CEREO Living Atlas

Final Report

Center for Environmental Research, Education, and Outreach (CEREO)

Living Atlas Development Team:

Yaru Gao

Zachary Garoutte

Jonathan Simmons

Mentor:

Ananth Jillepalli

CptS 423 Software Design Project II

Fall 2025

TABLE OF CONTENTS

I. Introduction 4
II. Team Members & Bios 4
III. Project Requirements Specification 5
II.1. Project Stakeholders 5
II.2. Use Cases 5
II.3. Functional Requirements 14
II.4. Non-Functional Requirements 16
II.5. Standards 17
III. Software Design - From Solution Approach 18
III.1. Architecture Design 18
III.1.1. Overview 19
III.1.2. Subsystem Decomposition 19
III.2. Data design 20
III.3. User Interface Design 20
III.4. Constraints 21
IV. Test Case Specifications and Results 21
IV.1. Testing Overview 21
IV.2. Environment Requirements 23
IV.3. Test Results 24
V. Projects and Tools used, Libraries, and Frameworks 24
VI. Description of Final Prototype 25
VII. Social Responsibility and Broader Impacts 30
VIII. Product Delivery Status 30
IX. Conclusions and Future Work 31
IX.1. Limitations and Recommendations 31
IX.2. Future Work 32
X. Acknowledgements 32
XI. Glossary 33
XII. References 35
XIII. Appendix A – Team Information 35
XIV. Appendix B - Example Testing Strategy Reporting 36
XV. Appendix C - Project Management 37

I. Introduction

The CEREO Living Atlas is a web-based platform built through collaboration with the Center for Environmental Research, Education, and Outreach (CEREO) at Washington State University. It's intended as a flexible tool for a wide community—researchers, tribal groups, educators, students, and government agencies alike—making it easier to gather, view, and share environmental data relevant to the Pacific Northwest. Providing users with a single, map-focused interface to upload and explore geospatial information. From tracking water quality to monitoring restoration work and community engagement, the platform is geared toward surfacing patterns, highlighting case studies, and guiding decisions with clarity.

For years, environmental datasets were fragmented—tucked away in separate systems, hard to access, and rarely interoperable between institutions. That's a big part of why the Living Atlas came to be. It offers a public-facing, visual interface where users can submit "cards" packed with metadata, photos, location data, and even links to research. These cards are searchable and filterable by category, tag, or custom fields, giving users a more intuitive and visual way to navigate complex datasets.

The platform didn't start out this way. What began as a simple data viewer gradually grew into a fully interactive application with strong backend support. During its alpha phase, a number of useful features were introduced: things like account logins, role-specific permissions, bookmarks, sortable cards, thumbnail previews, and an easier submission workflow for spatial data.

Throughout development, CEREO stakeholders—including faculty, student researchers, and external collaborators—voiced a clear need: the platform had to be user-friendly for those without technical backgrounds, but still robust enough for deeper academic or policy-related work. Their input pushed the team to focus on accessibility and scalability while respecting data ownership. These concerns, in many ways, shaped how the system functions today.

II. Team Members & Bios

Zachary Garoutte is a software engineering student. He is experienced in working with C, C++, C#, Python, Java, SQL, HTML, and CSS. For this project, his responsibilities have been to create a new database and cloud data service for the project, and to add the ability for users to upload thumbnails to cards in the Atlas, and to add downloadable file attachments to cards.

Yaru Gao is a computer science student. He is experienced in working with C, C++, Python, Java, HTML, CSS, and JavaScript. For this project, his responsibilities have been to refine the frontend UI, fix the bugs of the password reset feature and card display feature, and implement a bookmarking feature that users can use to add, remove or sort their favorite cards.

Jonathan Simmons is a computer science student. He is experienced in C, C++, C#, Python, and Java. For this project, his responsibilities have been to add a sorting feature that can organize the cards in the Atlas by different criteria, such as closest to current location and most recently added, and to add the ability for users to attach files to cards in the Atlas.

III. Project Requirements Specification

II.1. Project Stakeholders

The primary stakeholder is the **Center for Environmental Research, Education and Outreach** (CEREO) at Washington State University. CEREO is an organization dedicated to addressing environmental challenges through research and educational support, aiming to develop meaningful solutions for critical environmental issues. Currently, CEREO relies on the Living Atlas as a key research tool. They desire an application that is more functional and user-friendly, enabling broader adoption and improved collaboration among users.

Other stakeholders include both current and future researchers working with CEREO. They could benefit from a more efficient and robust application to enhance their research and collaboration.

II.2. Use Cases

The use cases for the application are based on a role-based access control system. Any user has access to the map data and can use search functionality on this data. A registered user is a user who has created an account on the website, and these users can login to their account and bookmark data. An authorized user is a user with authority to add data to the map, and view, edit, or remove this data later. An administrator has authority to grant authorization to add data and can edit or remove any current data on the map. Administrative functions are currently planned to be done through the backend, but administrator login may be implemented in the future.

A use-case UML diagram for the Living Atlas is provided in Appendix D.

Add Geospatial Data To Map

Actors	Authorized user
Preconditions	• Be logged in to an authorized user account • On main map page
Postconditions	• Geospatial data and attached information card is added to database and is publicly displayed on map
Path	1. Authorized user clicks Upload Data option from homepage

	2. User enters geospatial information such as geospatial data types 3. User selects a pin on the map to select data location, or manually enters coordinates 4. User enters relevant information such as study results to attach to these coordinates as a card 5. Optionally upload photo to display on card 6. User submits data addition 7. Data is displayed on map at given coordinates and new card is added
Related Requirements	III.3.1 User Management: Role-Based Access Control III.3.2 Data Management: Geospatial Data Loading & Editing III.3.3 Map & Visualization Features: Interactive Mapping Interface III.3.3 Map & Visualization Features: Picture Data Upload & Display

View Added Geospatial Data

Actors	Authorized user
Preconditions	• Be logged in to an authorized user account • On main map page
Postconditions	• Information cards corresponding to the geospatial data that was previously added by the logged in user are displayed
Path	1. Authorized user clicks View Added Data option from homepage 2. Information cards added by the user are displayed
Related Requirements	III.3.1 User Management: Role-Based Access Control III.3.3 Map & Visualization Features: Interactive Mapping Interface

Edit Added Geospatial Data

Actors	Authorized user
Preconditions	• Be logged in to an authorized user account • On main map page
Postconditions	• Edit the text and/or photo on an information card previously added by the logged in user

Path	1. Authorized user clicks View Added Data option from homepage 2. Information cards added by the user are displayed 3. Click Edit option on card to be edited 4. User enters updated information for card 5. Optionally user uploads new photo for card 6. User submits edit 7. Information and photo is publicly updated in real time on the card
Related Requirements	III.3.1 User Management: Role-Based Access Control III.3.2 Data Management: Geospatial Data Loading & Editing III.3.3 Map & Visualization Features: Interactive Mapping Interface III.3.3 Map & Visualization Features: Picture Data Upload & Display

Remove Geospatial Data From Map

Actors	Authorized user
Preconditions	• Be logged in to an authorized user account • On main map page
Postconditions	• Geospatial data and attached information card previously added by the logged in user is removed from the map
Path	1. Authorized user clicks View Added Data option from homepage 2. Information cards added by the user are displayed 3. Click Delete option on card to be deleted 4. User confirms deletion 5. Geospatial data on map is removed from map and the attached card is removed
Related Requirements	III.3.1 User Management: Role-Based Access Control III.3.3 Map & Visualization Features: Interactive Mapping Interface

Bookmark Data

Actors	Registered user of any authorization
Preconditions	• Be logged in to an account • On main map page
Postconditions	• Card(s) is added to user's favorites

Path	1. User clicked on the book button on the right sidebar or the double arrow button in the center-right of the screen to open the card container 2. Click on the empty bookmark icon on the top right corner of a card to bookmark it. 3. Card is added to user's favorites
Related Requirements	III.3.2 Data Management: Data Filtering & Search

Unbookmark Data

Actors	Registered user of any authorization
Preconditions	• Be logged in to an account • On main map page • Card(s) is bookmarked by user previously
Postconditions	• Card(s) is removed from user's favorites
Path	1. User clicked on the book button on the right sidebar or the double arrow button in the center-right of the screen to open the card container 2. Click on the solid bookmark icon on the top right corner of a card to unbookmark it. 3. Card is removed from user's favorites
Related Requirements	III.3.2 Data Management: Data Filtering & Search

Access All Bookmarked Data

Actors	Registered user of any authorization
Preconditions	• Be logged in to an account • On main map page
Postconditions	• Information cards previously bookmarked by the logged in user are displayed
Path	1. User clicks View Bookmarks option from homepage 2. Information cards bookmarked by the user are displayed
Related Requirements	III.3.2 Data Management: Data Filtering & Search

Search (updated)

Actors	Any user
Preconditions	• On main map page
Postconditions	• Display information cards based on date, location, and other sort and filter options
Path	1. Optionally click sort option to sort list of cards 2. Optionally click filter option to filter list of cards 3. Display cards best matching the search criteria
Alternate Path	1. User clicks search bar 2. Type keywords to be matched with cards in database 3. Display cards relating to keywords 4. Optionally click sort option to sort list of cards 5. Optionally click filter option to filter list of cards 6. Display cards best matching the search criteria
Related Requirements	III.3.2 Data Management: Data Filtering & Search

Reset Password

Actors	Registered user of any authorization
Preconditions	• On login page or profile page
Postconditions	• The user's password is updated
Path	1. User clicks Forgot Password option 2. A one-time verification link is sent to the email attached to the user's account 3. Click link in email account 4. Enter new password 5. User submits password 6. Login password is updated to new password
Related Requirements	III.3.1 User Management: User Registration & Authentication

Load Spatial Data onto Map

Actors	Any user
Preconditions	• On homepage
Postconditions	• The interactive layers of ArcGIS data are displayed on the map with popup modal ready to open.

Path	1. User clicks on the Browse GIS Data button featured as an earth icon on the left sidebar to open the upload panel. 2. Click to expand the main folder or service folder to see the servers/layers 3. Click on the load button of a service(s) or the checkbox of a layer(s) to load interactive layers to the map 4. Colored areas of loaded interactive layers will gradually appear on the map in a short while
Related Requirements	III.3.3 Map & Visualization Features: Interactive Mapping Interface

Remove Loaded Spatial Data from Map

Actors	Any user
Preconditions	• On homepage • Service(s) or layer(s) are loaded onto the map (see 'Load Spatial Data onto Map')
Postconditions	• The interactive layers of ArcGIS data are removed from the map.
Path	1. Click on the Browse GIS Data button on the left sidebar if the upload panel is not open. 2. Click to expand the main folder or service folder to see the servers/layers 3. Click on the remove button of a service(s) or the checkbox of a layer(s) to add it to the map 4. Removes loaded service(s)/layer(s) from the map
Related Requirements	III.3.3 Map & Visualization Features: Interactive Mapping Interface

View Layer Information

Actors	Any user
Preconditions	• Colored areas are fully displayed on map, see 'Load Spatial Data onto Map'
Postconditions	• A pop-up modal containing information about the layer will be displayed on the map

Path	1. User clicks on the colored areas of loaded interactive layers within the map. 2. A pop-up modal that contains detailed information of a layer(s) is open.
Related Requirements	III.3.2 Data Management: Data Filtering & Search III.3.3 Map & Visualization Features: Interactive Mapping Interface

View Service/Layer Information

Actors	Any user
Preconditions	• The upload Panel is open with the folder expanded
Postconditions	• A pop-up modal containing information about the service/layer will be displayed on the right side of the upload panel.
Path	1. User clicks on the learn-more button featuring an “..." icon with respect to each service/layer. 2. A pop-up modal that contains detailed information of a service(s)/layer(s) is open.
Related Requirements	III.3.2 Data Management: Data Filtering & Search III.3.3 Map & Visualization Features: Interactive Mapping Interface

Search for a Folder(s)/Service(s)/Layer(s) & Sublayer(s)

Actors	Any user
Preconditions	• On Upload Panel
Postconditions	• The desired folders, services, layers, or sublayers are displayed in the upload panel.
Path	1. Click on the Browse GIS Data button on the left sidebar if the upload panel is not open. 2. Enter keyword(s) into the search bar on the top corner of upload panel 3. Click on the search icon 4. Folder(s)/Service(s)/Layer(s)/Sublayers that match the keyword(s) will be displayed in the upload panel 5. Optionally click on the close button next to the search icon to clear search and reset upload panel

Related Requirements	III.3.2 Data Management: Data Filtering & Search

Show Loaded Service(s)/Layer(s) & Sub-Layers Only

Actors	Any user
Preconditions	• User opened upload panel • Service(s) or layer(s) are loaded onto the map (see 'Load Spatial Data onto Map')
Postconditions	• Only loaded service(s) or layer(s) are be displayed in the upload panel
Path	1. Click on the Browse GIS Data button on the left sidebar if the upload panel is not open. 2. Check/Uncheck the checkbox below the search bar that has text "Show only services added to map" 3. The upload panel will display the loaded service(s) or layer(s) only
Related Requirements	III.3.2 Data Management: Data Filtering & Search

Hide/Display User Defined or Built-in Layers

Actors	Any user
Preconditions	• On homepage
Postconditions	• Card/Colored Areas will be hidden/displayed on the map.
Path	1. User clicks on the layer button featured as a layer icon on the right sidebar to open the layer panel. 2. Check/Uncheck the checkbox of a card or user-added colored areas. 3. The selected Cards or Colored areas will be removed from the map or displayed on the map.
Related Requirements	III.3.3 Map & Visualization Features: Interactive Mapping Interface III.3.2 Data Management: Data Filtering & Search

Move Current View to Location of a Card

Actors	Any user
Preconditions	• On homepage

	• Card container is open
Postconditions	• User's current view is moved to the location of the card he clicked.
Path	1. User clicks the empty space on a card 2. User's current view will be moved to the location of that card
Related Requirements	III.3.3 Map & Visualization Features: Interactive Mapping Interface

Toggle to display ArcGIS data of different states

Actors	Any user
Preconditions	• On homepage • The upload panel is open
Postconditions	• User sees the ArcGIS data of either Washington, Idaho or Oregon within the upload panel.
Path	1. User locate the state menu attached to the upper right edge of the upload panel. 2. User clicks on either one of the options (WA, ID, OR) in the state menu. 3. The state menu display the ArcGIS data of the corresponding state (Washington, Idaho, or Oregon).
Related Requirements	III.3.2 Data Management: Data Filtering & Search III.3.3 Map & Visualization Features: Interactive Mapping Interface

Attach Downloadable File to Card

Actors	Authorized user
Preconditions	• Be logged in to an authorized user account • On main map page
Postconditions	• Uploaded file(s) are attached to the selected card. • Files are securely stored in cloud storage. • Downloadable links are displayed on the card.
Path	1. Authorized user clicks **Add Data** or **Edit Card** option from homepage. 2. User selects one or more files from their local device to attach. 3. System automatically compresses and uploads the files to Google Cloud Storage.

