled with an existing cloud-based data management system, or provide generalized components that can be ingested and integrated into other development efforts. Google Earth (https://www.google.com/earth/) facilitates web-based spatial data visualization through its system of user-developed KML (https://developers.google.com/kml/) files that can be uploaded and shared via the Google web services. Esri's ArcGIS Online (http://doc.arcgis.com/en/arcgis-online/reference/what-is-agol.htm) system facilitates a commercial web and cloud-based data storage system that allows users to develop custom map data and share it with users in a web-based geographic information system (GIS). The OpenStreetMap Foundation and its contributors have developed a massive web-based open access spatial data repository via crowd-sourced individual contributions (https://www.openstreetmap.org/about) and include cloud-based tools for viewing and interacting with these datasets.

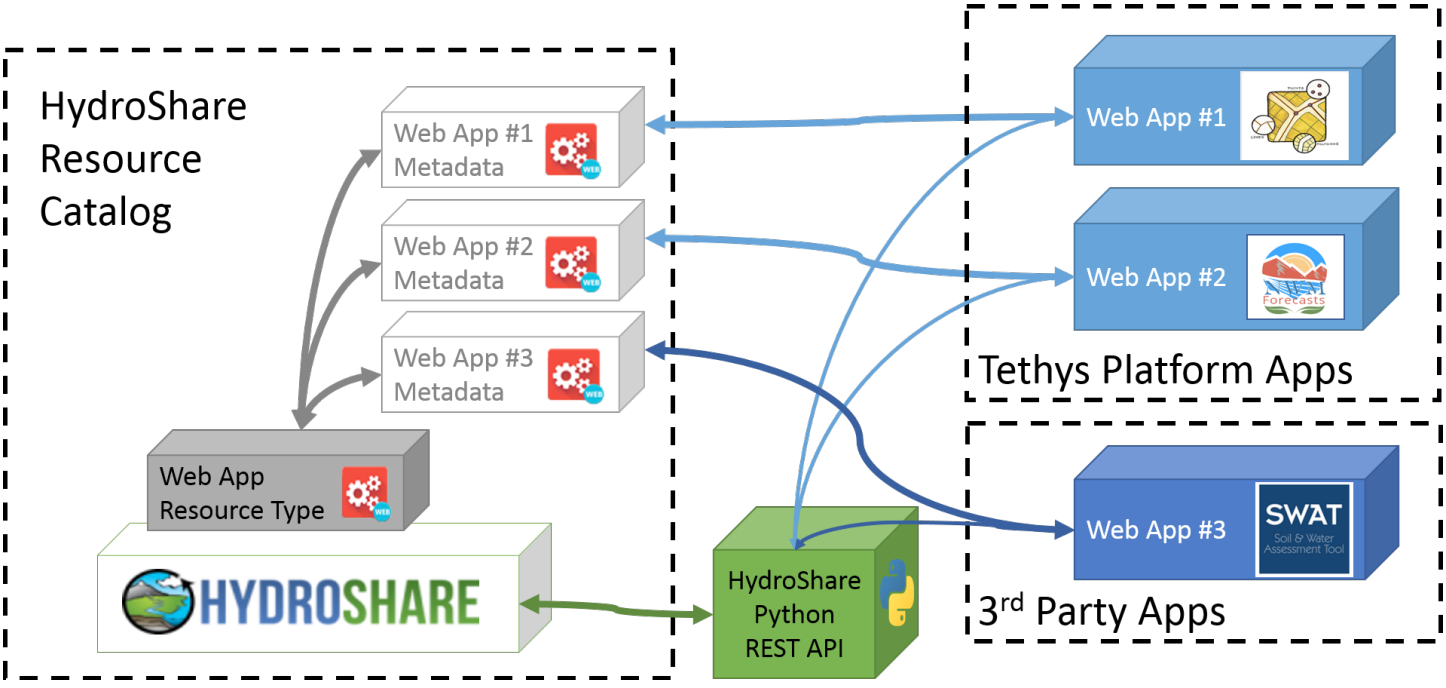

Figure 2: Diagram showing how HydroShare is set up to interact with web applications

The good news is that much advancement and innovation has been achieved in the field of spatial data cloud computing in recent years that could contribute to and greatly simplify the development of a cloud-based application for interacting with HydroShare's spatial data. For example, in 2015, Swain, Latu et al. (2015) performed a study of available free and open source software (FOSS) for geographic information systems (FOSS4G). As a result of this study, many of the best libraries were incorporated into the software development kit (SDK) of Tethys Platform, HydroShare's chosen web application platform. Additionally, the Open Geospatial Consortium has provided standards and protocols that have largely facilitated the integration of distinct spatial-centric systems (http://dx.doi.org/10.1016/j.envsoft.2012.11.010). CyberGIS is another organization that has been leading research efforts in the fields of spatial data science and spatial cyber infrastructure since 2010 [*Wang,* 2010; 2016; *Wang et al.,* 2013; *Yin et al.,* 2017].

In light of all of these advances, we undertook the development effort to create a lightweight, cloud-based GIS that would provide customizable viewing of one or more HydroShare spatial resources on an interactive spatial data display (map). The app, developed using Tethys Platform, was named HydroShare GIS and is currently deployed on HydroShare's app portal.

This paper first describes the methods that were used to develop HydroShare GIS, including the software design and a few potential use cases for the app. Then, the resulting web application as it currently stands is discussed in terms of its features, behavior, and ability to perform as expected during the potential use cases. Finally, conclusions are drawn from the development and deployment of HydroShare GIS, lessons learned, and opportunities for the future of HydroShare and other web based systems or projects that interact with spatial data.

2.0 Key Features and Functionality

We identified the following key features as being important in enabling HydroShare GIS to support customizable visualization of spatial data stored in HydroShare.

1. An interactive map that allows the user to:
   a. Pan
   b. Zoom
2. A way for the user to:
   a. Add one or more HydroShare resources to the map, each as a layer
   b. Add one or more local files to the map, each as a layer
   c. Save the current configuration of HydroShare GIS back to HydroShare as a resource
3. An interactive and dynamic list of the current layers, where each layer entry can be interacted with to:
   a. Change the display order of its corresponding layer
   b. Toggle the visibility of its corresponding layer
   c. Change its own display name text
   d. View the attribute table of its corresponding layer (if applicable)
   e. Modify the symbology of its corresponding layer
   f. View the legend of its corresponding layer
   g. Zoom to the extents of its corresponding layer
   h. Open the web page of its corresponding resource in HydroShare
   i. Remove its corresponding layer from the map and itself from the current layers list

We also wanted HydroShare GIS to exist as a single-page web application because of the many benefits of doing so[Mikowski and Powell, 2013]. This would mean that all features and functionality would need to be accessible without reloading the page or loading a new page. The graphical user interface (GUI), or the way this single page would be laid out, in HydroShare GIS can be seen in Figure 3. Because the core purpose of this app is to view spatial data, the map would be the main focus, taking up about 75% of the screen space. Most of the remaining space would be taken by the side panel, which would contain a list of the layers currently added to the map and a few buttons for adding new layers to the map and saving the current state of the app.

We designed HydroShare GIS to be launched both directly from its root URL, and from the landing page of any HydroShare resource that the app can interact with. If launched directly, HydroShare GIS would display a blank map and an empty current layers list. Layers could then only be added to the map through a user clicking on either the "Add HydroShare Resource" button or the "Add Local File" button. If launched from the landing page of a HydroShare resource, the app would launch and then automatically begin loading the correspond-

