COS 397: Computer Science Capstone

Blue Marble Geographics
Geospatial Data Portal
Critical Design Review Document

Team Undershrub
Anthony Jackson
David Sincyr III
Devin Carter
Grant Shotwell
Steven Kaplan

Dec. 13th 2021

Table of Contents Page

Document History

Name	Date	Reason for Changes	Version
David Sincyr	11/8/2021	Initial Creation	1.0
David, Anthony, Devin	11/10/2021	Completed various sections	1.1
Grant Shotwell	11/12/2021	Finished analysis section	1.2
Devin	11/13/2021	Added new content to the analysis section (all the to-dos that were not completed the day prior)	1.3
Stephen Kaplan	11/13/2021	Added details to various sections, added conclusion section	1.4

1.0 Synopsis

1.1 Executive Summary

Blue Marble Geographics offers professional-grade Geographic Information System (GIS) and geodetic software tools. Their Geospatial Data Portal seeks to provide these same tools to users but on an online platform. This route will supply its users with a way to remotely store, process, analyze and visualize their GIS data within a scalable environment. Additionally, the portal will allow its users to automate the processing and visualization of GIS data through a Python script editor. To provide these features to users, the Geospatial Data Portal will utilize Amazon Web Services (AWS) to provide scalable and high-performance servers to its users. Furthermore, the portal will employ Amazon's Lambda serverless computing environment to execute user-supplied code securely. Finally, the portal will utilize JS React for front-end development and Django for its back-end development. There will be three different user interface types in order to provide different view styles for different types of data. These choices will give users the best possible experience, meet all their needs, and allow administrators to control and maintain the portal effortlessly.

1.2 Preface

There are special circumstances that need to be met in order to meet the requirements. GIS data files take a lot of disk space so the portal needs to be scalable. Data visualization tools must be able to handle specific file types. Metadata will have to be searchable so that group members or anonymous users will be able to access the data. Lastly, the data portal has to be able to be modular so that more features and new pages can be added in quickly.

In order to prepare for these requirements, the group had to do a lot of self-study and complete tutorials. This assisted in the group's ability to provide a better product to the customer. The data portal targets users, from single individuals to large businesses that use GIS software. The goal is to show a need to move from their desktop or online software and to move to our online platform.

There is a reason I make a distinction between a caption and the title for a figure. The list of figures can include the title of the figure without the longer, detail caption.

1.3 List of Figures

SRS Figures

SDD Figures

UIDD Figures

1.4 List of Tables

SRS Tables

SSD Tables

UIDD Tables

1.5 Nomenclature Glossary? New page?

Within this section, you will find a listing of symbology, letters, signs, and terminology which are used throughout the project. Included alongside each of these parameters is a short description providing a definition or explanation of the term.

- Geographic Information System (GIS): A Geographic Information System is a framework that allows someone to describe or understand spatial and geographic data.
- Vector Data: Vector data is a way to represent real-world features such as trees, rocks, and buildings within a GIS environment. This is done through interconnected vertices that describe positions in the X, Y & Z axis. When these

- Paths: While a point is a single vertex, when a line intersects between two or more vertices it is referred to as a path. An example of this would be a marked road on a map.
- Raster Data: While normally this term is defined as a graphical mechanism to represent images in a grid of squares, within the scope of this project the term raster data is in reference to photographs or imagery taken from aerial viewpoints like satellites.
- Parcels: This term refers to an extended area of land, within the scope of this project a parcel is measured in whatever format and constraints are supplied by the user in their personal storage database.
- Virtual Private Cloud (VPC): This Amazon product is used to control virtual networking environments, allowing you to group EC2 instances and other AWS resources in an isolated section of the AWS cloud protected by custom access rules.
    - VPC Subnet: A VPC subnet is a network within a larger VPC network environment representing a logical division of the larger network's resources and IP address space.
- Elastic File System (EFS): This Amazon product allows database and storage systems to automatically resize when adding or removing files.
- Relational Database Service (RDS): This is an Amazon product that allows a user to operate databases in a cloud environment, automating administration tasks such as patching and backups. A more in-depth guide can be found in *bibliography section 3*.
- Elastic Cloud Compute (EC2): This is an Amazon Web Service (AWS) product that provides secure, stable, and scalable cloud computing for the project. A more in-depth guide can be found in *bibliography section 5.*
    - EC2 Autoscaling: This service provides the ability to automatically remove and add EC2 instances whenever needed or defined, allowing for dynamic and scalable services.
    - EC2 Network Load Balancer: This service automatically distributes incoming web traffic to multiple sub-servers to ensure EC2 instances, along with other AWS resources, are not harmfully overloaded, preventing things such as; server imbalance, server damage, or server slowdown.
- MySQL database: This is an open-source database management system and service which can be used as a template for handling large amounts of data and multiple types of backend libraries, programs, and tools. A more