	4. Backend generates and saves public download links in the database. 5. Uploaded files appear as downloadable attachments within the card modal once submission is complete.
Related Requirements	**III.3.1 User Management:** Role-Based Access Control **III.3.2 Data Management:** File Upload & Storage **III.3.3 Map & Visualization Features:** Card Display & Download Links

Rename ArcGIS Folders and Services

Actors	Registered User (Admin)
Preconditions	• Be logged in to an authorized user account • The upload panel is open.
Postconditions	• The name of a folder or service is changed.
Path	1. User opens the upload panel and expands the folders to view the services. 2. User double clicks on the name area of a folder or service to choose rename action. 3. User enters the new name for the folder/service. 4. User can press ‘enter’ to save and apply the changes or press ‘escape’ or click empty area to discard the changes.
Related Requirements	**III.3.2 Data Management:** File Upload & Storage

Remove ArcGIS Services

Actors	Registered User (Admin)
Preconditions	• Be logged in to an authorized user account • The upload panel is open.
Postconditions	• A service is removed from the upload panel and moved to the remove services panel
Path	1. User opens the upload panel and expands the folders to view the services. 2. User clicks on the remove button. 3. User confirms removal in the pop-up modal by selecting okay.
Related Requirements	**III.3.2 Data Management:** File Upload & Storage

Update ArcGIS Services

Actors	Any User
Preconditions	• On the homepage • The upload panel is open.
Postconditions	• The service items in the upload panel are synchronized with the items on the ArcGIS server.
Path	1. User opens the upload panel and scrolls down. 2. User clicks on the update button at the bottom of the upload panel. 3. The result of the update is displayed.
Related Requirements	**III.3.2 Data Management:** File Upload & Storage

II.3. Functional Requirements

The CEREO Living Atlas is a web-based application developed to assist researchers, tribal communities, and government agencies in visualizing and sharing critical environmental data. To ensure the system remains accessible, efficient, and scalable, a structured set of functional requirements has been established.

These functional requirements define the system's essential capabilities, outlining what the application must achieve to effectively meet user needs. They are categorized into three primary modules: **user management, data management, and mapping visualization.**

Each requirement is aligned with stakeholder needs and assigned a priority level based on its significance:

- **Priority Level 0:** Essential, non-negotiable functionality.
- **Priority Level 1:** Important but not critical features.
- **Priority Level 2:** Optional enhancements or future upgrades.

III.3.1 User Management

- **User Registration & Authentication:**
  - The system must allow users to create accounts and log in securely.
  - Users should be able to reset passwords via email authentication.
  - Emails must be reliably forwarded from the application’s Gmail account to WSU emails.
  - Source: CEREO’s need for controlled data access and email functionality improvements.

  - **Priority:** Level 0 (Essential)
- **Role-Based Access Control:**
  - The system must differentiate user roles (e.g., researchers, administrators, public users).
  - Certain data upload and modification features must be restricted to authorized users.
  - There is currently no admin login, but this may be required in the future.
  - Source: Stakeholder request for future expansion options.
  - **Priority:** Level 1 (Desirable)

III.3.2 Data Management

- **Geospatial Data Loading & Editing:**
  - Users must be able to upload geospatial files (e.g., shapefiles, GeoJSON) instead of hardcoding data.
  - Alternatively, geospatial data must be able to be loaded directly from the webserver.
  - The system should support additional data types, such as watershed boundaries.
  - Polygons representing geospatial data should be customizable, such as color coding to represent different environmental factors.
  - Source: Client request for flexible spatial data storage and visualization.
  - **Priority:** Level 0 (Essential)
- **Data Filtering & Search:**
  - Users should be able to filter datasets based on date, location, and environmental metrics.
  - A search function must allow users to locate specific data points easily.
  - Users should be able to bookmark data to make the data easily accessible at a later time.
  - Source: Stakeholder need for enhanced data navigation.
  - **Priority:** Level 1 (Desirable)
- **File Upload & Storage**
  - Users must be able to upload one or more files (e.g., `.zip`, `.csv`, `.pdf`, `.docx`, etc.) when creating or editing cards.
  - The system should automatically compress uploaded files for efficient storage while preserving original file types and names.
  - Uploaded files must be stored securely on Google Cloud Storage, with downloadable links displayed in the frontend.
  - 09000Server-side validation should ensure safe uploads and prevent malicious content.
  - **Source:** Client request to allow data attachments directly within the Atlas interface rather than linking external sources.
  - **Priority:** Level 0 (Essential)

III.3.3 Map and Visualization Features

- **Interactive Mapping Interface:**
  - The map must dynamically display geospatial data and be updated in real-time. Users should be able to toggle interactive map layers and view detailed info about them.
  - The map must include a legend of symbol meanings, categories, and active layers.
  - Source: Usability improvement request from researchers and policy analysts.
  - **Priority:** Level 0 (Essential)
- **Picture Data Upload & Display:**
  - Users must be able to upload images associated with geospatial data points.
  - Ensure smoother handling of image uploads to avoid failures.
  - Source: Client feedback on frontend usability.
  - **Priority:** Level 0 (Essential)
- **Performance Optimization**:
  - The website should be able to work faster and handle a larger user base.
  - Reduce delays in loading data points on the map.
  - Source: Client's primary goal for enhancement.
  - **Priority:** Level 0 (Essential)

II.4. Non-Functional Requirements

Non-functional requirements are aspects such as performance, security, reliability, and scalability. While the functional requirements define what the system must do, non-functional requirements determine how efficiently and effectively it operates under different working conditions.

To support its expanding user base and large datasets, the CEREO Living Atlas must ensure data integrity, high availability, and an intuitive user experience. Meeting long-term stakeholder expectations requires a strong focus on efficiency, security compliance, and system extensibility.

These requirements are categorized into key areas, including performance, security, cross-platform compatibility, reliability, and future scalability. Each is assessed and prioritized based on its influence on system usability and long-term sustainability.

III.4.1. Performance & Scalability:

- The system shall support 500+ concurrent users without performance degradation.
- The backend should be optimized to handle large datasets efficiently.
- Source: Client's requirement for scalability.
- Priority: Level 0 (Essential)

III.4.2. Security & Authentication:

- Email forwarding issues must be resolved to allow WSU emails to receive system messages.

- Future improvements should consider adding email-based user verification.
- Source: Client's security concerns.
- Priority: Level 1 (Desirable)

III.4.3. Cross-Platform Compatibility:

- The web application shall function on modern browsers (Chrome, Firefox, Edge).
- The system should support mobile access with a responsive design.
- Source: User accessibility requirements.
- Priority: Level 1 (Desirable)

III.4.4. Reliability & Uptime:

- The system shall maintain a 99.5% uptime with failover mechanisms.
- Regular database backups must be implemented to prevent data loss.
- Source: Client expectation for high-availability services.
- Priority: Level 0 (Essential)

III.4.5. Extensibility & Future Integrations:

- The system should be modular to allow future integrations (e.g., external GIS platforms).
- API endpoints should be designed for third-party compatibility.
- Source: Long-term system evolution considerations.
- Priority: Level 2 (Stretch Goal)

III.4.6. Improved Application Accessibility:

- Develop a standalone WinForms application that wraps the CEREO web application.
- Users can launch the app via a desktop shortcut instead of running npm start in a terminal.
- The application can be easily installed by downloading and extracting a ZIP file from GitHub, providing a more intuitive setup for non-technical users.
- Priority: Level 1 (Desirable)

II.5. Standards

For the **CEREO Living Atlas** application, several open and industry-recognized standards were identified as both relevant and feasible to implement. These standards promote data interoperability, improve application reliability, and ensure the platform aligns with modern web and data practices.

Implemented and Candidate Standards:

1. **GeoJSON (RFC 7946)** – GeoJSON defines a standard format for encoding geographic data structures using JSON. Implementing this standard allows spatial data such as

points, lines, and polygons to be represented consistently across the application. Integration into the existing FastAPI and PostgreSQL (PostGIS) architecture is straightforward, requiring only minor adjustments to data serialization and storage.
Benefit: Standardized geospatial representation and compatibility with GIS tools such as ArcGIS, Leaflet, and QGIS.

2. **OpenAPI Specification (OAS 3.0)** – The OpenAPI Specification provides a standardized way to document RESTful APIs. FastAPI natively supports OpenAPI, automatically generating documentation endpoints (`/docs` and `/openapi.json`) for all routes.
   Benefit: Enhances developer experience, enables automated testing, and provides consistent API documentation.
3. **Dublin Core Metadata Standard (ISO 15836)** – The Dublin Core standard defines a simple vocabulary for describing digital resources using metadata terms such as title, creator, subject, and date. Mapping existing database fields (e.g., title, description, username, category) to Dublin Core terms would improve data discoverability and compliance with open-data repositories.
   Benefit: Promotes interoperability and supports integration with research data catalogs and institutional repositories.
4. **JSON Schema (Draft 2020-12)** – JSON Schema provides a formal way to describe and validate the structure of JSON data. FastAPI and Pydantic already support this standard internally for request and response validation, ensuring all data exchanged through the API conforms to expected formats.
   Benefit: Increases reliability, automates data validation, and simplifies debugging and client integration.

By adopting these lightweight and compatible standards, the CEREO Living Atlas project improves maintainability, enhances data sharing, and aligns with established software engineering best practices.

III. Software Design - From Solution Approach

The functionality of the application's system should be focused on providing a user-friendly experience for all users that will be navigating through the Living Atlas, while also accommodating support for features for more technical users, such as adding complex geospatial data types to the database. To achieve an intuitive user interface, we implemented features for loading spatial data including raster layers and metadata from the endpoints of ArcGIS REST webserver of each U.S. state and generate interactive components such as clickable layers and pop-up info modal for the loaded data. Additionally, we will implement features that support uploading spatial data files from other geodatabases such as HydroShare. For users adding data to the database, the system should support upload of shapefiles and/or GeoJSON files to represent the geospatial data as shown on the map. The system will be designed such that these geospatial data files can be transferred from a user upload at the application frontend to the backend and into the database, which the frontend will read from and display its data on the map.

III.1. Architecture Design

III.1.1. Overview

The Living Atlas is designed with a three-layer architecture, comprising the frontend, backend, and database. The frontend, developed using React.js, delivers an interactive and user-friendly interface, enabling seamless map visualization and intuitive user interactions. On the backend, FastAPI acts as the core intermediary, efficiently managing incoming API requests, authentication, and data processing. Supporting this all is a PostgreSQL database, which securely and effectively stores geospatial data, user credentials, and metadata related to uploaded datasets. This modular approach ensures that each component can be updated or modified independently, enhancing the system's scalability, flexibility, and long-term maintainability.

A diagram of the basic subsystem layers of the architecture can be found in Appendix E.

III.1.2. Subsystem Decomposition

The system is structured into three core subsystems: the frontend, backend, and database. The frontend handles data visualization, processes user interactions, and provides tools for data submission. To enhance mapping capabilities, it leverages libraries such as Mapbox and Leaflet.js. The backend manages user authentication, enforces role-based access, validates data uploads, and delivers geospatial data through API endpoints. Meanwhile, the database efficiently organizes structured data, ensuring rapid retrieval and optimized querying through spatial indexing.

Frontend Subsystem

The frontend subsystem is responsible for managing user interactions and rendering geospatial data. It consists of key components such as the homepage, an interactive map, and data submission forms. By leveraging API calls, it seamlessly communicates with the backend to ensure smooth data processing and storage. A diagram of the frontend subsystem can be found in Appendix F.

Backend Subsystem

Serving as the system's processing core, the backend manages user authentication, enforces access control, and validates geospatial data submissions. Positioned as the bridge between the frontend and the database, it ensures secure and efficient data flow. To optimize performance, various algorithms are implemented to streamline API request handling and minimize response times. A diagram of the backend subsystem can be found in Appendix G.

Database Subsystem

The database subsystem is responsible for securely storing crucial system data, including user credentials, authentication records, and geospatial datasets. Built on PostgreSQL with PostGIS extensions, it is optimized for handling spatial queries with high efficiency. Its scalable architecture enables rapid data retrieval, advanced filtering, and indexing to ensure seamless access to stored information. A diagram of the database subsystem can be found in Appendix H.

III.2. Data design

The database design consists of structured tables to store geospatial data, user information, and role-based permissions. The Geospatial Data Table contains dataset entries, including spatial attributes, metadata, and associated files (e.g., shapefiles, GeoJSON). The User Table maintains account credentials, authentication tokens, and access levels. The Permissions Table defines role-based access to ensure only authorized users can upload or modify data. Data indexing and relational integrity are enforced through PostgreSQL with PostGIS, allowing for optimized querying and retrieval of large-scale environmental datasets.

There will be two major data structures present in the system database: The data structure containing all geospatial data points (including attached card information), and the data structure containing the account information of all users. Each object in the geospatial data structure will contain a shapefile so that the data can be displayed properly on the map in the correct location, and the card information for that data point, which can include the name of the user who added the data, the user's email, the organization who collected study data, a link to the study dataset, the title of the linked dataset, a description of the data set, the data category, and search tags. Each object in the account information data structure will contain a user ID, username, email, encrypted password, and the user's authorization level (view only, authorized to add data, or admin).

III.3. User Interface Design

The CEREO Living Atlas provides an interactive web interface for users to explore and manage environmental data. The homepage serves as the main point of interaction. Users can search, filter, and navigate data efficiently, while authorized users can contribute and manage geospatial information. The features described here align with those listed in the Use Cases section.