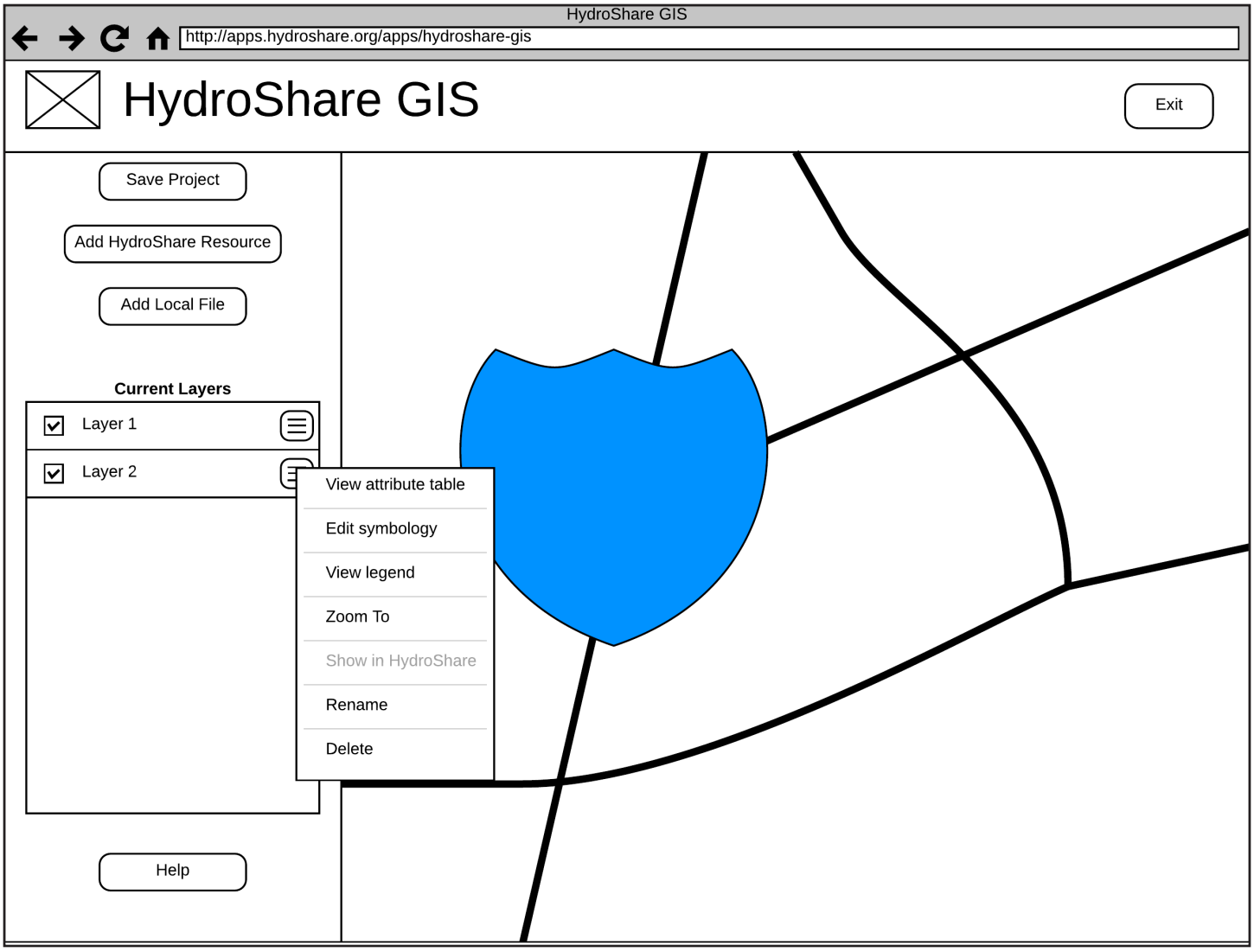

Figure 3: Mockup of the user interface representing the main screen of the HydroShare GIS

ing resource as if it had been selected using the "Add HydroShare Resource" button method described next.

In the case that the user clicks the "Add HydroShare Resource" button, the appearing UI modal (Figure 4) would contain a list of all of the resources stored in HydroShare that the user has permission to access, which would include both resources owned by the user and those owned by others, but made public. From here, the user would be able to select one of the listed resources and then click an "Add" button to trigger the process of adding the chosen resource to the map. The successful termination of this process would result in the rendering of the associated spatial data in its appropriate location as a layer on the map, and the appearance of a new corresponding entry at the top of the current layers list. As mentioned, this is the state the app would be initially loaded in if launching the app from the landing page of a valid HydroShare resource.

In the case that a user clicks the "Add Local File" button, the appearing UI modal would contain a form that lets the user specify both the location where the uploaded file(s) will be stored and the specific file(s) to be added. The user can either store the file(s) in a new, separate HydroShare resource, or within the resource that contains the previously saved state of the current HydroShare GIS session. If the user chooses to store the file(s) in a new, separate resource, they will be given the additional options to specify the title, type, abstract, and keywords of the new resource. Once all of the required options are specified, the user could then select an "Add" button, which would trigger the process of storing the file(s) in the appropriate resource and adding the file(s) to the map. Just as with the process following a click of the "Add" button from the "Add HydroShare Resource" modal, the successful termination of this process would result in the rendering of

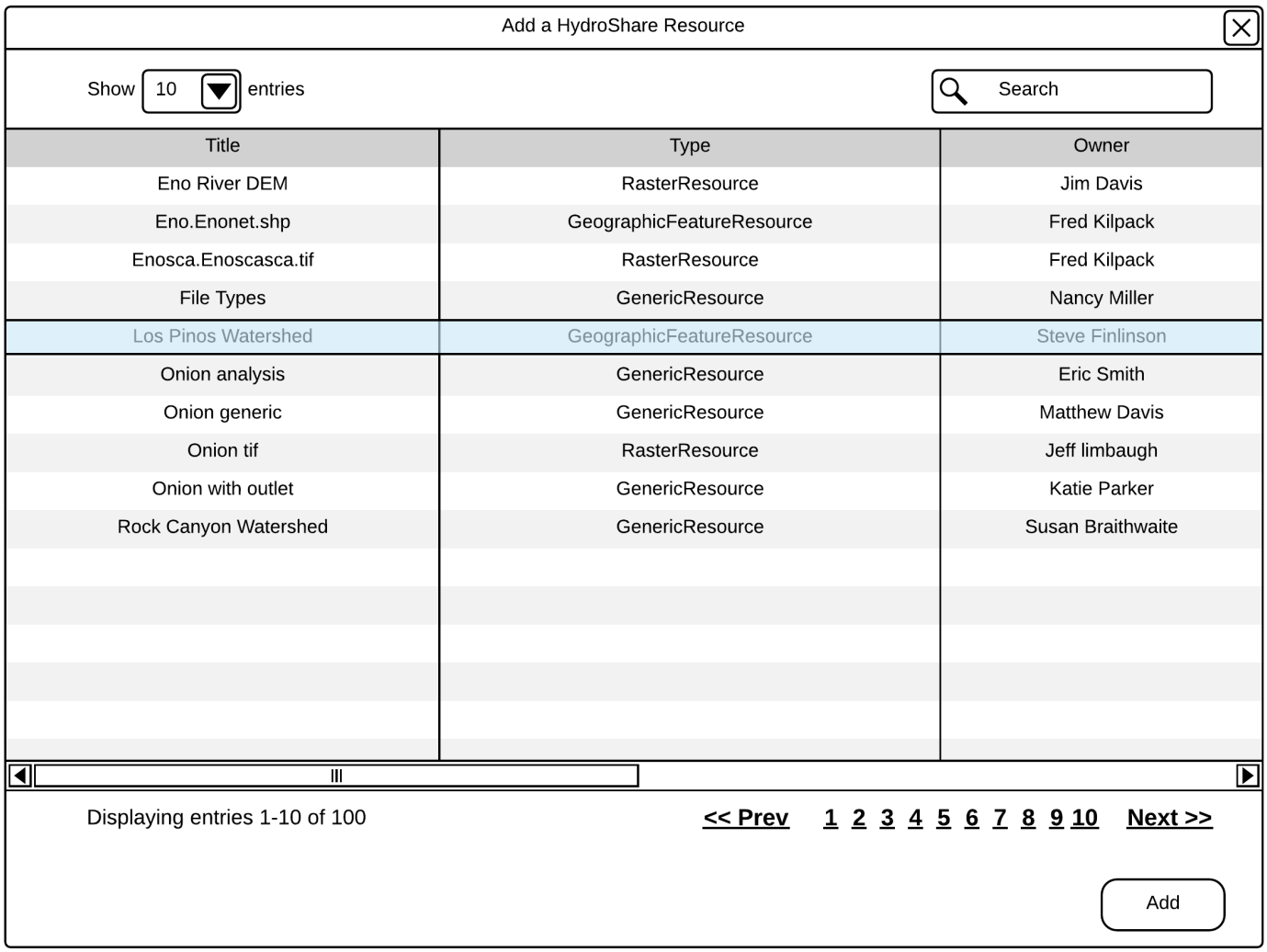

Figure 4: Mockup of the modal dialog used to add a HydroShare resource to the current HydroShare GIS project

the associated spatial data in its appropriate location as a layer on the map, and the appearance of a new corresponding entry at the top of the current layers list.

We designed each layer in the current layers list to contain three main parts. First, a checkbox on the left side of the layer list item would allow the user to toggle the visibility of the corresponding layer on the map, with checked being visible and unchecked being invisible. Second, a modifiable display name for the corresponding layer would appear immediately to the right of the visibility checkbox that defaults to either the corresponding resource's title or file name. Modifying the display name would not modify the underlying resource or file. The third part of the layer list item would be a hamburger menu icon that, when clicked, would bring up a dropdown context menu providing context-specific options for the corresponding layer.

Since the core feature of HydroShare GIS is the ability to visualize spatial data (map layers) in a customizable way, the most important context-specific dropdown option item is the "Modify symbology" button. When the user clicks this button, a modal dialog will appear that will look much like the mockup in Figure 5. This figure captures what it might look like when modifying the symbology of a vector-based spatial dataset, such as a shapefile of polygons. Color pickers would allow the user to choose any possible color based on red, green, and blue (RGB) values; hue, saturation, and value (HSV) values, or a visual color wheel. The user would also be able to turn feature labels on and off, specify which attribute field to provide the labels from, and choose the label's font size and color.

The "Save Project" button would bring up a UI modal with form data to specify the title, type, abstract, and keywords of the new resource that would be created to store the file that encodes the current state of the app. This file would be in JavaScript Object Notation (JSON) and store key/value pairs that capture the state of the app at the time of clicking the "Save" button.

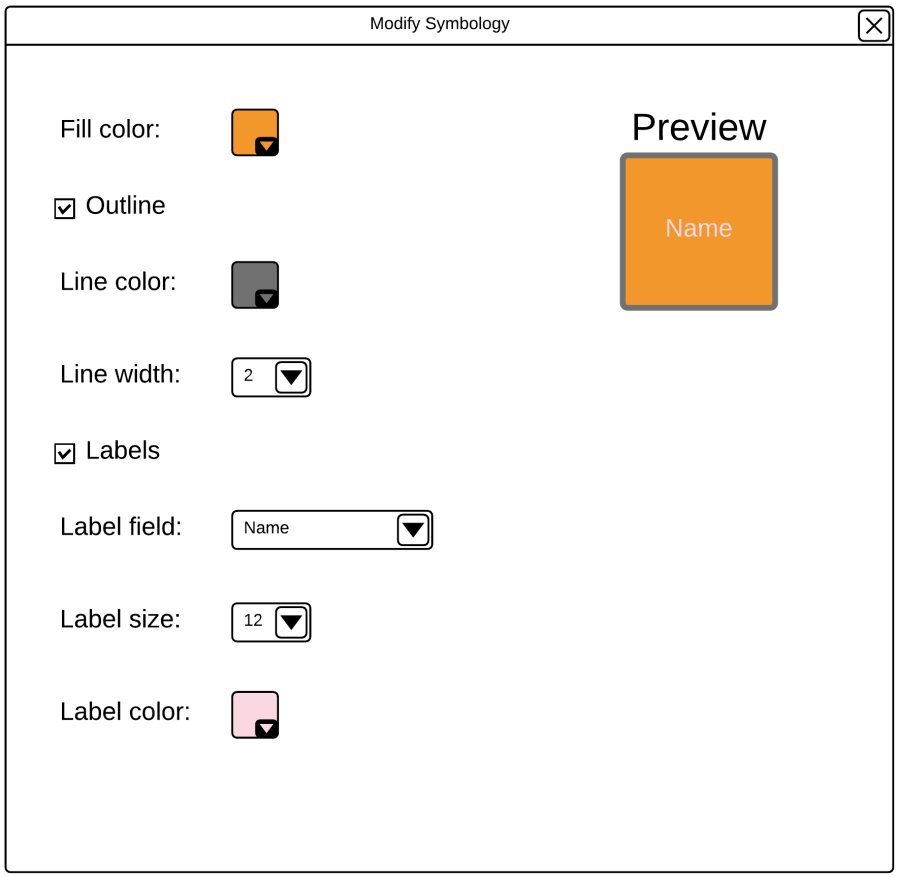

Figure 5: Mockup of a modal dialog to modify layer symbology in HydroShare GIS

2.1 System Architecture

To provide all of the key features and functionality of HydroShare GIS with the desired UI layouts, many existing technologies were integrated to form the system architecture seen in Figure 6. The remainder of this section will go into detail breaking down and explaining each component seen therein.

HydroShare GIS was designed using the Model-view-controller (MVC) pattern, which separates the application into three interconnected parts – the Model, View, and Controller – to isolate the logic and functionality of each part and to simplify the development process. This pattern was chosen since the Django and Tethys platform frameworks that it is built upon are each MVC frameworks themselves.

Django (https://www.djangoproject.com/) is a free and open source, high-level Python web framework that facilitates Web development. It simplifies the Model aspect of MVC by encapsulating the bulk of the SQL and database interaction into custom Python classes, properties, and methods that can be inherited from and added to. It simplifies the Controller aspect of MVC by leveraging URL mappings that specify Python functions that should be triggered based on the URL endpoint browsed to. It simplifies the view aspect of MVC by providing pseudo-object-oriented HTML templates that allow for inheritance, overriding and the insertion of Django-specific logic, filters, and variables that are converted to standard HTML by the Django functions that render them. These features, and others, make Django very versatile and perfect for any web development project.

Tethys Platform is a web development framework that is powered by Django [*N R Swain et al.*, 2016]. While Django is a versatile, generalized web development platform, Tethys Platform adds to it a carefully chosen set of libraries and tools to support the web app feature and functionality needs of the water resource engineering and hydro-informatics community. Ultimately, the development of HydroShare GIS was largely simplified by taking advantage of the spatial and GIS features provided by Tethys Platform.

2.1.1 Model

In the MVC architecture, the Model defines how all of the application's data is stored and

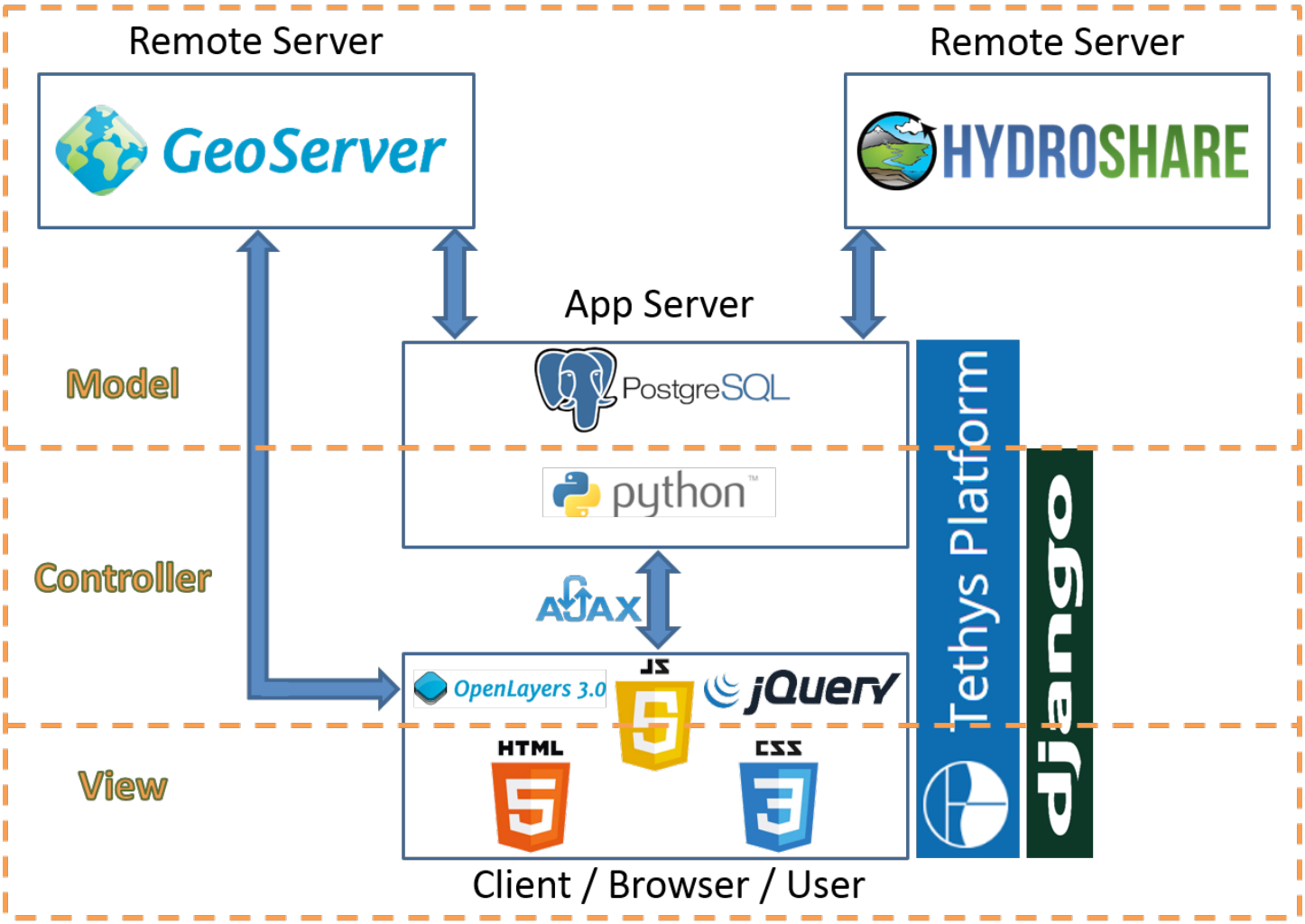

Figure 6: System architecture of HydroShare GIS

accessed. In HydroShare GIS, the Model is composed of HydroShare, GeoServer, and PostgreSQL.

In the context of the HydroShare GIS system architecture, HydroShare is the principal component of its model. Essentially all of the data that is handled by the app and made available to the end users comes from HydroShare. Even the files representing the saved state of HydroShare GIS are stored back on HydroShare to be persisted there as resources.

HydroShare resources generally consists of (1) one or more files that share many common metadata, and (2) the metadata itself. Each resource has a specific type, which restricts the breadth of file types that are associated with it in order to provide more specificity, simplicity, and depth. For example, TIF files have many specific metadata that would not be applicable to, say, Python files. Thus, a TIF file can be uploaded as a Raster resource rather than a Script resource, which would accept Python files. However, because files and their metadata can also be grouped based on a relationship other than common file types, such as being part of a project or being collected at the same location, HydroShare also provides a Generic resource, which sacrifices the depth of specific metadata for the breadth of associated files. Any number and type of files can be contained in a Generic resource.

With all resource types except Generic, HydroShare will automatically extract much of the file-specific metadata upon resource creation based on the resource type. For example, the coordinate reference system, spatial extents, and cell value extremes are automatically extracted and stored for Raster resources. In addition to automatic metadata extraction, certain metadata can be manually entered for any resource, such as spatial coverage metadata representing where the associated data was collected or what spatial location or entity it represents.

Because we designed HydroShare GIS principally as a spatial data viewer, the resource types that HydroShare GIS should interact with at a minimum are Raster and Geographic Feature. However, since all resources may

contain spatial coverage metadata, we extended the design and scope of the app to be able to also interact with Referenced Time Series, Script, and Generic resources. However, it is possible for a Generic resource to contain spatial files, such as TIFs and shapefiles, and in this case it would be treated somewhat as if it were one or more Raster or Geographic Feature resources.

GeoServer acts as another key component of the HydroShare GIS model. The spatial files retrieved from Raster and Geographic Feature HydroShare resources (or Generic resources storing similar files) are uploaded to GeoServer, whereupon the data is both stored on the server and also processed and converted into spatially referenced (projected) image tiles that are then retrieved using a URL endpoint and rendered on the map.

GeoServer was chosen as a key component of HydroShare GIS because it would easily facilitate the persistence of the HydroShare resource layers by acting as a layer database. Once a resource's spatial files are uploaded to GeoServer and processed as web mapping service layers, these layers can be stored there indefinitely. This means that the delay experienced by a user during this uploading and processing would be a one-time expenditure. All subsequent viewing of that resource using HydroShare GIS would happen relatively instantaneously since the layers would persist and need only be re-fetched.

It is possible that a HydroShare user modifies the files belonging to a resource that has been previously viewed using HydroShare GIS. In this case, it would not be desirable to simply re-fetch this resource's previously processed layer since the updates made to this resource's files would not be reflected in the existing layer. To handle this, HydroShare GIS extracts the timestamp metadata corresponding to the most recent modification of the resource and stores it in a PostgreSQL database (see the next section for a full explanation of PostgreSQL). This is done on the first load of all resources. Then, on each subsequent load of the same resource, its current modification timestamp metadata is extracted and compared to the previously stored timestamp. If the modification timestamp of the file is more recent than what the database reflects, the resource is reloaded as if it were the first time and overwrites the existing, outdated layers. Otherwise, the resource's corresponding layers are simply re-fetched from GeoServer.