The homepage offers several key features. Users can search for data using filters in the top navigation bar. Geospatial data can be added through the "Add" button in the cards section. Viewing data is straightforward, with coordinates and shapes displayed on the left-side map and corresponding cards on the right. Users can also remove data from the same card section. Bookmarked data is accessible in this area as well. For diagrams illustrating these features, refer to Appendix I.

Editing data is available by clicking "Learn More" on a card, which opens a detailed window containing an edit option. Additionally, this window also allows authorized users to delete a card, update information and attached files as needed. For visual references of this feature, see Appendices J and K.

The user center includes account management functions. Password resets require entering the current password and a new one. This ensures security while keeping the process simple. A diagram of this feature can be found in Appendix L.

The homepage navbar provides access to key pages, including the registration page and administrator dashboard. The registration page allows non-logged-in users to fill out a form requesting an account. Submitted requests are sent to the admin panel, where an administrator can approve or deny them. For a diagram of this process, see Appendix M and N, respectively.

The administrator dashboard displays a list of all registered users in the Living Atlas. Admins can manage user accounts by modifying access levels or deleting accounts. This ensures proper role-based access control within the system. For a detailed view of the admin dashboard, refer to Appendix O.

III.4. Constraints

For the **CEREO Living Atlas** project, several practical constraints guide the development and deployment process:

1. **Technical and Resource Constraints** – The project is developed within the limits of its selected technology stack (FastAPI, React, PostgreSQL, and Google Cloud Storage) and relies on limited free-tier or academic resources. These restrictions influence hosting capacity, file size limits, and integration with third-party APIs.
2. **Time and Schedule Constraints** – Development follows a fixed academic timeline with defined sprint deliverables. Prioritization is required to ensure that essential features such as card creation, data uploads, and visualization tools are fully functional within the available timeframe.
3. **Security and Privacy Constraints** – The system must securely handle user data, credentials, and API keys. Measures such as password hashing, environment variable usage, and secure storage access are required to protect user information.
4. **Usability and Maintainability Constraints** – The interface must remain intuitive for diverse users while ensuring that code remains modular and maintainable for future feature expansion, bug fixes, and performance improvements.

By recognizing and addressing these constraints, the project maintains a balance between functionality, performance, and sustainability within the available technical and organizational limitations.

IV. Test Case Specifications and Results

IV.1. Testing Overview

To validate the functionality and robustness of the CEREO Living Atlas alpha prototype, a combination of unit, integration, and manual tests were conducted across both frontend and backend systems. Testing focused on verifying core features such as card creation, thumbnail uploads, user role access, and data filtering. Continuous integration checks ensured the backend builds successfully on every commit. Manual testing also confirmed proper rendering and responsiveness across modern browsers. Together, these tests provide confidence in the stability of the current prototype while highlighting key areas for further refinement in the next development phase.

V.1.1 Unit Testing

Our team will follow all standard unit testing procedures. Most unit testing will focus on code developed by previous teams, as new development is mostly built using these previously developed units. The previously developed units to be tested include loading a card from the database, adding or deleting a card from the database, adding or deleting an account, submitting a form, and loading the map from Mapbox. It is crucial that these previously developed units are tested as the database host has been switched to Microsoft Azure and these units must still function with the new database for the overall system to function correctly. Some newly developed units to be tested include changing an account’s password, sending a reset password email, and loading files and images from the database.

V.1.2. Integration Testing

Given the multi-layer design of our application, consisting of a FastAPI backend, React Front end and integration with google cloud storage- the integration testing focuses on validating the flow of data between these layers. Unlike a monolithic structure, this modular setup introduces complexities in testing complete workflows and can cause issues with the flow of data between the layers.

The development team will primarily rely on manual testing within local and staging environments to replicate integrated scenarios like form submissions involving file and thumbnail uploads, database entry creation, and frontend content display. At this stage, automated integration testing remains limited due to the complexities of mocking Google Cloud APIs and maintaining database state across different layers. However, specific endpoints may still be tested using tools such as Postman or pytest with FastAPI’s TestClient.

Although the ultimate goal is comprehensive end-to-end integration testing across all components, certain elements—like frontend thumbnail display logic—may be tested in isolation when backend deployment or effective mocking proves challenging.

V.1.3. System Testing

V.1.3.1. Functional testing:

Functional testing is carried out manually by developers, following the project’s Requirements and Specifications document. Each functional requirement is matched with a specific test case. This includes:

- Creating a card with all metadata fields
- Uploading data files and thumbnail images
- Retrieving and displaying card data, including image previews
- Downloading uploaded files
- Deleting cards and confirming cascade deletions (files and thumbnails)

Tests are validated through the browser interface and network tools like Chrome DevTools or Postman. Any failed tests are documented with clear reproduction steps and assigned back to the original developer for resolution. As the application evolves, the testing plan will be updated accordingly.

V.1.3.2. Performance testing:

Performance testing targets two main areas: backend response time and frontend rendering speed.

Backend testing includes:

- Measuring FastAPI response times with large payloads (e.g., multi-megabyte uploads)
- Tracking latency when accessing files and thumbnails stored on Google Cloud

Frontend testing includes:

- Monitoring load times when rendering pages with numerous cards and images
- Evaluating performance on lower-end devices or slower networks

Manual observations are supported with browser profiling tools like Lighthouse and Chrome DevTools. While not benchmarked against strict numerical thresholds, performance is assessed qualitatively based on overall responsiveness and user experience.

V.1.3.3. User Acceptance Testing:

User acceptance testing will be conducted exclusively by the client. Instead of structured forms, feedback will be gathered through direct meetings.

A. Process

- The client will test key workflows, including creating, viewing, editing, and deleting cards.
- Particular focus will be placed on new features, such as image upload and thumbnail display.
- Observations and feedback will be shared in scheduled meetings.

B. Revision Process

- Developers will document and categorize feedback as bugs, feature requests, or general improvements.
- Actionable items will be tracked in the issue management system (e.g., GitLab Issues).
- Any necessary changes will be implemented and deployed to staging before final release.

The final iteration will be considered complete once the client confirms that all key features function as expected and the user experience meets their needs.

IV.2. Environment Requirements

For the frontend, we are planning to use React, which allows us to utilize tools like Jest and Selenium for testing. These tools facilitate easy testing of our UI across different browsers and JavaScript components. Lighthouse can be used to measure loading times and ensure performance standards are met. For staging environments, we are considering using Docker which will help create containerized versions of the application that can be easily deployed and tested across various machines. For the database, tools like Alembic or Flyway will manage schema changes and seed data during testing. We may switch to other tools as the project evolves, and the tools currently selected are subject to change. Version control is handled through GitHub. There are no specific hardware requirements for the testing environment.

IV.3. Test Results

At this stage, we have not yet begun formal testing of the prototype. No structured test case results are available beyond informal integration testing performed manually with DevTools to verify basic functionality. Formal unit and system testing will be introduced in future sprints, and results will be documented once testing begins.

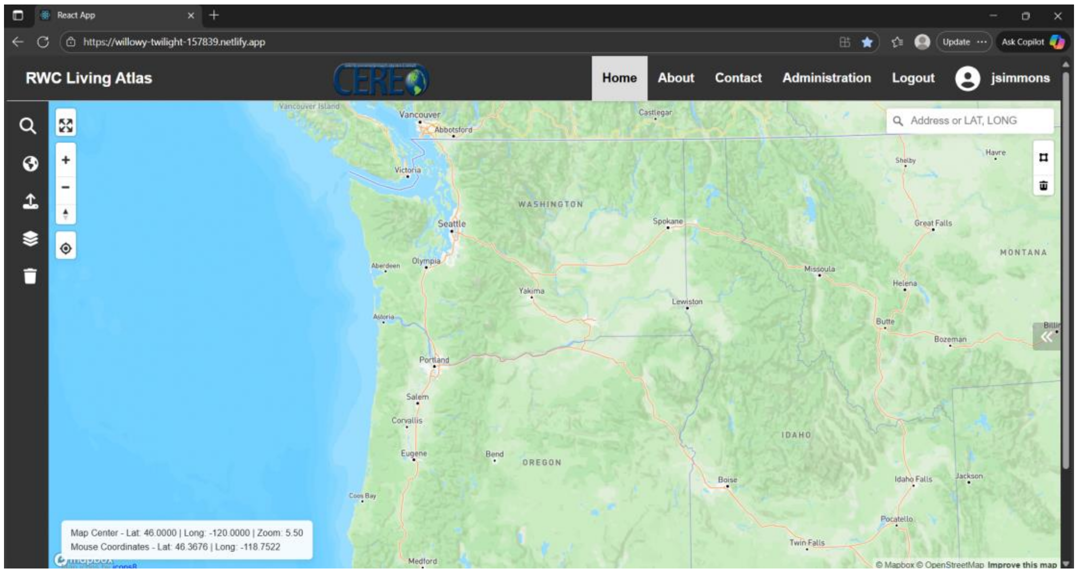
V. Projects and Tools used, Libraries, and Frameworks

Tool/Library/Framework	Quick note on what it was for
React.js	Built the frontend user interface and handled dynamic components.
Bootstrap / CSS	Provided styling, layout, and responsive design for the web app.
FastAPI	Framework used to build the backend API and connect to the database.
Microsoft Azure Database	Database for storing card data, user info, and metadata.
Google Cloud Storage	Hosted uploaded files and thumbnails for cards.
Render	Deployed and hosted the FastAPI backend.
Netlify	Deployed and hosted the React frontend.
Mapbox GL JS	Displayed interactive maps and supported drawing/viewing markers.
Git/GitHub	Version control and team collaboration.
Visual Studio Code	Main IDE for frontend and backend development.

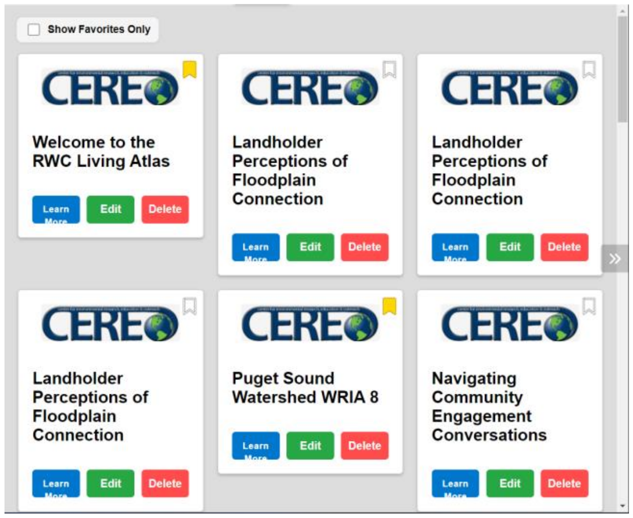
Language	Usage
JavaScript	Frontend (React components, API calls, dynamic UI).
Python	Backend (FastAPI routes, database queries, file handling).
SQL	Queries for PostgreSQL database operations.
HTML5	Page structure in frontend components.
CSS	Styling and responsive design for web app.

VI. Description of Final Prototype

The prototype of our web application is designed to centralize environmental and spatial information in one platform. The homepage features a top navigation bar that directs users to other pages, along with sidebars on the left and right that provide key functionalities (see below).


React App
https://willowy-twilight-157839.netlify.app
RWC Living Atlas
Home
About
Contact
Administration
Logout
jsimmons
Address or LAT, LONG
Map Center - Lat: 46.0000 | Long: -120.0000 | Zoom: 5.50
Mouse Coordinates - Lat: 46.3676 | Long: -118.7522

The user can click on the two-arrows button located on the right side of the screen to open the card container and view all the cards. Users can also drag the left border of the card container to the left to adjust its width (See Appendix R). To see the location of a card on the map, the user can click on the empty space of the card, and the map will center on that card's location.

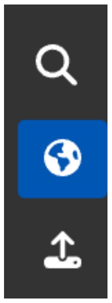

Show Favorites Only
Welcome to the RWC Living Atlas
Landholder Perceptions of Floodplain Connection
Landholder Perceptions of Floodplain Connection
Landholder Perceptions of Floodplain Connection
Puget Sound Watershed WRIA 8
Navigating Community Engagement Conversations
Learn More
Edit
Delete

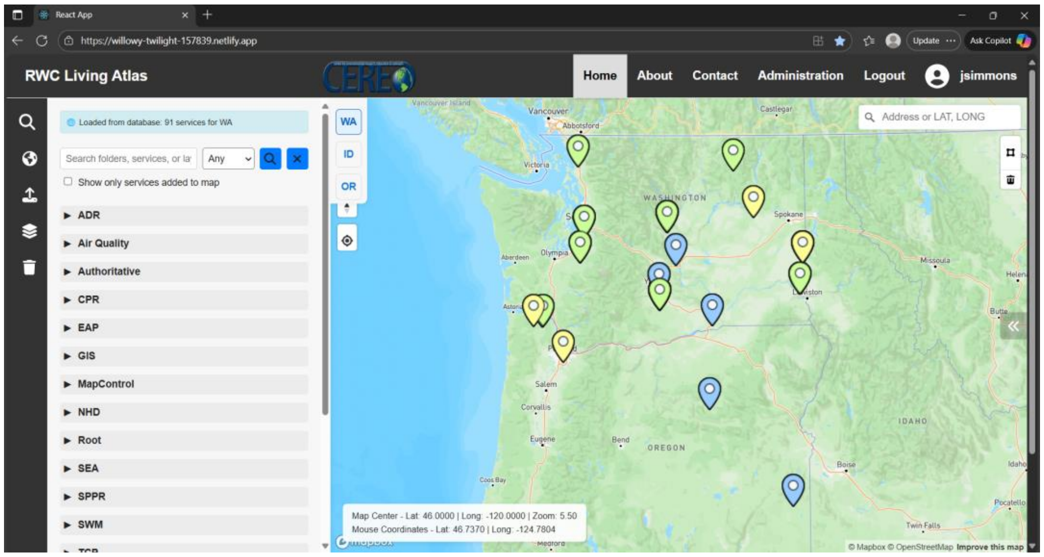
Clicking the "Learn More" button on a card opens an info modal with details about that card (see Appendix T). Authorized admin users can edit or delete a card using the "Edit" and "Delete" buttons respectively (see Appendix U and Appendix V).

To add a new card, click the Upload Card button on the left sidebar. This opens a modal where you can scroll through and fill in all the required information. When finished, click "Submit" to add the card (see Appendix W).

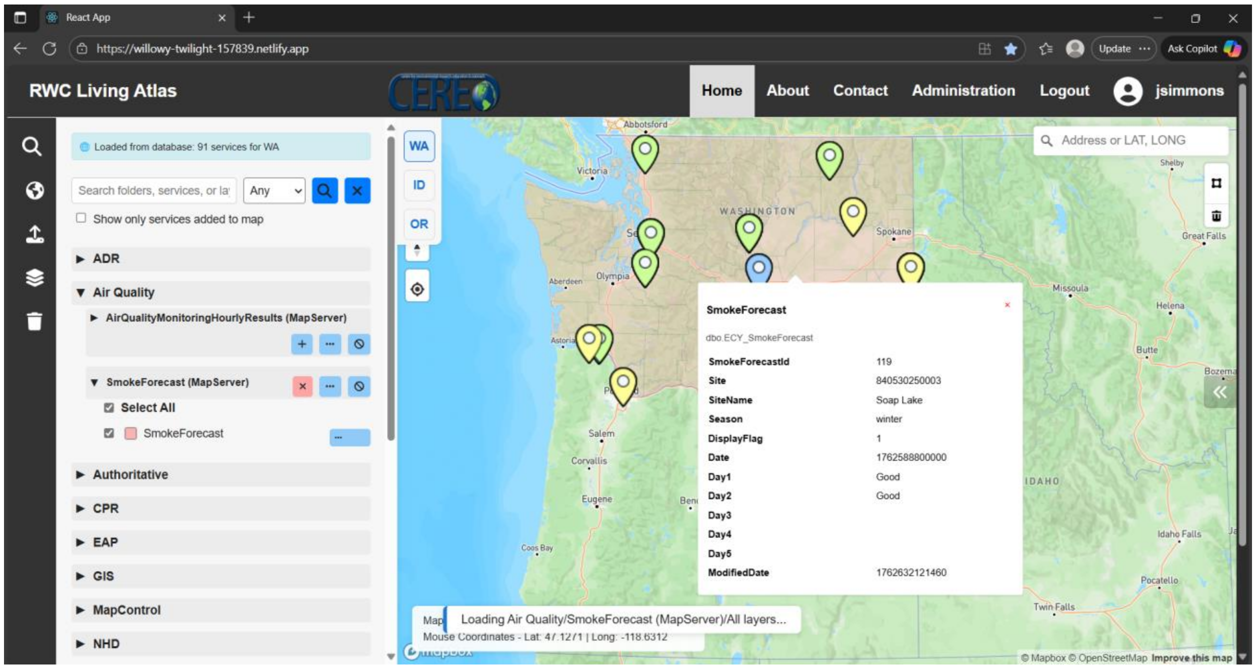
Users can also add or remove a card from their favorites by clicking the bookmark icon in the top-right corner. To view only bookmarked cards, check the “Show Favorites Only” box (see the image above).

To display spatial data from ArcGIS on the map, click the Earth icon on the left sidebar to open the panel. The panel shows data in a folder structure: items like “AQ” are folders, “SmokeForecast (MapServer)” is a service, and each item under a service with a legend is a layer. Users can click “Load” or “Select All” to add or remove all layers under a service or check and uncheck individual boxes to display specific layers on the map.




RWC Living Atlas
Home
About
Contact
Administration
Logout
jsimmons
Loaded from database: 91 services for WA
Search folders, services, or la
Any
Show only services added to map
ADR
Air Quality
Authoritative
CPR
EAP
GIS
MapControl
NHD
Root
SEA
SPPR
SWM
WA
ID
OR
Address or LAT, LONG
Map Center - Lat: 46.0000 | Long: -120.0000 | Zoom: 5.50
Mouse Coordinates - Lat: 46.7370 | Long: -124.7804

Once the data finishes loading, the layers appear on the map as colored regions, lines, points, or other shapes. Users can click on these elements to open a pop-up modal displaying details of the selected geospatial layer.

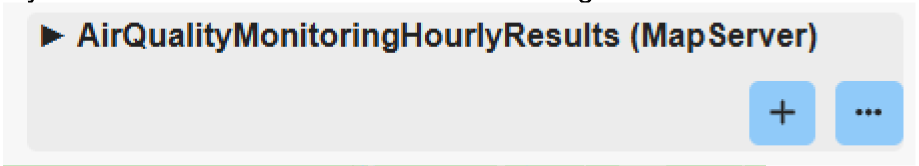

React App
https://willowy-twilight-157839.netlify.app
RWC Living Atlas
CEREO
Home
About
Contact
Administration
Logout
jsimmons
Loaded from database: 91 services for WA
Search folders, services, or la
Any
Show only services added to map
ADR
Air Quality
AirQualityMonitoringHourlyResults (MapServer)
SmokeForecast (MapServer)
Select All
SmokeForecast
Authoritative
CPR
EAP
GIS
MapControl
NHD
WA
ID
OR
Address or LAT, LONG
Loading Air Quality/SmokeForecast (MapServer)/All layers...
Mouse Coordinates - Lat: 47.1271 | Long: -118.6312
© Mapbox © OpenStreetMap Improve this map

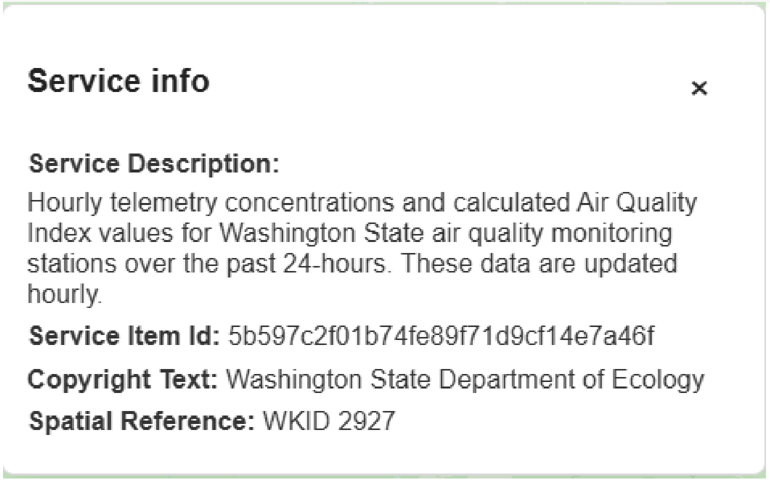
SmokeForecast

dbo.ECY_SmokeForecast

SmokeForecastId	119
Site	840530250003
SiteName	Soap Lake
Season	winter
DisplayFlag	1
Date	1762588800000
Day1	Good
Day2	Good
Day3	
Day4	
Day5	
ModifiedDate	1762632121460

To view more details about a service or layer, users can click on the learn-more button, which features a "..." icon. It will then open a pop-up modal that contains the details of the service or layer. The modal also contains a link showing the source of data for the service.




Service info ×

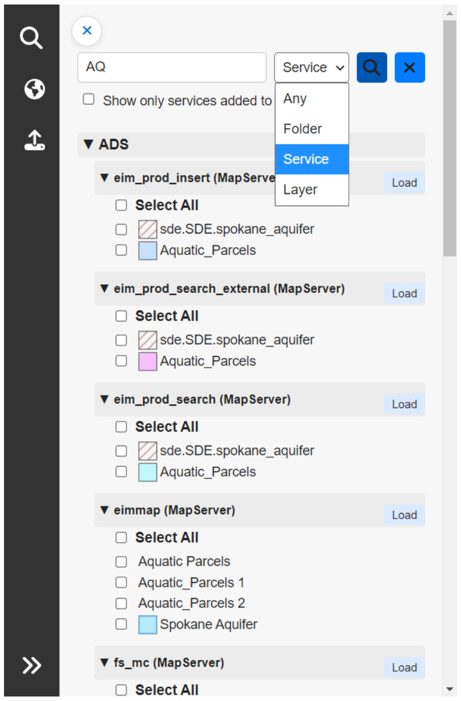
Service Description:
Hourly telemetry concentrations and calculated Air Quality Index values for Washington State air quality monitoring stations over the past 24-hours. These data are updated hourly.

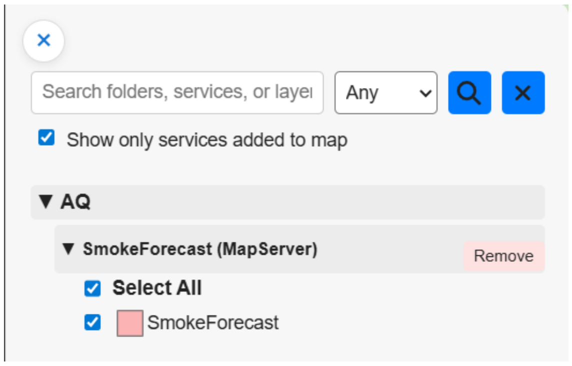
Service Item Id: 5b597c2f01b74fe89f71d9cf14e7a46f

Copyright Text: Washington State Department of Ecology

Spatial Reference: WKID 2927

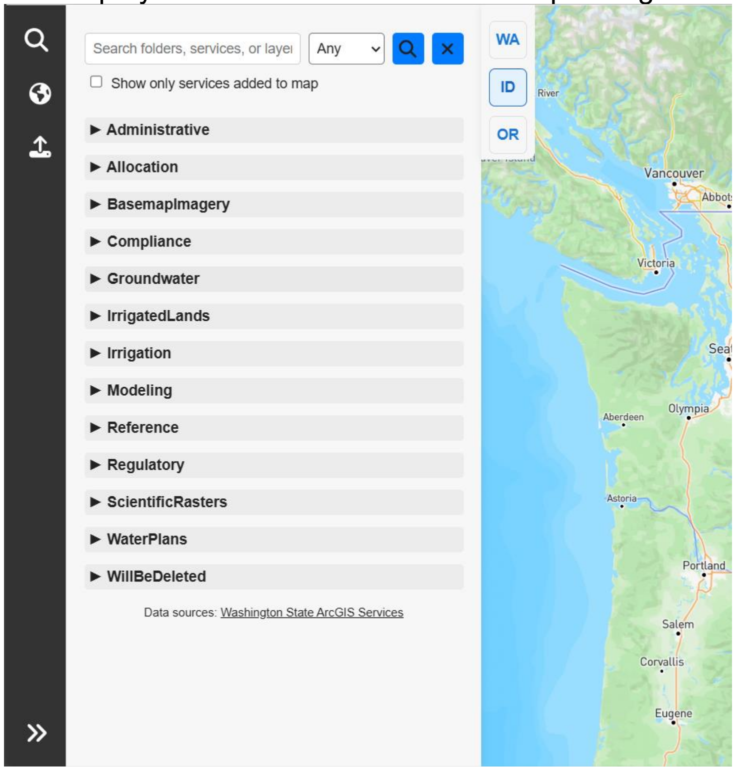
Users can search for a specific folder, service, or layer by entering keywords into the search bar and clicking on the search button. The panel will then display all matching results. To narrow the search scope, users may also select an option (folder, service, or layer) from the drop-down menu. To clear the search, click the "x" icon next to the search bar to reset the panel to its default view.

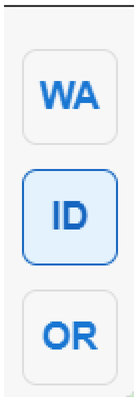



Search folders, services, or layer
Any
Show only services added to map
AQ
SmokeForecast (MapServer)
Remove
Select All
SmokeForecast

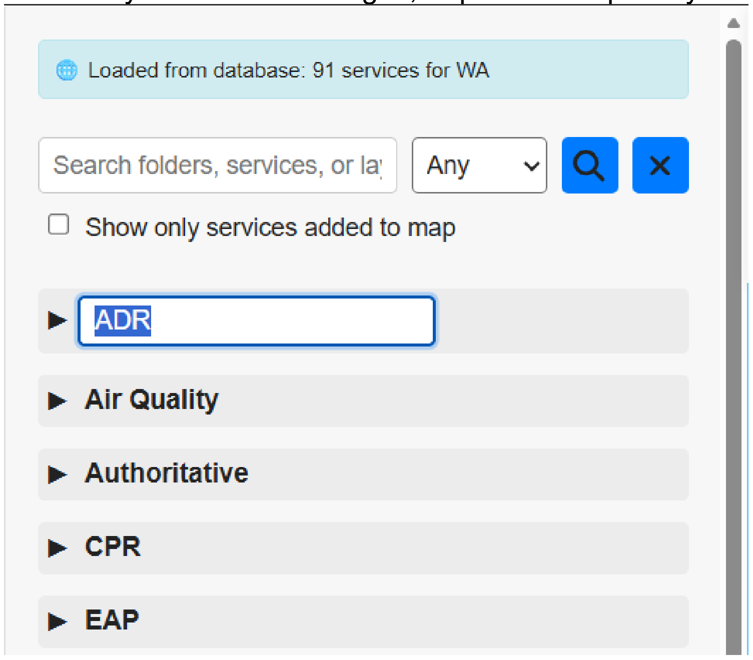
Additionally, checking the “Show only services added to map” box will display only the services that have already been loaded onto the map.

Currently, the Living Atlas contains ArcGIS data for Washington, Idaho, and Oregon. Users can click the state buttons in the menu attached to the upper-right edge of the upload panel to toggle and display the ArcGIS data for the corresponding state in the panel.