PostgreSQL, though not essential to the core functionality of HydroShare GIS, was integrated into its model to optimize the performance of the app. It does so by providing a database management system that stores the subset of resource metadata from HydroShare and the subset of layer metadata from GeoServer that are needed by the app, as well as other metadata calculated by the app itself. This would provide quicker access to each resource's corresponding metadata on post-initial loads than HydroShare and GeoServer are able to provide. This is because (1) SQL databases are engineered and optimized for data retrieval, and (2) the database would be hosted on the same server as the app, as opposed to remotely.

There are at least two cases in which, without the PostgreSQL database, the app would not be able to detect whether or not a resource should be re-processed after its initial load. The first case, described in the previous section, would occur when a resource had been updated on HydroShare since being previously processed by the app for viewing. The second case would occur when attempting to view a Generic resource whose files constitute more than one spatial layer to be stored in GeoServer. Since most resources correspond to one GeoServer layer, the resource's unique identifier (ID) was chosen to also be the unique GeoServer layer identifier used to re-fetch the layer. This, however, no longer works in a one-resource-to-many-layers relationship. If the PostgreSQL database was integrated into the app, each individual GeoServer layer created would get its own entry in the database that would contain its associated metadata, such as the ID of the resource it corresponds to. Thus, the unique ID of all layers associated with one resource could easily be discovered by querying the database for all entries whose corresponding resource ID match the ID of the resource in question. Then, each layer could easily be re-fetched from GeoServer using its unique layer ID.

With the PostgreSQL database implemented, after a resource has been loaded for viewing once, each subsequent viewing would generally consist of extracting all of the relevant metadata from this database, as opposed to from HydroShare and GeoServer.

2.1.2 View

In the MVC pattern, the View consists of essentially everything that is displayed or accessible to the user. For web development in general, and thus for HydroShare GIS, the view is created and managed using three technologies: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript (JS). Many web development frameworks, libraries, and plugins exist that implement one or more of these tech-

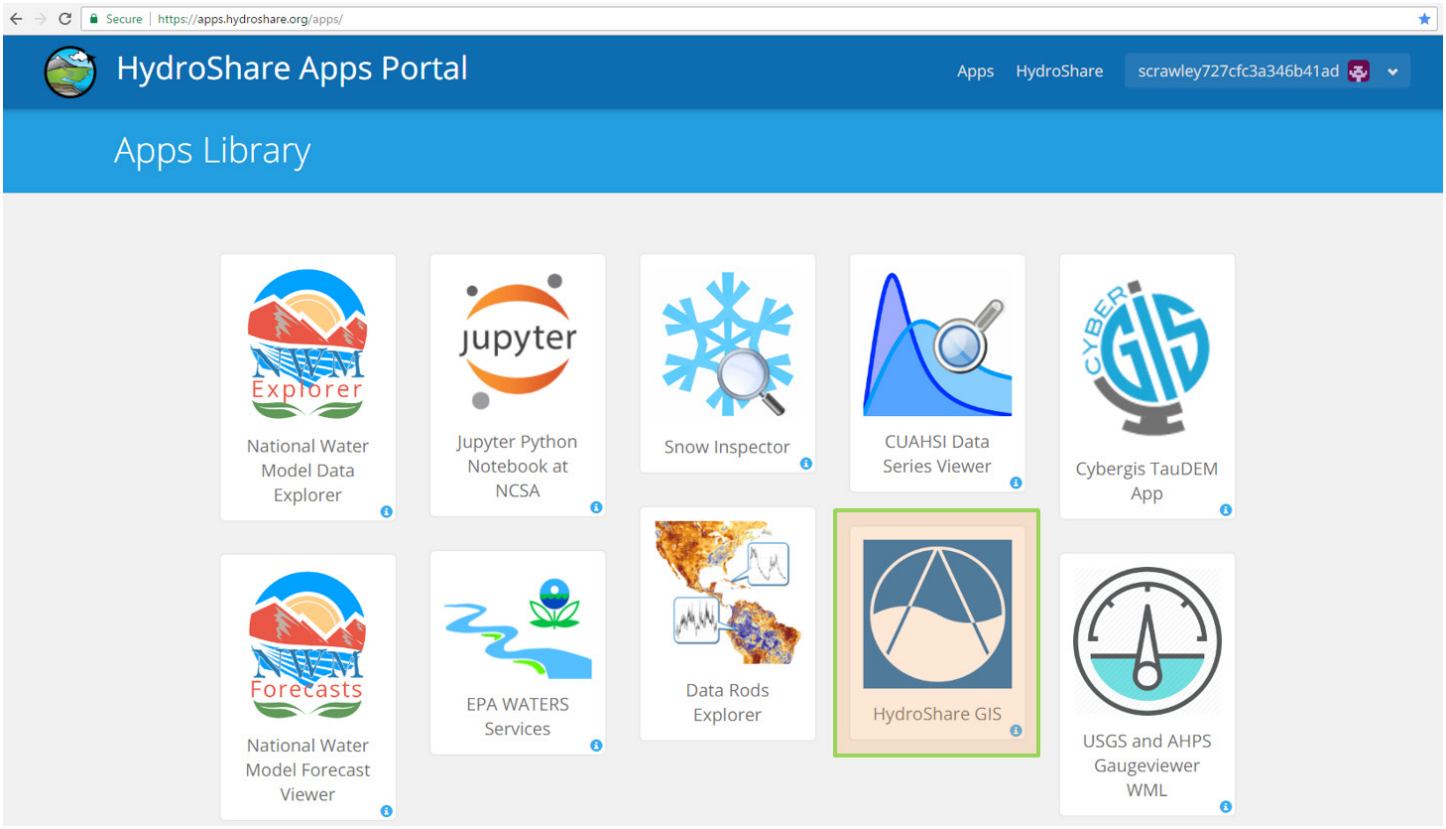

Figure 7: Screenshot of HydroShare Apps Portal highlighting the HydroShare GIS application

nologies. Of these, those most heavily implemented by HydroShare GIS are contained in the following list containing a brief description of the functionality provided.

OpenLayers (http://openlayers.org) - Renders the map, interacts with GeoServer to fetch and display the HydroShare resource layers, and provides an API to aid in handling all user interactions with the map, such as panning and zooming.

Bootstrap (http://getbootstrap.com) - Simplifies the style, layout, and responsiveness of the app's graphical user interface (GUI) by providing well-designed form elements (i.e. inputs, buttons) and modal windows and by making the content resize and fill space appropriately based on the user's browser size.

jQuery (https://jquery.com) - Provides functions that simplify the programming required to react to a user's interaction with the app and update its content accordingly. It also provides simplifies much of the communication between the user's browser (client) and the app server that happens using Asynchronous JavaScript and XML (AJAX).

jQuery UI (https://jqueryui.com/) - Simplifies the programming required to react to the specific user interaction in which layer list items in the current layers list are reordered by click-and-drag. This is accomplished through jQuery's Sortable interaction (https://jqueryui.com/sortable).

contextMenu.js (http://ignitersworld.com/lab/contextMenu.html) - Provides the dropdown menu of options for each layer list item in the current layers list and either provides or simplifies the handling of all user interactions with it.

DataTables (http://datatables.net/) - Provides the style and layout of the resource layer attribute table and the table of all HydroShare resources and either provides or simplifies the handling of all user interactions with it. This includes the user's ability to sort the table by column in ascending or descending order; a search field that dynamically filters the table's contents by partial or full match of a user's typed string; and table pagination with contextual info about what is currently being displayed.

Spectrum (http://bgrins.github.io/spectrum/) - Used extensively in the "Modify Symbology" modal to provide to users with interactive color pickers that provide simplified, intuitive GUIs for choosing from millions of colors to customize the visualization of their resource layers in the map.

2.1.3 Controller

Figure 8: Screenshot of the main screen of HydroShare GIS

The Controller in the MVC paradigm consists of the logic that connects the Model to the View. In other words, it handles the details of how data is transferred between the Model and View, what data should be transferred, and when the data should be transferred.

In HydroShare GIS, the core Controller functionality is programmed using both JavaScript and Python and leverages many of the features provided by the technologies already discussed such as Django, Tethys Platform, jQuery, and OpenLayers. This section will focus on additional third-party Python libraries that are relevant to the HydroShare GIS Controller model. These include HydroShare's Representational State Transfer (REST) Client (hs_restclient), the Geospatial Data Abstraction Library (GDAL), Requests, and the xmltodict package.

hs_restclient (http://hs-restclient.readthedocs.io/) – Wraps the HydroShare REST API calls in Python functions and is used to establish all connections to transfer data between HydroShare GIS and HydroShare.

GDAL (http://www.gdal.org/) – Used in certain situations to extract additional metadata from the raster datasets that HydroShare might not provide and modify the projection metadata of raster files.

Requests (http://docs.python-requests.org/) – Used to establish connections to transfer data between HydroShare GIS and all external servers except HydroShare.

xmltodict () – Unifies the way responses are handled in the app by converting all XML responses to JSON-like Python dictionaries.

2.2 Use Cases

To demonstrate and test the efficacy of the developed HydroShare GIS web application, two use cases were explored. This section describes each use case, discusses the common procedures used to setup and test each use case, and lists the criteria for measuring the success of HydroShare GIS in each use case.

Utah Lake Data Integration Use Case. A HydroShare user needs to aggregate all available water quantity and quality data on HydroShare with other related, non-spatial HydroShare datasets for presentation of a water quality study in Utah Lake. The data exists in multiple HydroShare resources of various types, such as a Raster resource, a pair of Geographic Feature resources, a Referenced Time Series resource, a Script resource, and a couple Generic resources.

Hydrologic Terrain Analysis Use Case. The CyberGIS TauDEM app is another app linked to HydroShare that lets users perform advanced hydrologic terrain analyses

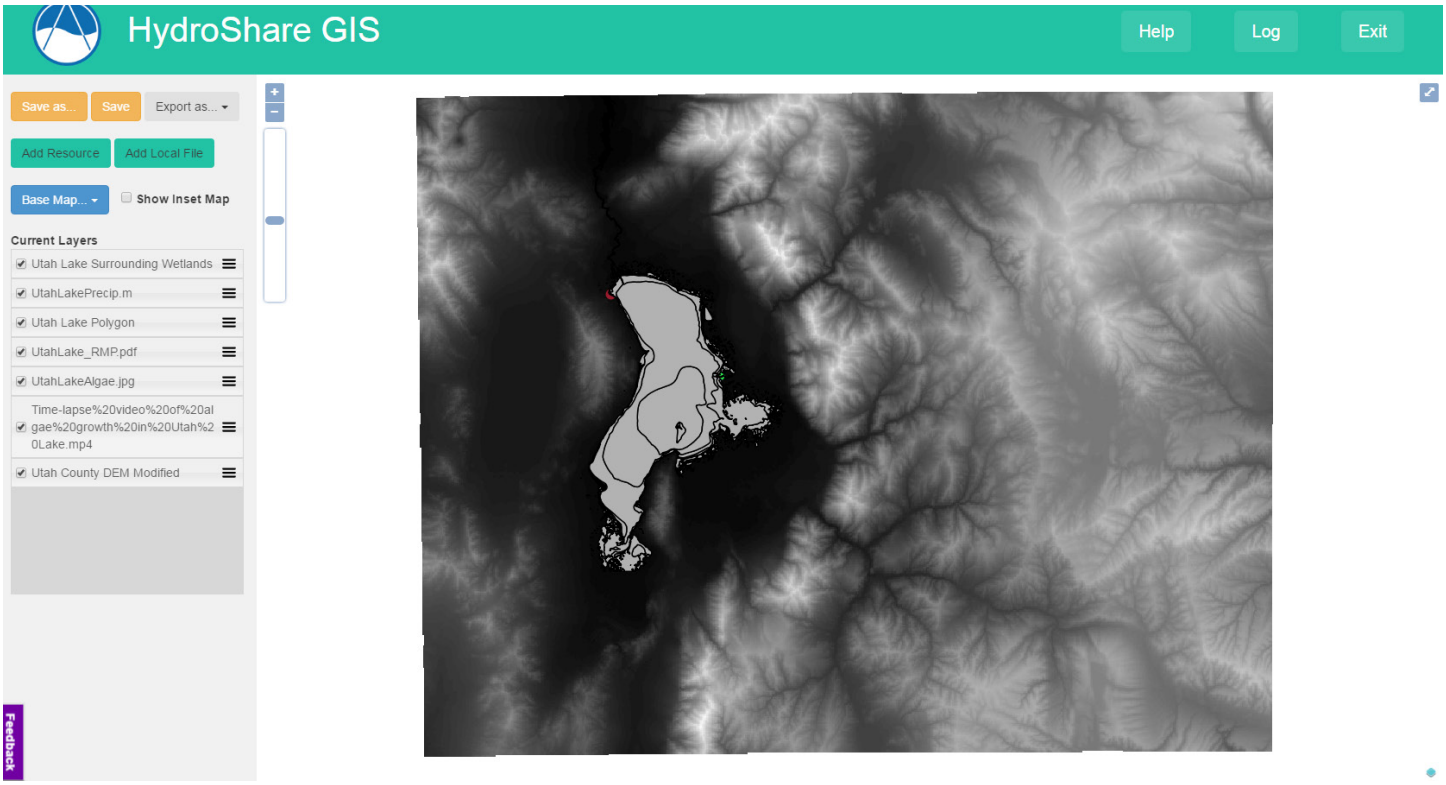

Figure 9: Screenshot of the initial state of HydroShare GIS upon loading data corresponding to the Utah Lake Data Integration Use Case.

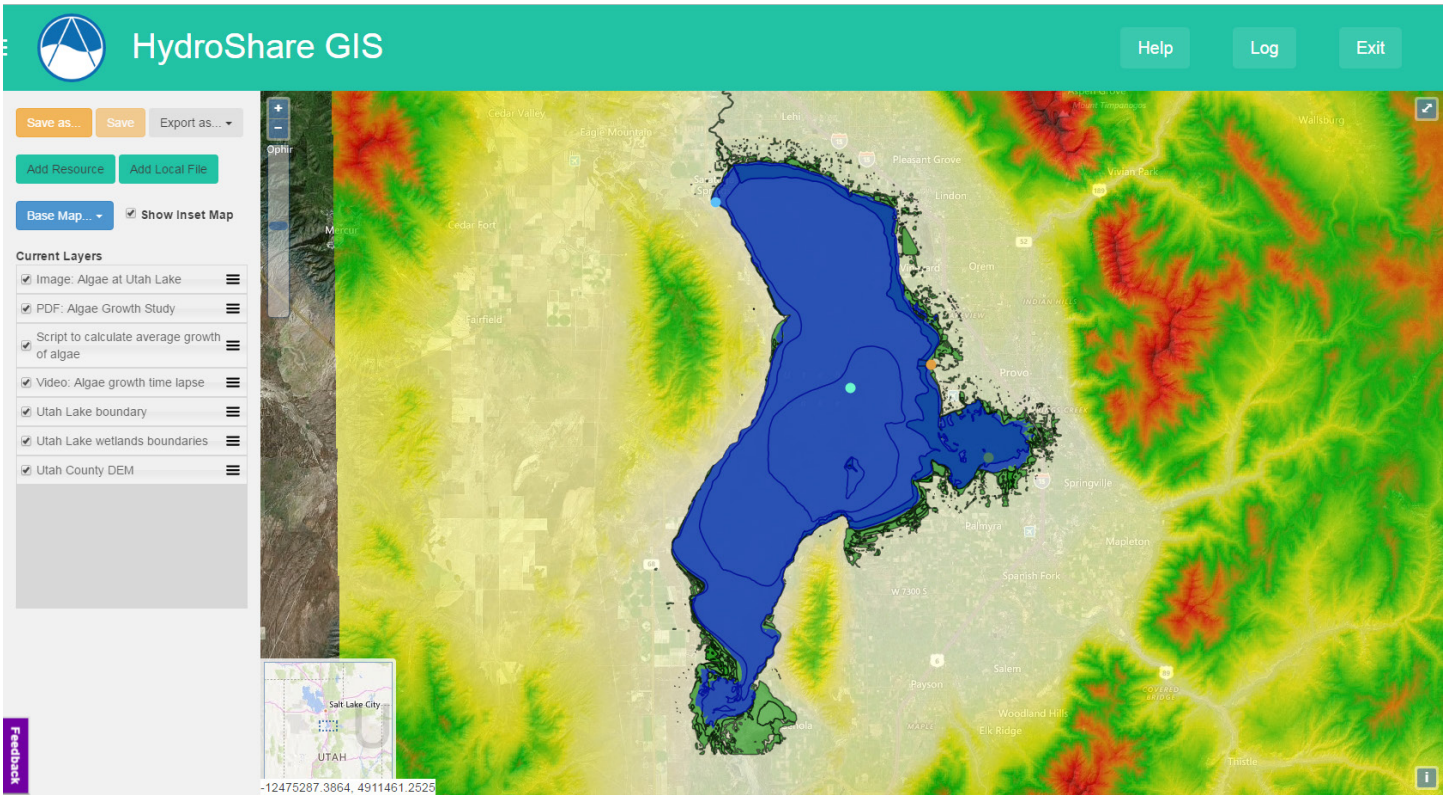

Figure 10: Screenshot of the state of HydroShare GIS after modifying the display properties and symbology of the data corresponding to the Utah Lake Data Integration Use Case.