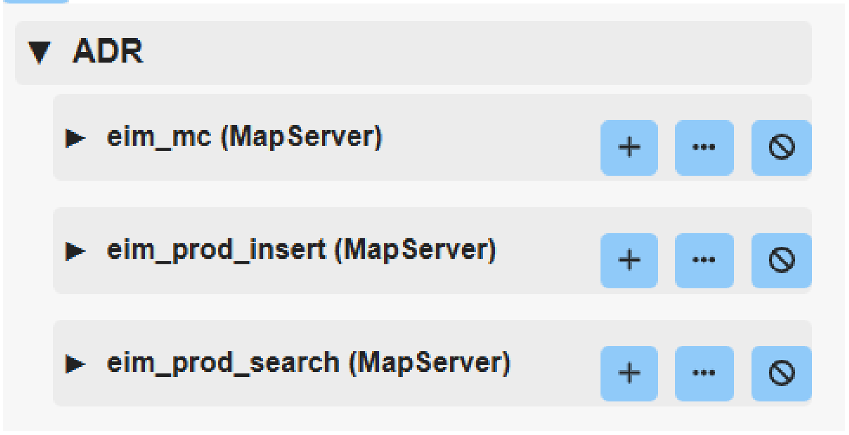




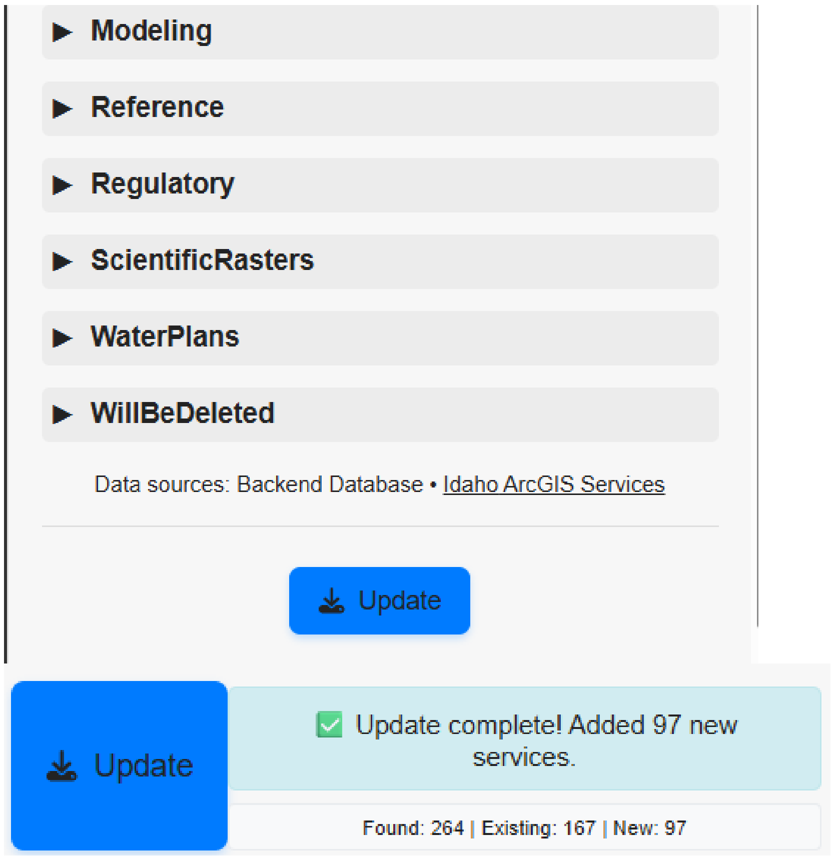
To rename a folder or service, double click on the name area to choose rename action. Press 'enter' key to save the changes, or press 'escape' key to discard the changes.


Loaded from database: 91 services for WA
Search folders, services, or la
Any
Show only services added to map
ADR
Air Quality
Authoritative
CPR
EAP

To remove a service from the upload panel, user can simply click on the remove button and confirm the deletion in the pop-up window.



Users can synchronize all items in the Upload panel with the ArcGIS server by clicking the Update button at the bottom of the panel. This ensures the service items in the Upload panel are up to date. When the update completes, the system displays statistics such as how many items were added.



Data sources: Backend Database • Idaho ArcGIS Services

✅ Update complete! Added 97 new services.

Found: 264 | Existing: 167 | New: 97

For testing, since formal unit, integration and system testing has not yet been implemented, all tests were conducted manually using DevTools and backend logs (see Appendix S).

VII. Social Responsibility and Broader Impacts

CEREO (Center For Environmental Research, Education, and Outreach) is dedicated to addressing environmental challenges through research and educational support, aiming to develop meaningful solutions for critical environmental issues. The Living Atlas will help CEREO in their mission by providing an easy-to-use interface to view important data sets for our local waterways and other environmental features. Researchers will be able to use the Living Atlas to enhance collaboration with other researchers, as the user-contributed data of other researchers will be easily viewable. Tribal communities will be able to use the Living Atlas to monitor the local ecosystem, identify environmental concerns, and address critical issues. Government agencies will also benefit from the Living Atlas because making these data sets easily accessible to them will give them greater confidence to invest in environmental solutions with CEREO as there is evidence that these funds are being utilized effectively. By fostering collaboration and transparency for environmental data, the Living Atlas will support cooperative and informed decision-making. While developing the Living Atlas, our team always kept in mind that this application must be accessible and useful to our key stakeholders, that being environmental researchers, tribal communities, and government agencies, as we believe that is our responsibility as software engineers. By making the user interface easy-to-use while providing a large amount of accessible information for our stakeholders, we achieve our goal of putting our work towards improving the world we live in.

VIII. Product Delivery Status

The Living Atlas has already been delivered to our client, and they have been actively using the product. While the initial version of the system has been functional, our team has continued to make numerous frontend improvements, feature additions and stability updates based on feedback. These changes are ongoing but are expected to be wrapped up and fully polished within the next month to month and a half.

The project was demonstrated to the clients during our scheduled check-ins, and handoff will be made directly to the CEREO research team for continued use.

Project Location Information

1. **Source Repositories**
    a. Frontend (React, deployed via Netlify) https://github.com/WSUCptSCapstone-S25-F25/-cereo-fullstackapp-
    b. Backend (FastAPI, deployed via Render): https://github.com/WSUCptSCapstone-S25-F25/-cereo-fullstackapp-/tree/main/LivingAtlas1-main/backend

2. **Equipment Storage Location**
    a. No physical equipment was required or left with the client. All project components are hosted online through Netlify, Render, Google Cloud Storage, and Microsoft Azure.

3. **Materials Needed to Rebuild the Project**
    a. Source code for both frontend and backend (available in GitHub repositories).
    b. Google Cloud service account credentials for file and thumbnail uploads.
    c. PostgreSQL database schema (stored in backend documentation and repository).
    d. Deployment instructions for Netlify (frontend) and Render (backend).

Installation and Setup Instructions
Detailed instructions for installing and running the project locally, as well as redeploying to

cloud services, are included in the project's README files within the repositories. This documentation covers environment setup, database connection, credentials configuration, and deployment steps so that future users can rebuild the system from the ground up if necessary.

IX. Conclusions and Future Work

IX.1. Limitations and Recommendations

Performance can be noticeably limited at times, for example, loading times for map pins and added data layers may be long, and they will not appear on the map immediately. This may affect the application's goal of being easy-to-use and user-friendly. These performance issues should be a priority to address because as the amount of data in the Living Atlas increases, these loading times may get worse. The best course of action would be to begin comprehensive performance testing to find the potential cause of the issues and make changes accordingly.

Every user should be able to attach a file to a card that they upload. The current practice for Living Atlas users wanting to attach a data set to their card is to include an external link that will lead to the data set. However, if users could attach the file itself to the card, then other users would be able to download the data set directly from the Living Atlas. This would make data easily accessible for all users.

App security and data integrity are limited in the current system. The web server and database lack defenses against common attacks such as SQL injection or denial-of-service (DoS). The reset password feature, which relies on SMTP, could also be vulnerable if not configured securely. Improvements such as stronger encryption, TLS for email delivery, rate limiting, and two-factor authentication (2FA) would help protect user data and build confidence as the application grows.

Currently, the Living Atlas relies on fetching toggleable layers from the ArcGIS REST server. While this provides live updates, the approach has limits. Raster layers are made interactive by overlaying transparent vector layers, which works for sparse data but struggles when features are dense or overlapping. Rendering raster layers as colored regions can also be slow, and users may wait before data appears. To improve this, future work could explore caching frequently used raster data or integrating ArcGIS feature services for more robust handling. Additionally, users should be able to upload their own GIS files so they can share data as toggleable layers, rather than relying only on developer-added pins or ArcGIS services.

The current fetching logic also has challenges. Service URLs are saved locally, which risks becoming outdated if ArcGIS changes or adds new services. A more dynamic update process is needed so the system can periodically re-fetch and refresh the catalog. Beyond REST services, other ArcGIS data types such as feature servers could be incorporated to expand interactivity and make spatial data more versatile.

IX.2. Future Work

Future developers of the Living Atlas could consider expanding the accessibility of this project by making it compatible with more platforms. The website functions on mobile devices but it could be more accessible to mobile users by resizing UI elements on smaller screens or developing a dedicated mobile application. Future teams could also develop dedicated Living Atlas software so that the application can be accessed outside of a browser. Giving users more options in how the Living Atlas can be accessed will expand its user base which leads to more data and collaboration.

Beyond accessibility, future work should consider commercial or institutional adoption. With additional investment, the Living Atlas could be hosted on dedicated infrastructure with higher reliability, stronger security, and custom domain integration. Commercialization opportunities could include subscription-based hosting for research institutions, integration with state or tribal monitoring systems, or partnerships with environmental agencies.

X. Acknowledgements

We would like to thank the 2023 development team, including Joshua Long, Mitchell William Kolb, Sierra Amelia Svetlik, Flavio Alvarez Penate, Wyatt Croucher, and Phearak Both Bunna, for laying the groundwork of the Living Atlas project. Their contributions provided the base on which we were able to continue building and improving the system. We also thank the 2024 development team members, Bryce Moser and Silas Peterson, for their work in refining the platform and adding new features.

We are grateful to our sponsors, Dr. Jan Boll and Dr. Julie Padowski from the Center for Environmental Research, Education, and Outreach (CEREO), for their valuable feedback and guidance. Their input helped shape the direction of the project and identify areas for improvement.

Finally, we thank our mentor, Ananth Jillepalli, for keeping the team on track, providing clear instructions, and guiding us through the development process. His direction was key in helping us move the project forward.

XI. Glossary

API (Application Programming Interface) – A set of rules and protocols that allow different software applications to communicate with each other.

Backend – The server-side logic of an application responsible for processing requests, handling data storage, and managing business logic.

Columbia River Basin – A geographic region covering the watershed of the Columbia River, which is the primary focus of the Living Atlas for water quality data.

Database – A structured collection of data stored electronically, which in this case includes geospatial data, user credentials, and metadata.

FastAPI – A modern web framework for building APIs with Python, known for its speed and ease of use.

Frontend – The part of the application that users interact with directly, typically consisting of a graphical user interface (GUI).

Geospatial Data – Information that includes geographic location attributes, often represented using coordinates and spatial features.

GeoJSON – A format for encoding geographical data structures using JSON.

GIS (Geographic Information System) – A system designed to capture, store, manipulate, analyze, and display spatial or geographic data.

Leaflet.js – An open-source JavaScript library used for interactive maps.

Living Atlas – A geospatial web application designed for visualizing and sharing environmental data, focusing on water quality in the Columbia River Basin.

Mapbox – A mapping platform providing tools for designing and implementing custom maps.

Metadata – Data that provides information about other data, such as descriptions, timestamps, and ownership details.

PostGIS – A spatial database extender for PostgreSQL that enables advanced geospatial queries.

PostgreSQL – An open-source relational database system used for managing structured data.

Role-Based Access Control (RBAC) – A security mechanism that restricts access to system functions based on user roles.

Scalability – The ability of a system to handle growth in users, data, or computational workload without performance degradation.

Shapefile – A common geospatial vector data format used for geographic information system (GIS) applications.

Spatial Indexing – A technique used in databases to optimize spatial queries, improving the efficiency of geographic data retrieval.

User Authentication – The process of verifying the identity of users before granting access to the system.

REST - A style for building web services using standard HTTP methods like GET, POST, PUT, and DELETE.

ArcGIS - A software platform by Esri for mapping, analyzing, and sharing geospatial data.

Raster Data - A grid-based format for storing spatial data, often used for images like satellite photos or elevation models.

Metadata - Descriptive information about data, such as source, format, time, and author.

Web Server - A system that delivers web pages or services to clients over the internet using HTTP/HTTPS.

Endpoints - Specific URLs in an API where requests can be sent to perform actions or retrieve data.

XII. References

Bruegge, Bernd., Dutoit, Allen H.. Object-oriented Software Engineering: Using UML, Patterns, and Java. United Kingdom: Prentice Hall, 2010.

Lethbridge, Timothy Christian., Laganière, Robert. Object-oriented Software Engineering: Practical Software Development Using UML and Java. United Kingdom: McGraw-Hill Education, 2005.

Li, Eldon Y. "Software testing in a system development process: A life cycle perspective." *Journal of Systems Management* 41, no. 8 (1990): 23-31.

XIII. Appendix A – Team Information

Zachary Garoutte (left), Jonathan Simmons (middle), Yaru Gao (right)

XIV. Appendix B - Example Testing Strategy Reporting

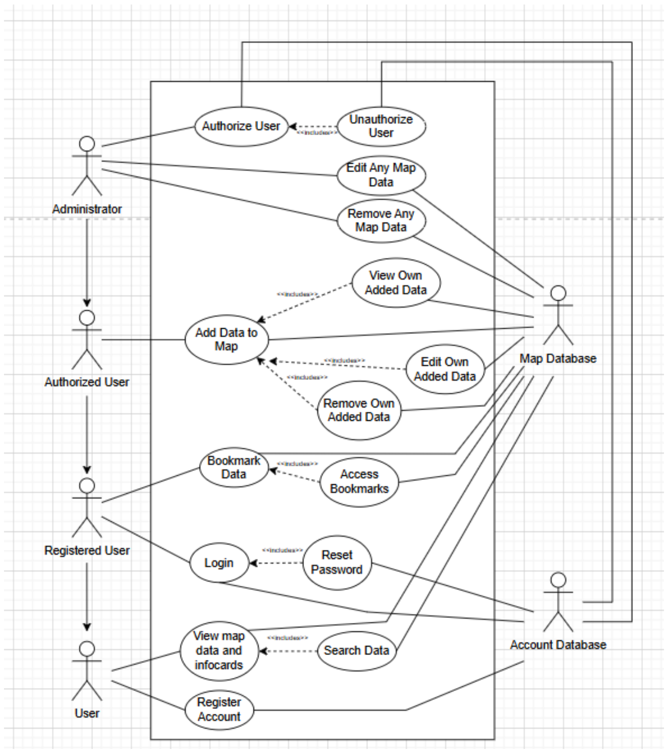
Our team has not yet performed formal unit testing or gathered user feedback other than from the client but only manual testing using browser DevTools. However, we plan to introduce unit testing in a future sprint, which we believe in crucial as our database has swapped hosts and we need to ensure all features still function. For user acceptance testing, feedback will be gathered through direct meetings with the client and testing will be considered complete once the client confirms that all key features function as expected and the user experience meets their needs. For more information about our testing plans, see Section V.

XV. Appendix C - Project Management

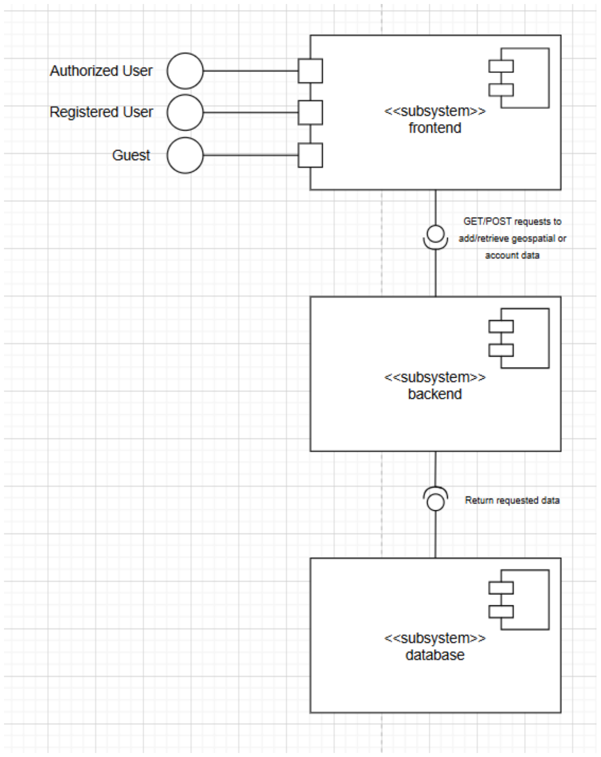
The recurring schedule for our team consisted of both a weekly client meeting with Julie Padowski and Jan Boll from CEREO and a weekly meeting with our mentor Ananth Jillepalli. Both of these meetings are conducted remotely via Zoom. The client meetings mainly consist of demonstrations of recently develop features and bug fixes by our team where we can receive feedback on these changes. These meetings have been beneficial to the development process as consistent feedback helps us better understand client expectations so we can update our process to meet them. The mentor meetings mainly consist of our mentor ensuring that all team members are contributing to the project and that our time spent in development meets expectations. The mentor meetings have also been beneficial to our team as our mentor as they keep us on schedule and can give us reminders for upcoming key deadlines. We do not have a scheduled meeting with only team members but there has been frequent communication via Discord throughout active development. Our team makes use of our GitHub repository's issues and projects board to organize all current features and bug fixes in development and assign each one to a team member. Using these tools have been very useful for our team as they provide a clear guide for what should be worked on next. Screenshots of the issues and projects board we have used are shown in Appendix R. We also made frequent use of email with both the client and our mentor for questions outside of our scheduled meeting times.

XVI. Other Appendices

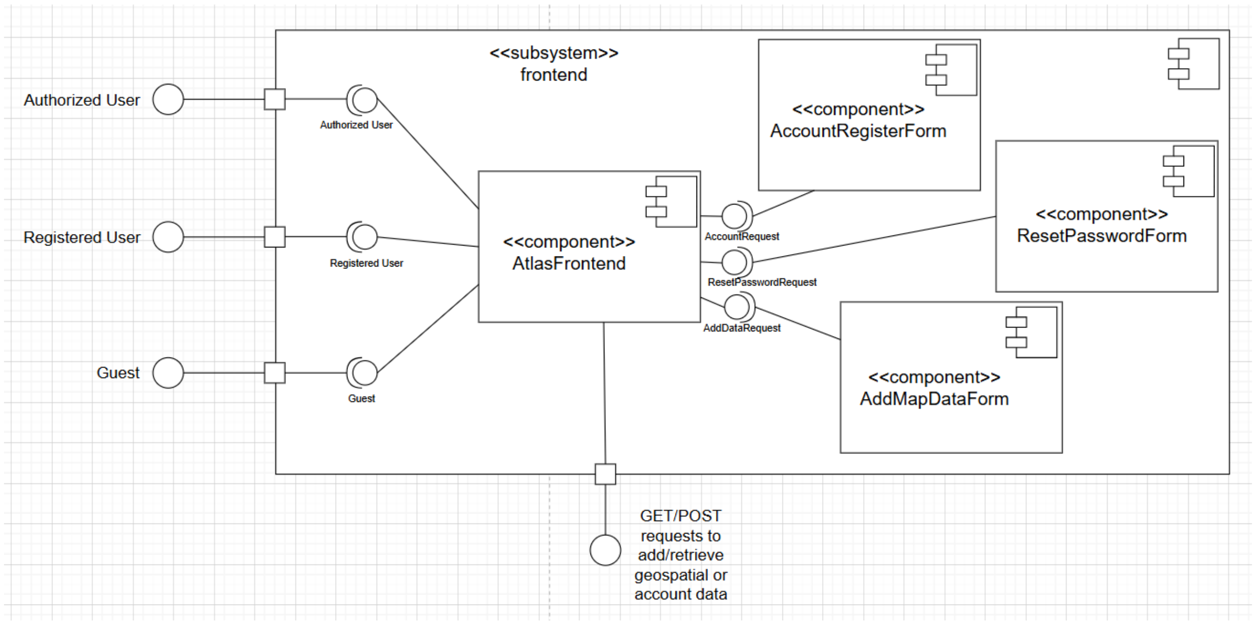
Appendix D: Use Case Diagram



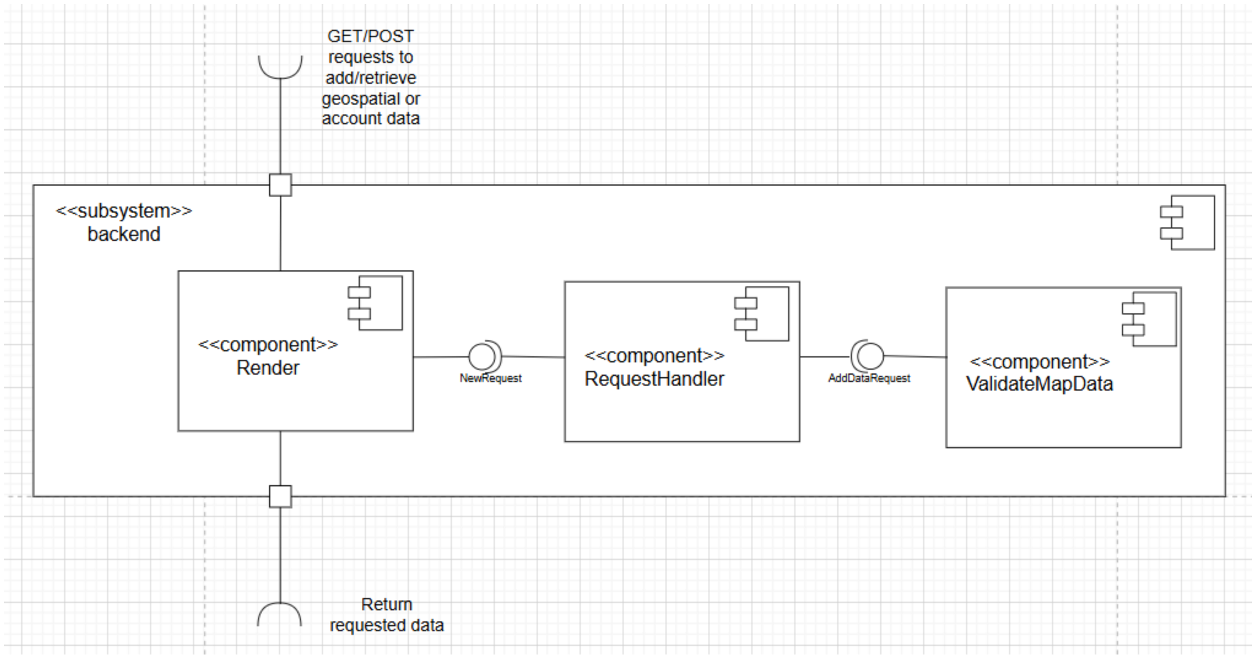
Appendix E: Architecture Design Overview Diagram



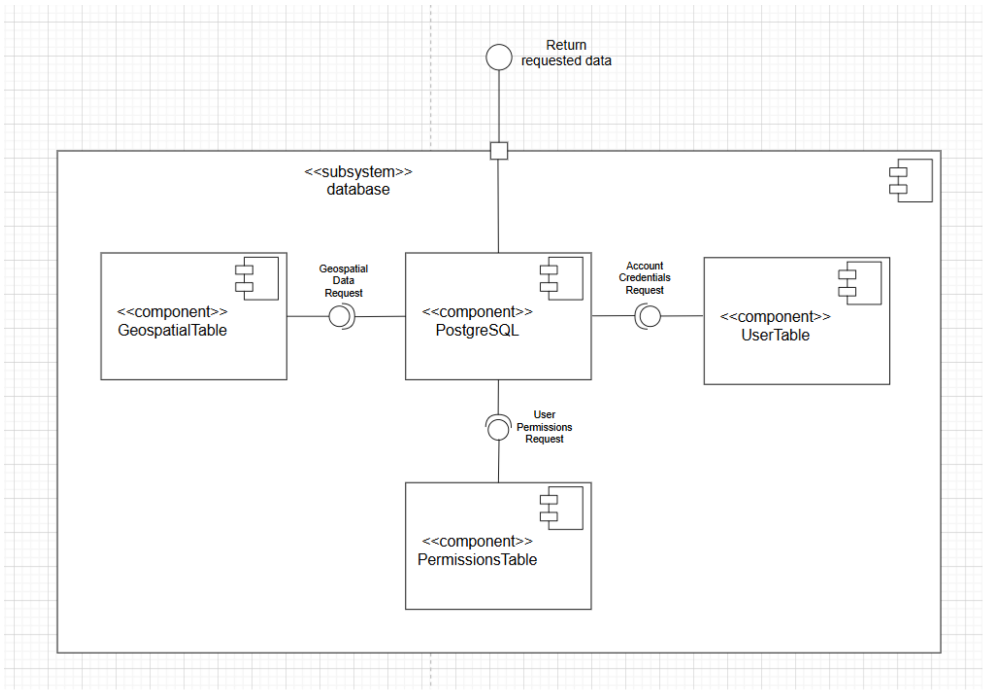
Appendix F: Frontend Subsystem Diagram



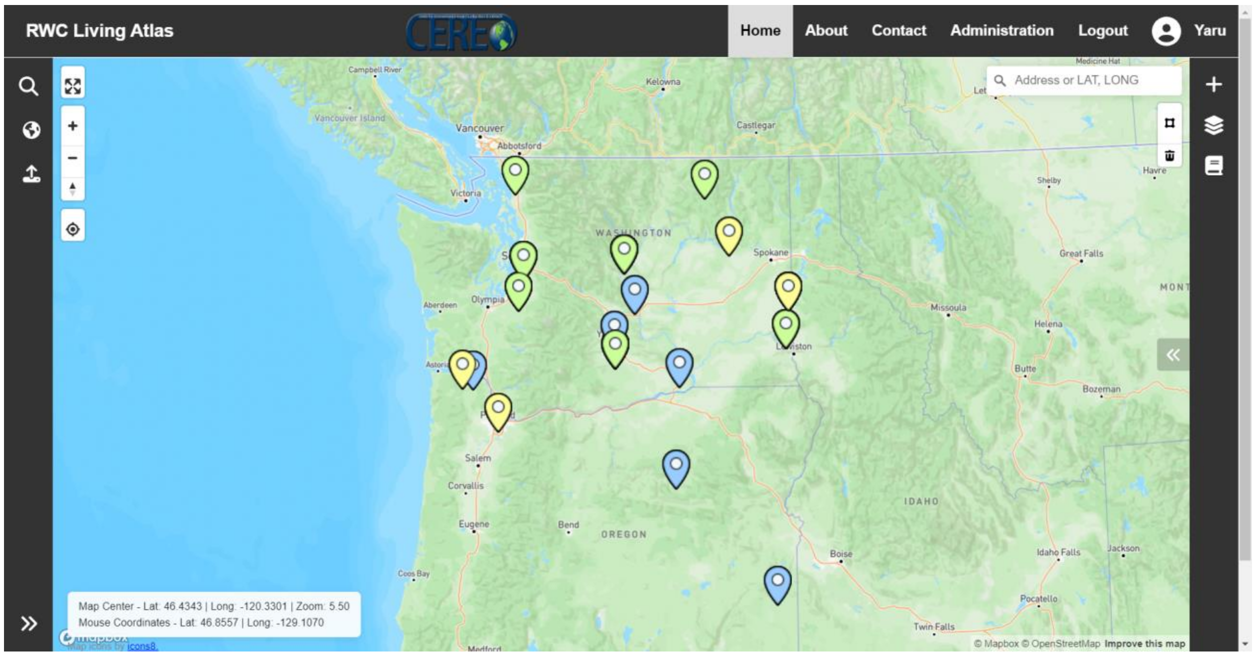
Appendix G: Backend Subsystem Diagram



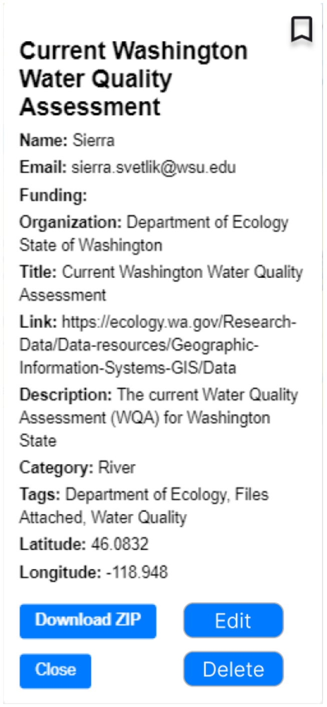
Appendix H: Database Subsystem Diagram



Appendix I: Mock-up diagram of the homepage, illustrating search, filtering, data addition, viewing, removal, and bookmark access.


RWC Living Atlas
CEREO
Home
About
Contact
Administration
Logout
Yaru
Address or LAT, LONG
WASHINGTON
OREGON
IDAHO
Map Center - Lat: 46.4343 | Long: -120.3301 | Zoom: 5.50
Mouse Coordinates - Lat: 46.8557 | Long: -129.1070
© Mapbox © OpenStreetMap Improve this map

Appendix J: Mockup diagram of the "Learn More" window, showing the edit option for modifying geospatial data.