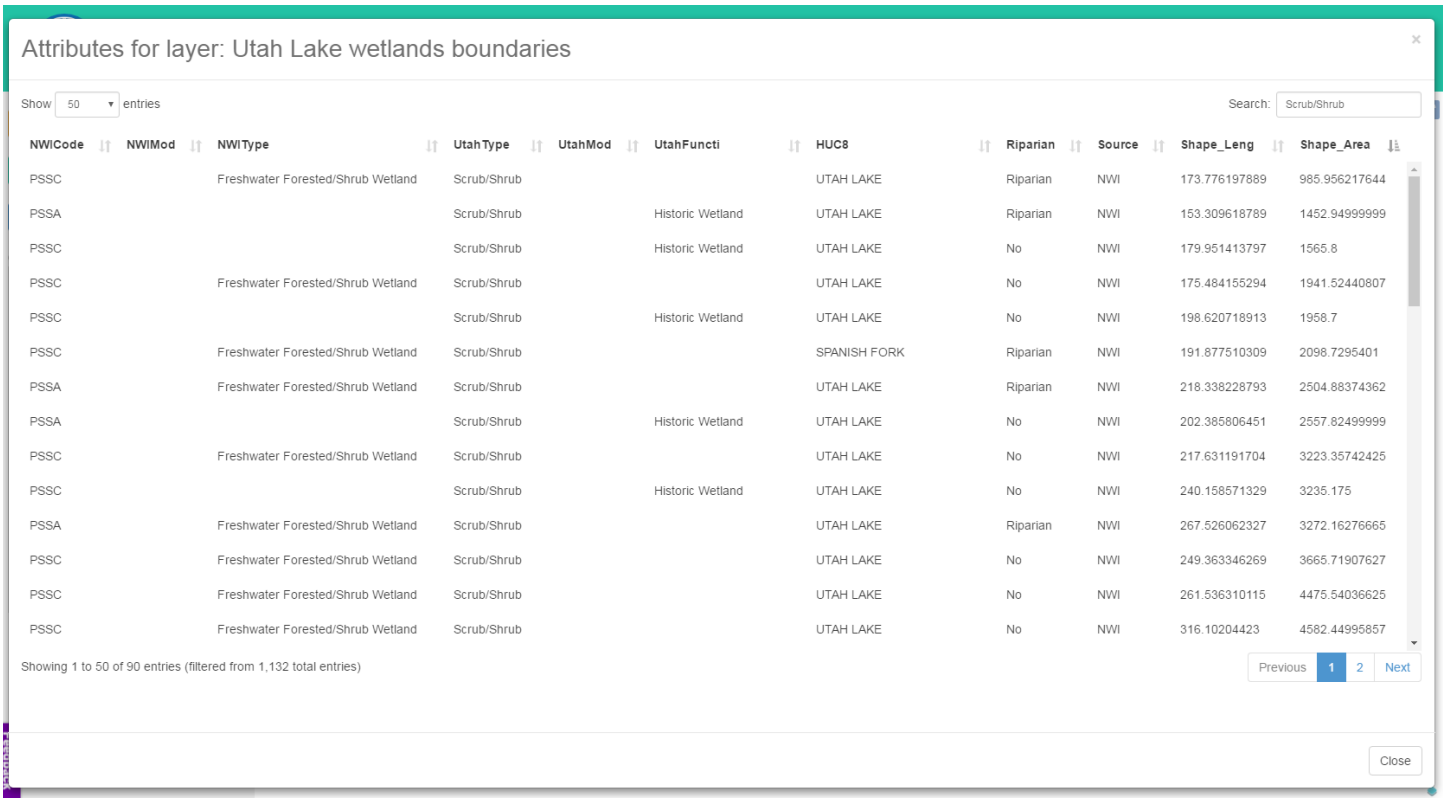

Figure 11: HydroShare GIS attribute table for Utah Lake wetlands layer as part of the Utah Lake Data Integration Use Case.

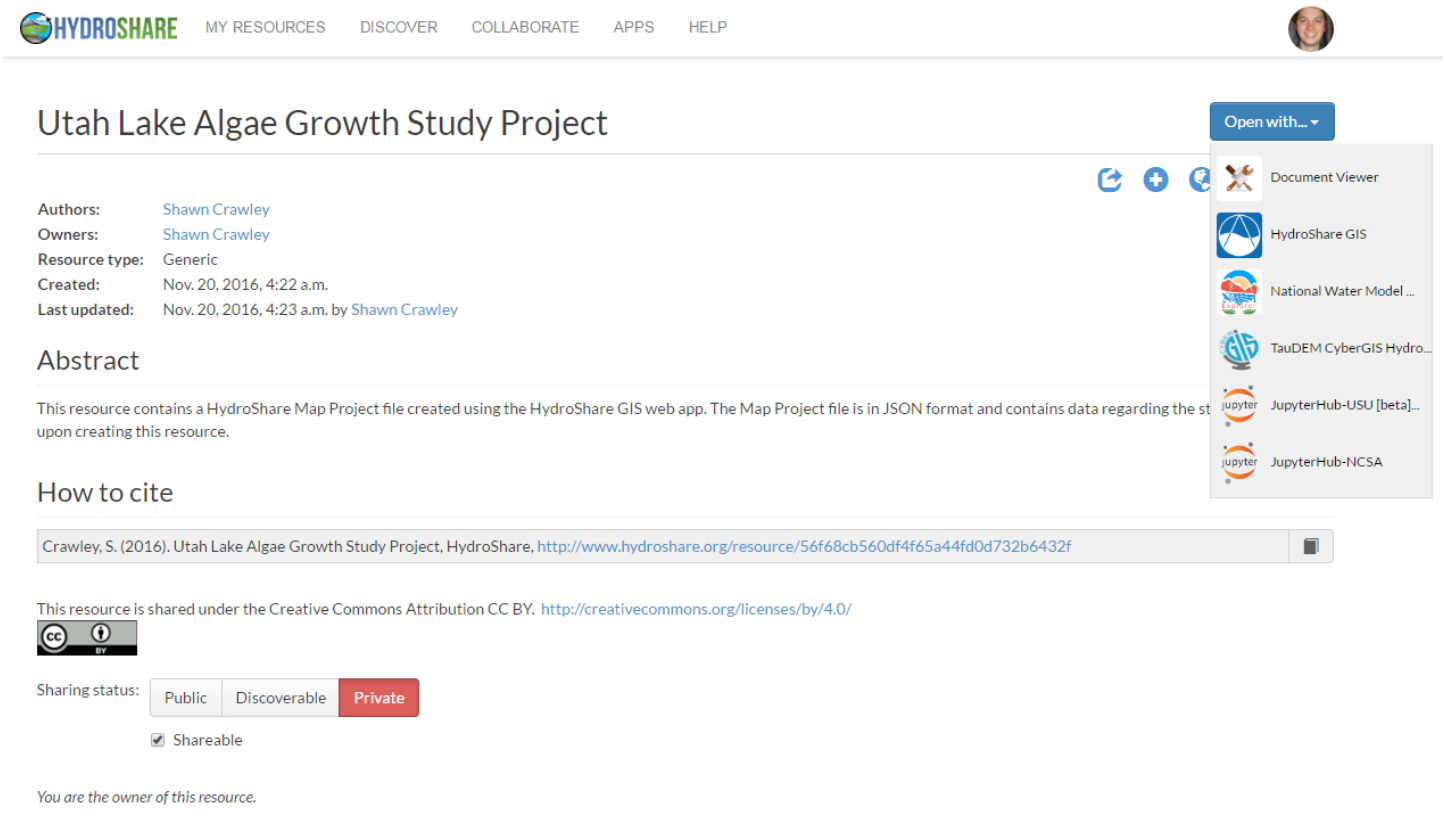

Figure 12: HydroShare landing page for HydroShare GIS project session state saved for the Utah Lake Data Integration Use Case.

in high performance computers at NCSA. A user starts from a digital elevation model in a HydroShare resource and does the following: (1) Opens the CyberGIS app, (2) selects the terrain analysis products to be computed, (3) configures any needed input parameters, and (4) submits for computation. Once the results are computed on the NCSA supercomputer, the results are saved back into HydroShare in a resource containing many files. The use case for HydroShare GIS is to organize and visualize these files, which are of varying types such as TIFs, shapefiles, text files, and a bash script.

For each use case, desired resources were loaded into a HydroShare GIS session either from its associated landing page on HydroShare by using the "Open with…" dropdown and selecting the "HydroShare GIS" option, or from within a fresh, blank instance of HydroShare GIS by using either the "Add Resource" button or the "Add Local File" button. Once all desired resources were loaded into the app, the current layer list entries were first reordered to change the rendering order of their corresponding layers. Then, the symbology of each layer was modified using the "Modify symbology" button appearing in the context menu of the layer's corresponding list item in the current layers list to customize the visualization of the layers. Then, the resulting state of the app was saved to HydroShare as a new resource using the "Save as…" button. Next, the newly-created project resource was located in HydroShare and then loaded into a new, separate instance of the app by using the "Open with…" dropdown and selecting "HydroShare GIS." Finally, these two existing instances of the app were compared.

Success of the HydroShare GIS app in each use case is determined by its ability to:

1. Accurately load and display each spatial dataset chosen from HydroShare as a distinct layer on the map.
2. Accurately load and display each spatial dataset uploaded from the user's local file system as a distinct layer on the map.
3. Create a new HydroShare resource for each spatial dataset uploaded from the user's local file system.
4. Provide a layer list item in the current layers list for each distinct spatial dataset layer.
5. Provide a relevant layer context menu for each layer list item.
6. Accurately update symbology for each spatial dataset layer, according to user specification, upon request.
7. Accurately display a legend for each layer upon request.
8. Accurately display an attribute table (where applicable) upon request.
9. Accurately remove each layer list item and its associated spatial dataset layer upon request.
10. Accurately display an updated (edited) layer name for each layer list item.
11. Accurately load the HydroShare landing page for the corresponding resource of each spatial dataset layer, upon request.
12. Save the app state (project file) back to HydroShare as a new resource.
13. Load the project file from the HydroShare resource landing page into a new HydroShare GIS instance whose state is indistinguishable from the app instance that created the project file resource.

3.0 Results

This section discusses the results of the HydroShare GIS web application development process in terms of its software implementation and use cases.

3.1 Software Implementation

HydroShare GIS, as it appears on the HydroShare Apps portal, can be seen in Figure 7. Clicking on its corresponding icon will launch the app, whereupon the browser will redirect to the application's main screen, which should resemble Figure 3. As compared to the UI Mockup in Figure 3, the final design was extremely similar. There were only minimal layout modifications made and a few additional features added that were not included in the original scope and plans.

3.2 Use Cases

For the Utah Lake Data Integration use case, a fresh instance of HydroShare GIS was initiated from the HydroShare Apps Portal by clicking the HydroShare GIS icon. From there, the "Add Resource" button was used to locate and add all of the desired resources corresponding to Utah Lake. This was simplified by filtering the results through a search for "Utah Lake" from within the "Add Resource" modal dialog that pops up when clicking the button.

Figure 9 shows the state of HydroShare GIS upon loading all of the data corresponding to the Utah Lake

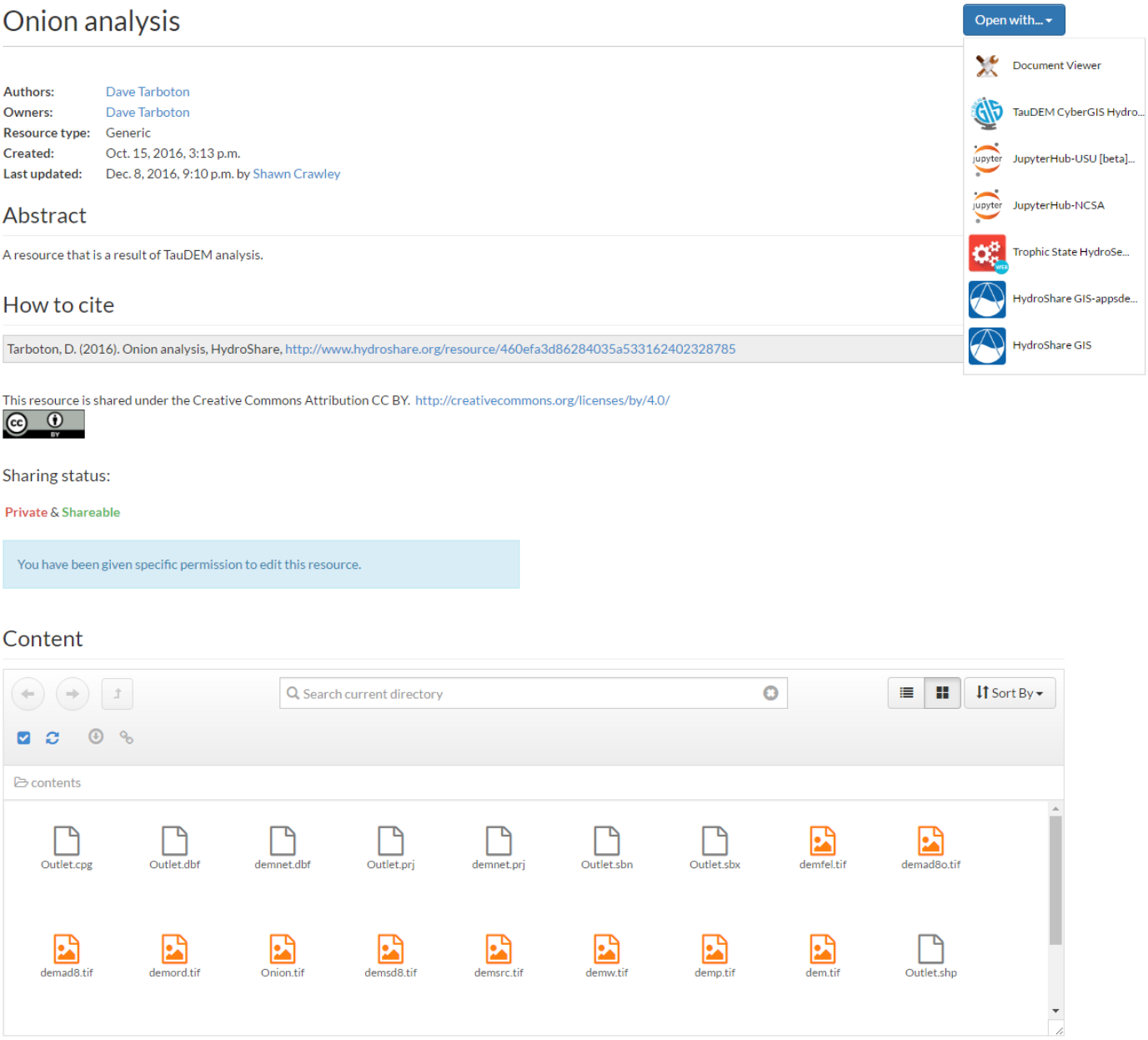

Figure 13: HydroShare landing page of a single Generic resource which was visualized in HydroShare GIS as part of the Hydrologic Terrain Analysis Use Case.

Data Integration Use Case. Many of the layers could not be seen on initial load due to the order in which they were initially loaded and rendered by the app. The topmost layers in the current layers list are rendered last and overlay the bottom-most layers. Also, each layer was displayed using a default color scheme.

It was possible to change the rendering order of the layers by reordering their corresponding entries in the current layers list. In this case it made sense that the raster be rendered first (the bottom-most layer), followed by the polygon shapefiles, and then finally the point shapefiles. The symbology of certain layers was changed to enhance their understandability and meaning. For example, the symbology of the shapefile representing the lake was modified to have a blue fill color, which is more indicative of the water that it represents.

The display names of each layer were also changed as desired to reflect which layer in the map it corresponds to, since neither the HydroShare resource name nor its corresponding file name was adequate. A base map and inset map were also added to enhance visual understanding and confirm that the layers were all rendered in their proper geospatial location. This resulting state of HydroShare GIS can be seen in Figure 10.

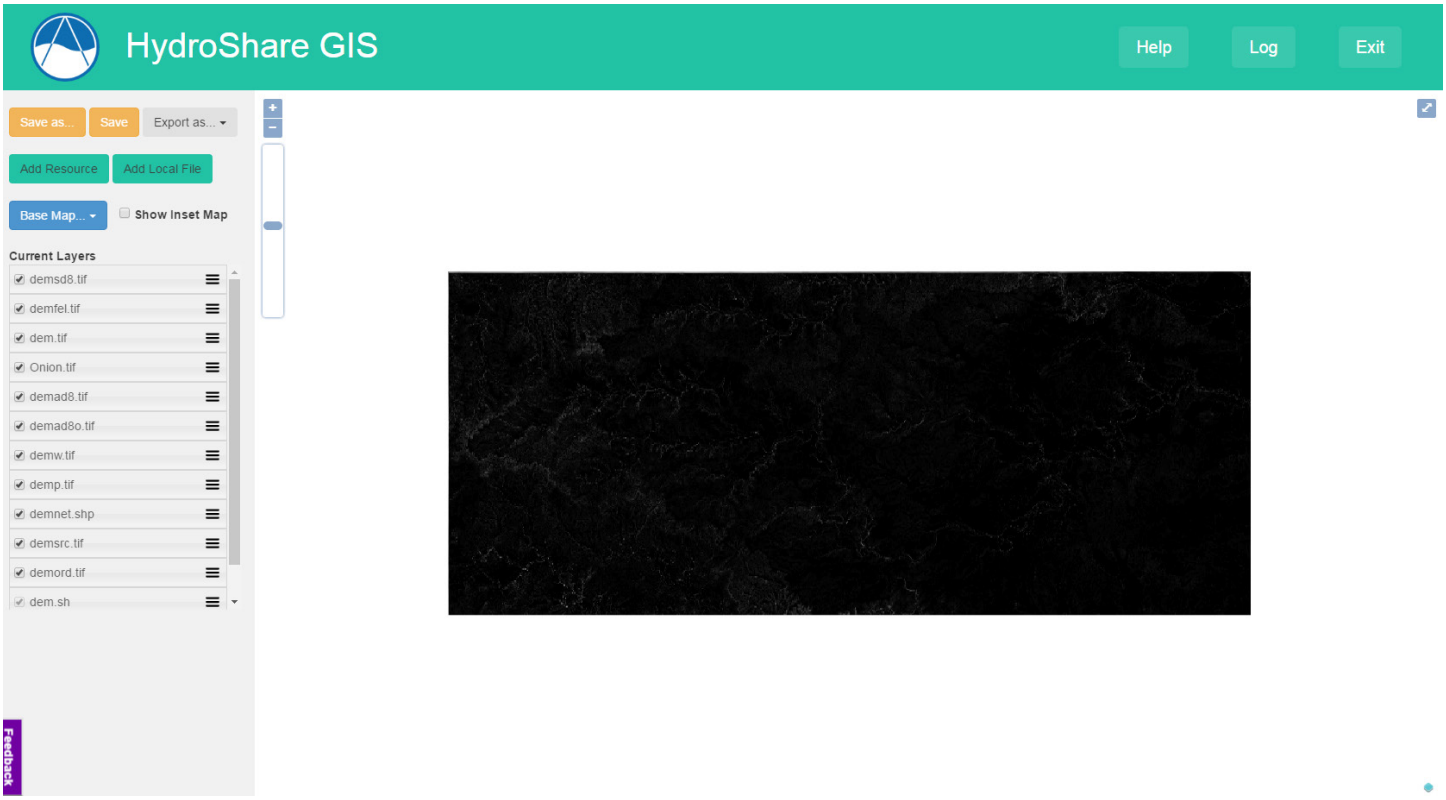

Figure 14: HydroShare GIS session upon successfully loading files associated with the Hydrologic Terrain Analysis Use Case.