Current Washington Water Quality Assessment

Name: Sierra

Email: sierra.svetlik@wsu.edu

Funding:

Organization: Department of Ecology State of Washington

Title: Current Washington Water Quality Assessment

Link: https://ecology.wa.gov/Research-Data/Data-resources/Geographic-Information-Systems-GIS/Data

Description: The current Water Quality Assessment (WQA) for Washington State

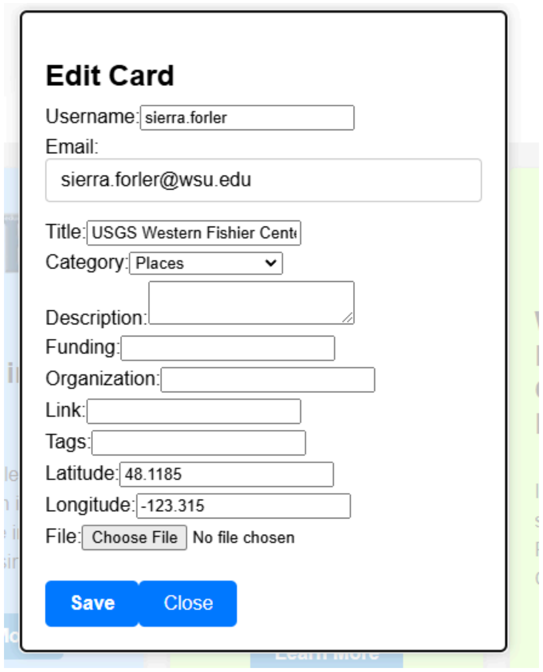
Category: River

Tags: Department of Ecology, Files Attached, Water Quality

Latitude: 46.0832

Longitude: -118.948

Appendix K: Mockup diagram of the edit functionality within the "Learn More" window.



Edit Card

Username: sierra.forler

Email: sierra.forler@wsu.edu

Title: USGS Western Fishier Cent

Category: Places

Description:

Funding:

Organization:

Link:

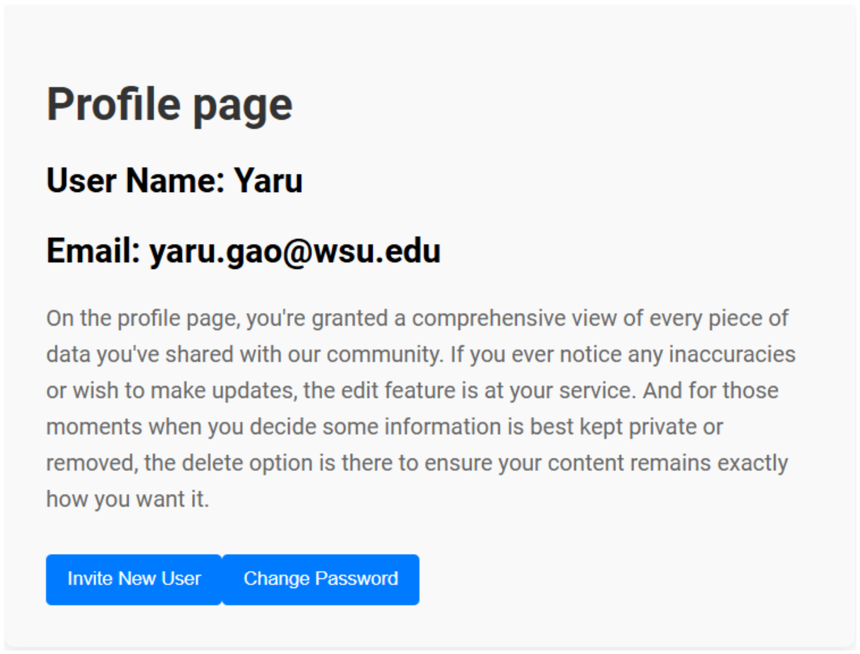
Tags:

Latitude: 48.1185

Longitude: -123.315

File: Choose File No file chosen

Appendix L.1 and L.2: Mockup diagram of the user center, depicting the password reset process.

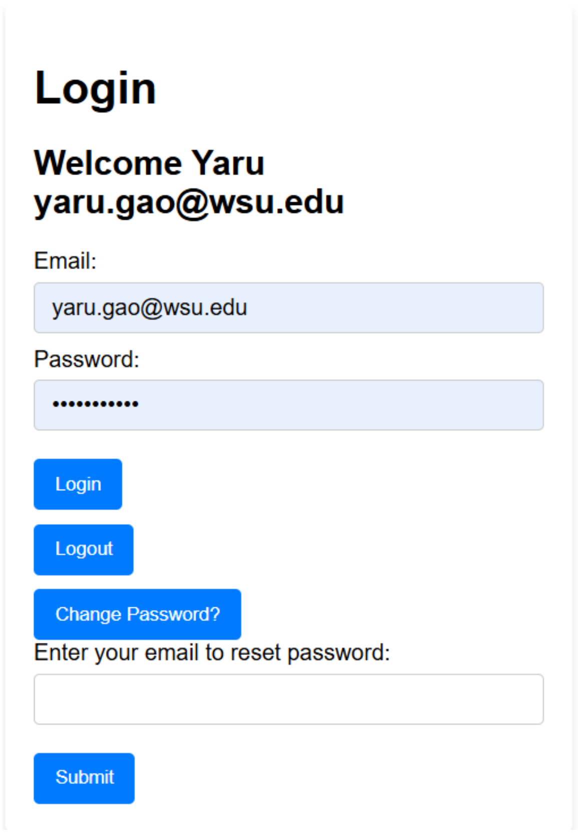


Profile page

User Name: Yaru

Email: yaru.gao@wsu.edu

On the profile page, you're granted a comprehensive view of every piece of data you've shared with our community. If you ever notice any inaccuracies or wish to make updates, the edit feature is at your service. And for those moments when you decide some information is best kept private or removed, the delete option is there to ensure your content remains exactly how you want it.



Login

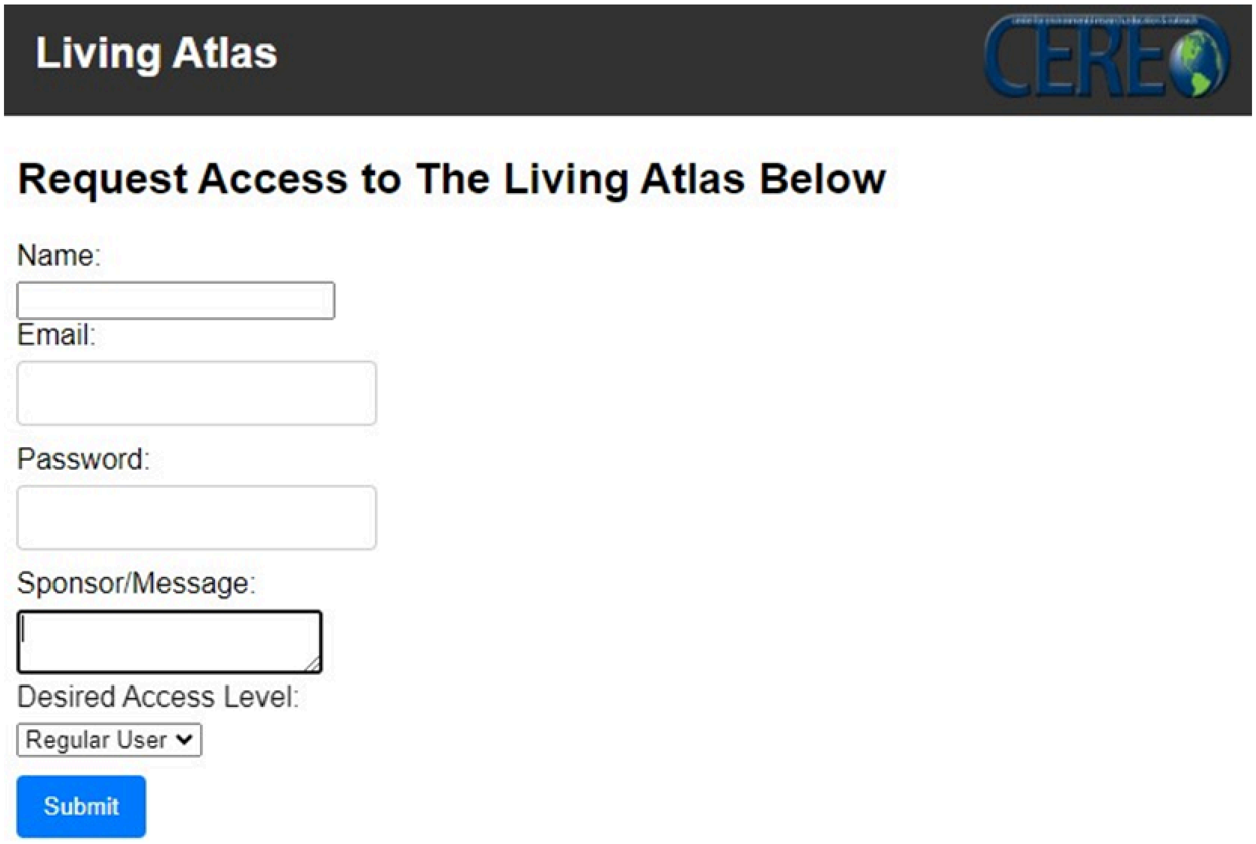
Welcome Yaru
yaru.gao@wsu.edu

Email: yaru.gao@wsu.edu

Password: •••••••••••

Enter your email to reset password:

Appendix M: Mockup diagram of the registration page, showing the user account request form.



Request Access to The Living Atlas Below

Name:

Email:

Password:

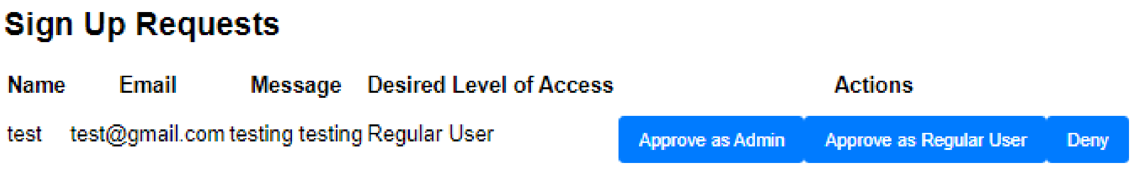
Sponsor/Message:

Desired Access Level:

Regular User

Submit

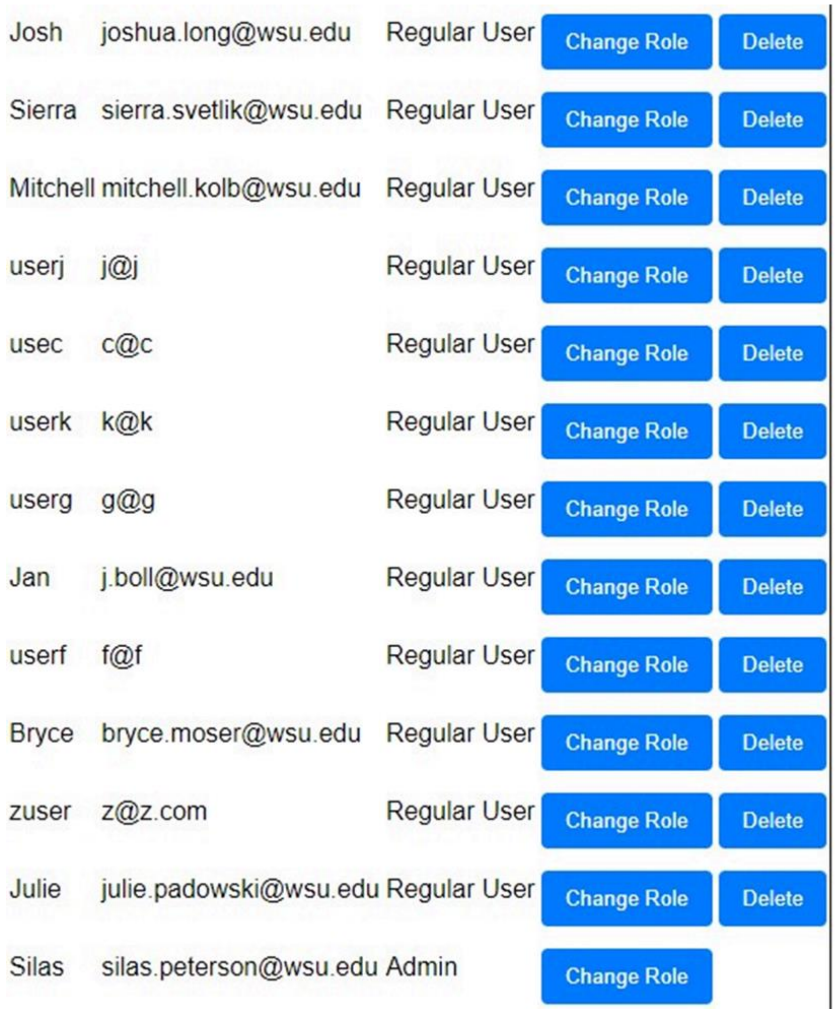
Appendix N: Mockup diagram of the administrator panel, illustrating user request approvals.