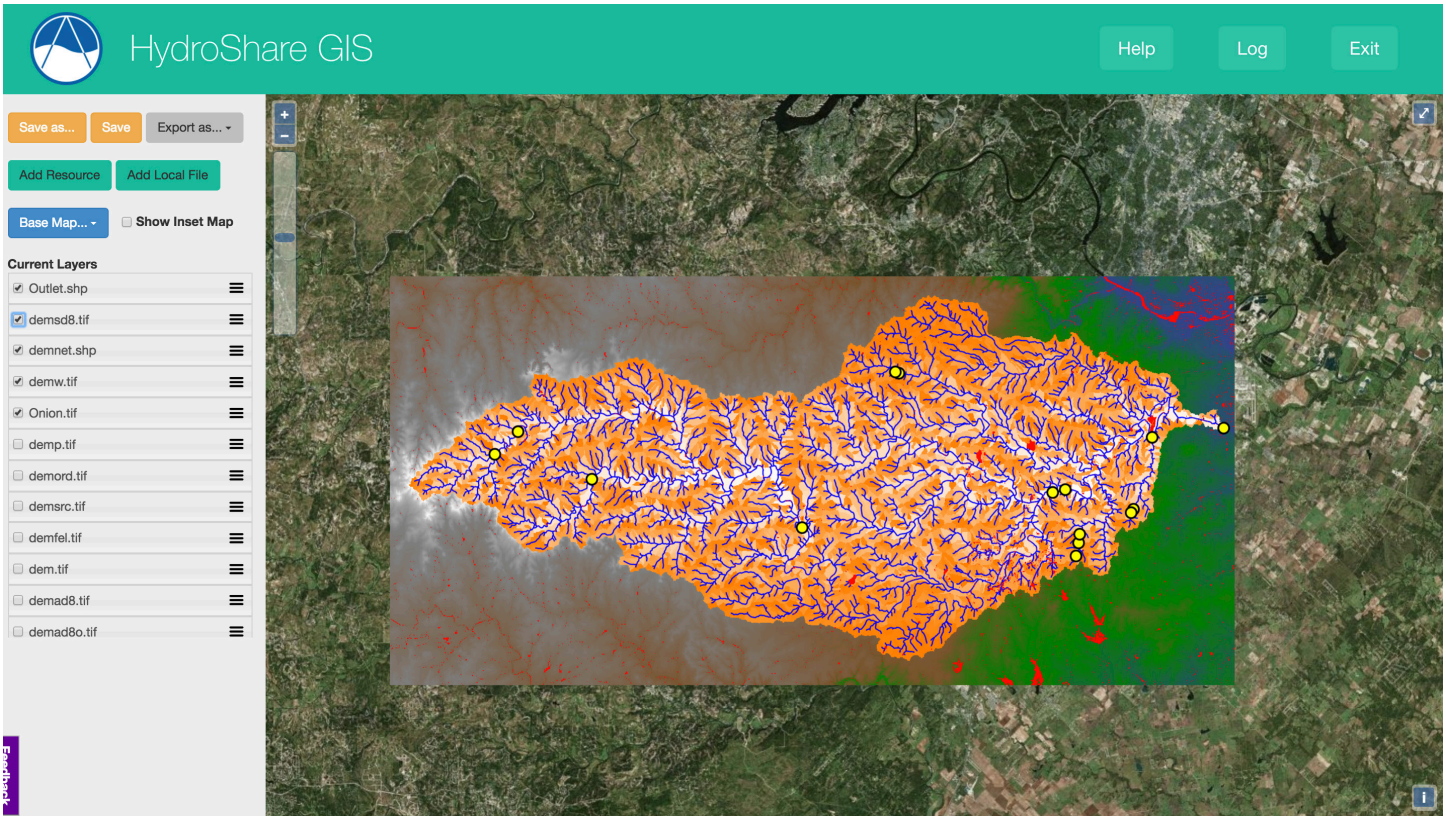

Figure 15: Screenshot of the resulting state of HydroShare GIS after modifying the display properties and symbology as part of the Hydrologic Terrain Analysis Use Case.

Each of the Geographic Feature resources (shapefiles) had a "View attribute table" option in its context menu (triggered by the hamburger icon). Figure 11 shows the actual attribute table that was generated for the layer titled "Utah Lake wetlands boundaries." This table was filtered by the "Shape_Area" column (from smallest to largest) and filtered by a search for "Scrub/Shrub".

For the Hydrologic Terrain Analysis Use Case there was only one resource to be visualized using HydroShare GIS, but this single resource was a Generic resource containing 22 separate files. Among these files were many raster datasets (TIF extension), shapefiles (a set of SHP, SHX, DBF, and PRJ extensions), and a shell script (SH extension). Because only one resource would be added to HydroShare GIS, this was done from the "Open with…" button on the resource's own landing page, which can be seen in Figure 13.

It took about a minute for HydroShare GIS to process and render all of the resource files, but did so successfully. Instead of loading the 22 separate files each as their own layer, the app successfully identified harvested all of the corresponding shapefile components and loaded each set into a single shapefile layer. In the case of the shell script, since this file was part of the resource, it too was added to the current list. However, because it was not an inherently spatial file and lacked any spatial component metadata, it had no corresponding map layer. To make that clear, the layer visibility checkbox is disabled for this particular entry in the current layers list.

Of the 22 files contained in the Generic resource, 13 total entries were added to the current layers list, 12 of which had corresponding layers that were added to the map. The state of HydroShare GIS immediately following the successful loading of the resource can be seen in Figure 14. As with the previous case study, the initial display of all rendered layers was not very interesting or helpful since the last layer to load was a large raster file that was then rendered on top of all other layers

As with the previous use case, the user proceeded to reorder each entry in the current layers list and modify each layer's symbology until a more interesting and useful map display was achieved. This involved toggling layers on and off or moving them closer to the bottom of the current layers list to determine which other layers they were covering on the map. Moving the shapefiles to the top and turning off most raster layers established the best configuration. It is important to note that it would not ever be possible to view all these layers at once because there are many raster layers that share the same extents.

Every entry in the current layers list correctly provided relevant options in its context menu based on the type of file it represented. Each entry provided an option for modifying its own display name text, viewing the web page of its associated resource in HydroShare, and removing itself and its associated layer from the project. Each entry representing a spatially based file with an associated layer had options to modify its symbology, view its legend, and zoom to its rendered extents. Each shapefile had the option to view its associated attribute table. The single entry representing the non-spatial resource had the unique option to view its associated file.

As with the previous use case, once the layers were modified to the desired state, the HydroShare GIS session state was successfully saved back to HydroShare as a Generic resource containing a map project file that was able to be successfully re-opened in HydroShare GIS from its landing page in HydroShare.

This use case demonstrated that HydroShare GIS is able to correctly identify files based on their type when loading them from a Generic resource. Though the app will add entries for all file types (both spatial and non-spatial) to the current layers list, it will provide different context menu options and appropriately enable or disable the checkbox to toggle the associated layer's visibility based on the resource's file type.

4.0 Conclusion

We developed a cloud-based application called HydroShare GIS to provide HydroShare users a cloud-based tool for visualizing their spatial data resources. It was designed using the model-view-controller (MVC) pattern and implemented primarily using Tethys Platform. The app is currently deployed on the HydroShare Apps Portal (http://apps.hydroshare.org).

HydroShare GIS is able to recognize and load spatial data from HydroShare whether contained in an inherently spatial resource type (Raster, Geographic Feature), or a Generic resource. It will also recognize non-spatial data contained in a Generic resource and provide a unique, relevant context menu with appropriate options for interacting with it. The state of any HydroShare GIS session can be saved back to HydroShare as a special map project file stored within a Generic resource. Once the resource is created, the saved map project can be re-launched from the "Open with…" option on its landing page within HydroShare.

HydroShare GIS has shown that there are many open source technologies available that can be lever-

aged to develop cloud-based applications and tools for visualizing and manipulating spatial data. The same architecture, patterns, and methods demonstrated by HydroShare GIS could be implemented elsewhere, built upon, and improved. The surface is just now only being scratched. Almost all of these technologies implemented are still under continual development and are releasing new features that will open up new opportunities for both HydroShare GIS and other similar projects.

Acknowledgements

This work was supported by the National Science Foundation under collaborative grants ACI 1148453 and 1148090 for the development of HydroShare (http://www.hydroshare.org). 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.

Software Availability

HydroShare GIS, 1.0.0, Shawn Crawley, MIT License, 08/30/2016, Source code available at https://github.com/hydroshare/tethysapp-hydroshare_gis.

References

Ames, D., Z. Li, X. Qiao, D. G. Tarboton, R. Idaszak, J. Horsburgh, V. Merwade, B. Miles, N. Swain, and R. Lineberger (2015), Web-Based Data Visualization and Analysis using HydroShare and the Open Source Tethys Platform, paper presented at AGU Fall Meeting Abstracts.

Covili, J. J. (2016), Going Google: Powerful tools for 21st century learning, Corwin Press.

Horsburgh, J. S., D. G. Tarboton, R. Idaszak, D. P. Ames, J. L. Goodall, V. Merwade, A. Couch, R. P. Hooper, P. Dash, and M. J. Stealey (2016), HydroShare: Promoting Collaborative Publication, Interoperability, and Reuse of Hydrologic Data and Research Products.

Idaszak, R., D. G. Tarboton, H. Yi, L. Christopherson, M. J. Stealey, B. Miles, P. Dash, A. Couch, C. Spealman, and J. S. Horsburgh (2016), 10 HydroShare–A Case Study of the Application of Modern Software Engineering to a Large Distributed Federally-Funded Scientific Software Development Project, in Software Engineering for Science, edited, pp. 217-234, CRC Press.

Mikowski, M. S., and J. C. Powell (2013), Single page web applications, B and W.

Miller, M. (2008), Cloud computing: Web-based applications that change the way you work and collaborate online, Que publishing.

Swain, N., S. Christensen, J. Nelson, and N. Jones (2015), Tethys Platform: A Platform for Water Resources Modeling and Decision Support Web Apps.

Swain, N. R., S. D. Christensen, A. D. Snow, H. Dolder, G. Espinoza-Dávalos, E. Goharian, N. L. Jones, E. J. Nelson, D. P. Ames, and S. J. Burian (2016), A new open source platform for lowering the barrier for environmental web app development, Environmental Modelling & Software, 85, 11-26.

Wang, S. (2010), A CyberGIS framework for the synthesis of cyberinfrastructure, GIS, and spatial analysis, Annals of the Association of American Geographers, 100(3), 535-557.

Wang, S. (2016), CyberGIS and spatial data science, GeoJournal, 81(6), 965-968.

Wang, S., L. Anselin, B. Bhaduri, C. Crosby, M. F. Goodchild, Y. Liu, and T. L. Nyerges (2013), CyberGIS software: a synthetic review and integration roadmap, International Journal of Geographical Information Science, 27(11), 2122-2145.

Yin, J., Y. Gao, and S. Wang (2017), CyberGIS-enabled urban sensing from volunteered citizen participation using mobile devices, in Seeing Cities Through Big Data, edited, pp. 83-96, Springer.