Sign Up Requests

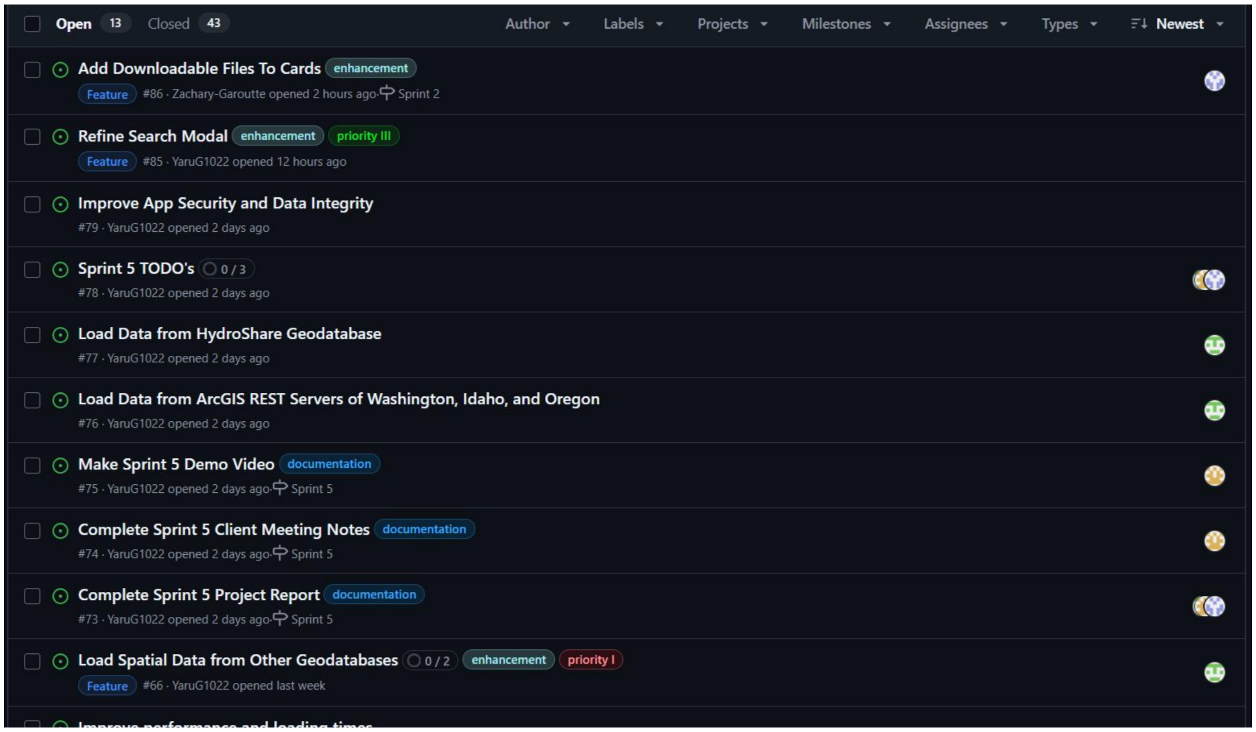
Name	Email	Message	Desired Level of Access	Actions
test	test@gmail.com	testing testing	Regular User	Approve as Admin / Approve as Regular User / Deny

Appendix O: Mockup diagram of the administrator dashboard, showing user management features for modifying access levels and deleting accounts.



Josh	joshua.long@wsu.edu	Regular User	Change Role	Delete
Sierra	sierra.svetlik@wsu.edu	Regular User	Change Role	Delete
Mitchell	mitchell.kolb@wsu.edu	Regular User	Change Role	Delete
userj	j@j	Regular User	Change Role	Delete
usec	c@c	Regular User	Change Role	Delete
userk	k@k	Regular User	Change Role	Delete
userg	g@g	Regular User	Change Role	Delete
Jan	j.boll@wsu.edu	Regular User	Change Role	Delete
userf	f@f	Regular User	Change Role	Delete
Bryce	bryce.moser@wsu.edu	Regular User	Change Role	Delete
zuser	z@z.com	Regular User	Change Role	Delete
Julie	julie.padowski@wsu.edu	Regular User	Change Role	Delete
Silas	silas.peterson@wsu.edu	Admin	Change Role	

Appendix P.1 and P.2: The GitHub issues board used to organize development on this project, and the project board that the issues are organized into.



Open 13 Closed 43 Author Labels Projects Milestones Assignees Types Newest

Add Downloadable Files To Cards enhancement
Feature #86 · Zachary-Garoutte opened 2 hours ago Sprint 2

Refine Search Modal enhancement priority III
Feature #85 · YaruG1022 opened 12 hours ago

Improve App Security and Data Integrity
#79 · YaruG1022 opened 2 days ago

Sprint 5 TODO's 0 / 3
#78 · YaruG1022 opened 2 days ago

Load Data from HydroShare Geodatabase
#77 · YaruG1022 opened 2 days ago

Load Data from ArcGIS REST Servers of Washington, Idaho, and Oregon
#76 · YaruG1022 opened 2 days ago

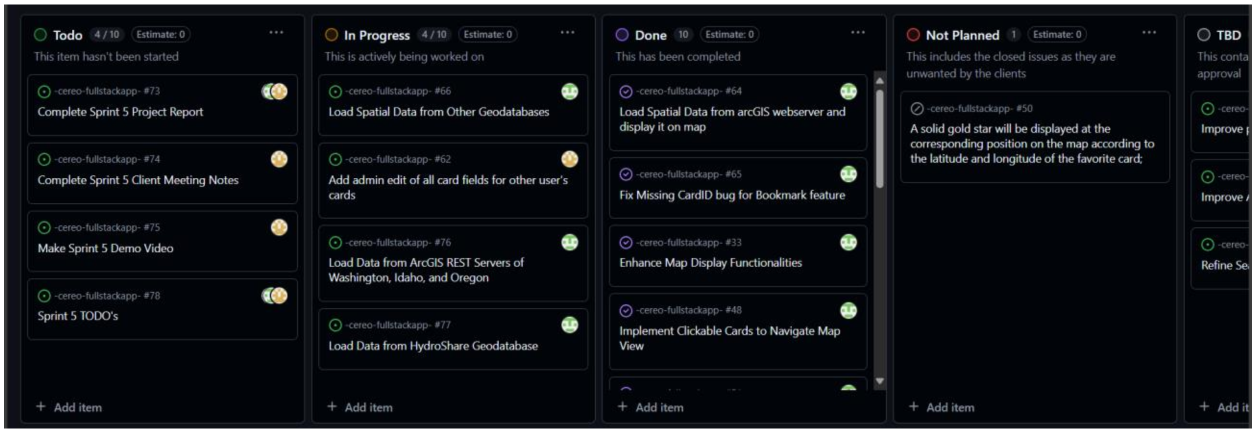
Make Sprint 5 Demo Video documentation
#75 · YaruG1022 opened 2 days ago Sprint 5

Complete Sprint 5 Client Meeting Notes documentation
#74 · YaruG1022 opened 2 days ago Sprint 5

Complete Sprint 5 Project Report documentation
#73 · YaruG1022 opened 2 days ago Sprint 5

Load Spatial Data from Other Geodatabases 0 / 2 enhancement priority I
Feature #66 · YaruG1022 opened last week

Improve performance and loading times



Todo 4 / 10 Estimate: 0
This item hasn't been started

-cereo-fullstackapp- #73
Complete Sprint 5 Project Report

-cereo-fullstackapp- #74
Complete Sprint 5 Client Meeting Notes

-cereo-fullstackapp- #75
Make Sprint 5 Demo Video

-cereo-fullstackapp- #78
Sprint 5 TODO's

+ Add item

In Progress 4 / 10 Estimate: 0
This is actively being worked on

-cereo-fullstackapp- #66
Load Spatial Data from Other Geodatabases

-cereo-fullstackapp- #62
Add admin edit of all card fields for other user's cards

-cereo-fullstackapp- #76
Load Data from ArcGIS REST Servers of Washington, Idaho, and Oregon

-cereo-fullstackapp- #77
Load Data from HydroShare Geodatabase

+ Add item

Done 10 Estimate: 0
This has been completed

-cereo-fullstackapp- #64
Load Spatial Data from arcGIS webserver and display it on map

-cereo-fullstackapp- #65
Fix Missing CardID bug for Bookmark feature

-cereo-fullstackapp- #33
Enhance Map Display Functionalities

-cereo-fullstackapp- #48
Implement Clickable Cards to Navigate Map View

+ Add item

Not Planned 1 Estimate: 0
This includes the closed issues as they are unwanted by the clients

-cereo-fullstackapp- #50
A solid gold star will be displayed at the corresponding position on the map according to the latitude and longitude of the favorite card;

+ Add item

TBD
This conta
approval

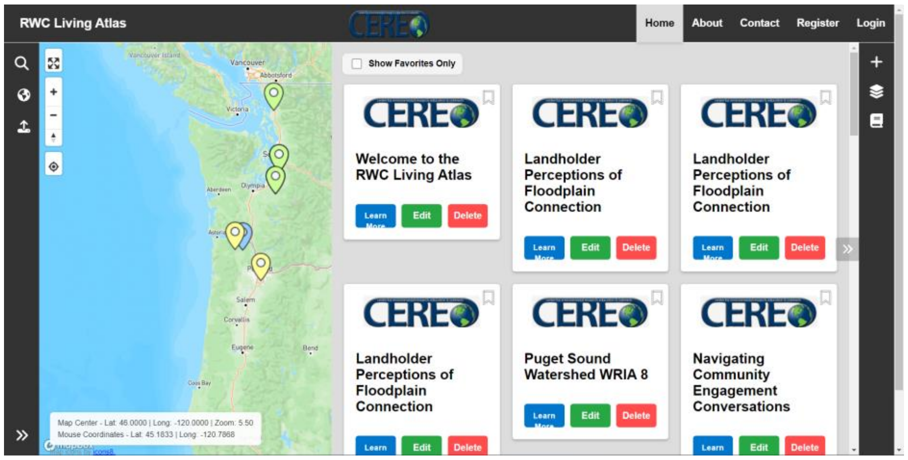
-cereo-
Improve

-cereo-
Improve

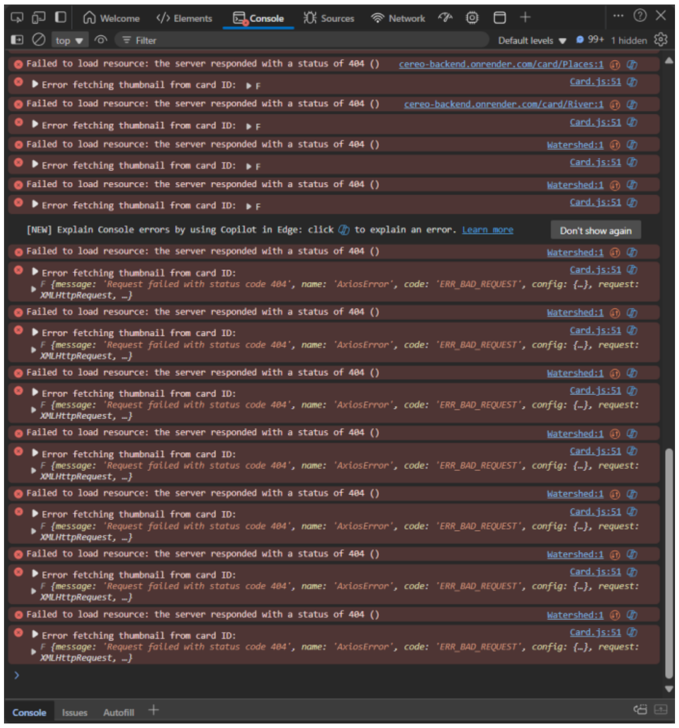
-cereo-
Refine Se

+ Add it

Appendix R: Drag the left edge of card container to expand it.

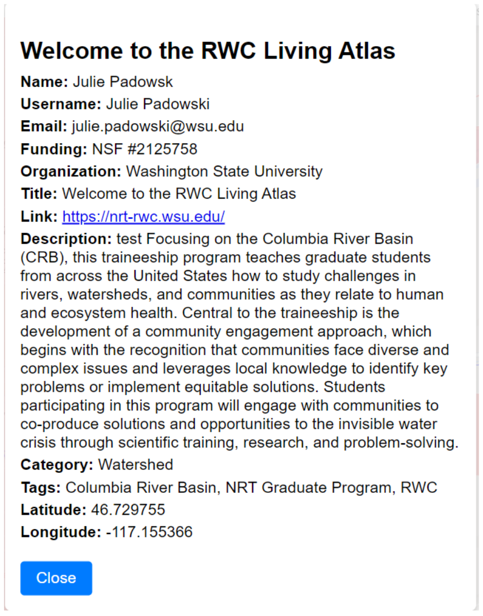

RWC Living Atlas
Home
About
Contact
Register
Login
Show Favorites Only
Welcome to the RWC Living Atlas
Landholder Perceptions of Floodplain Connection
Landholder Perceptions of Floodplain Connection
Landholder Perceptions of Floodplain Connection
Puget Sound Watershed WRIA 8
Navigating Community Engagement Conversations
Learn More
Edit
Delete
Map Center - Lat: 46.0000 | Long: -120.0000 | Zoom: 5.50
Mouse Coordinates - Lat: 45.1833 | Long: -120.7868

Appendix S: Example usage of DevTools for testing



```
Failed to load resource: the server responded with a status of 404 ()   cereo-backend.onrender.com/card/Places:1
▶ Error fetching thumbnail from card ID:  ▶ F                             Card.js:51
Failed to load resource: the server responded with a status of 404 ()   cereo-backend.onrender.com/card/River:1
▶ Error fetching thumbnail from card ID:  ▶ F                             Card.js:51
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:  ▶ F                             Card.js:51
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:  ▶ F                             Card.js:51
[NEW] Explain Console errors by using Copilot in Edge: click  to explain an error. Learn more   Don't show again
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
Failed to load resource: the server responded with a status of 404 ()   Watershed:1
▶ Error fetching thumbnail from card ID:                                  Card.js:51
▶ F {message: 'Request failed with status code 404', name: 'AxiosError', code: 'ERR_BAD_REQUEST', config: {…}, request:
  XMLHttpRequest, …}
>
```

Appendix T: Learn-More Modal of a Card



Welcome to the RWC Living Atlas

Name: Julie Padowsk
Username: Julie Padowski
Email: julie.padowski@wsu.edu
Funding: NSF #2125758
Organization: Washington State University
Title: Welcome to the RWC Living Atlas
Link: https://nrt-rwc.wsu.edu/
Description: test Focusing on the Columbia River Basin (CRB), this traineeship program teaches graduate students from across the United States how to study challenges in rivers, watersheds, and communities as they relate to human and ecosystem health. Central to the traineeship is the development of a community engagement approach, which begins with the recognition that communities face diverse and complex issues and leverages local knowledge to identify key problems or implement equitable solutions. Students participating in this program will engage with communities to co-produce solutions and opportunities to the invisible water crisis through scientific training, research, and problem-solving.
Category: Watershed
Tags: Columbia River Basin, NRT Graduate Program, RWC
Latitude: 46.729755
Longitude: -117.155366

Appendix U: Edit Modal of a Card

09000