in-depth guide can be found in bibliography section 4.

- Lambda: Amazon Web Service (AWS) Lambda is a serverless computing service that lets you test or run code on-demand in an isolated environment without impact on other system services and resources. A more in-depth guide of this can be found in *bibliography section 6*.
- Point Cloud: Point Clouds are a set of points which represent a 3D object or area, within the confines of this project it will often include environment geography notes such as trees or vague shapes of terrarium.
- Lidar (x, y, z): This is a form of scanning that uses lasers and varied time/distance increments to measure out 3D objects and environments, classifying it under the same terms as point-cloud.
- Metadata: Within the confines of this project, the term metadata refers to data about data which includes; coordinates of longitude and latitude, name of the file, 15 - 20 extra coordinate fields, text description, and date of creation.

1.6 Summary Summary of what?

The Blue Marble Geographics Geospatial Data Portal will offer users an online data visualization and manipulation tool. As seen in Section 1.5 (Nomenclature), there are a lot of specific terms that refer to geospatial data. Some terms of note are the data types Lidar, point cloud, and raster, which create large files that will need to be stored in a scalable environment. Users will be able to upload these files to the database to save their data online, share it with a user-defined group or set it to be publicly accessible so any user can access the data. In addition, a search feature will be available for users to access that data which will use north and south latitude as well as east and west longitude coordinates to allow users to get the exact data they want. This search feature will utilize metadata that will be stripped off a user's data when they upload and save it to the database.

The data visualization and manipulation feature will need to handle all the data types listed in Section 1.5. Furthermore, to manipulate the data faster, the built-in Python script editor will be able to take the data and manipulate it through Blue Marble Geographics modules and use publicly accessible libraries such as Numpy. Finally, concerning the system administrator, there will be a separate interface that will grant them special features to properly manage the system, such as monitoring data usage, site performance, and site maintenance.

The Geospatial Data Portal will utilize various aspects of Amazon Web Services to provide these features to users. Elastic File System (EFS) and Simple Storage Service (S3) instances will meet the scalability requirements for user storage by automatically resizing when users upload or delete their data. The Virtual Private

Cloud (VPC) service will allow control over the virtual network environment, such as resource allocation, security measures, and connectivity controls. The Relational Database Service (RDS) will maintain associations between users, data, projects, and authentication tokens, and will support queries on these associations from other services in our AWS cloud. Finally, to provide robust tools for web hosting and data visualization, the Elastic Cloud Compute (EC2) service will provide a secure and scalable computing environment for the visualization and manipulation of geospatial data. Furthermore, this service will allow managers and administrators to be able to add or remove more cloud computing resources when needed, which will add scalability. The EC2 Network Load Balancer will automate the balancing of resources, such as creating multiple regional instances. If one region goes down, another region will pick up the load so that our users can still use the data portal. AWS Lambda will allow user-supplied code to be run in a secure and safe environment as well as offer more computing power.

The system will use MySQL as a database management system. Autoscaled RDS instances will host a system-wide database partitioned across multiple availability zones. EC2 instances will host Django web servers that serve a front-end React website to users. Django is a web framework that offers many built-in features. Such features include: scalability for when new features are added, security by design approach meaning security was focused on in the development process, and content administration making management more effortless. The front-end will use React, which is an open-source JavaScript library that simplifies the design and development of web applications. Combining all of these features, the Geospatial Data Portal will be able to meet the growing needs of individuals or groups who use GIS software.

2.0 Details

2.1 Introduction

This section of the CDRD contains sections; 2.2 (purpose), 2.3 (methods), 2.4 (design), 2.5 (equipment), 2.6 (analysis) and Section 2.7 (conclusion). Section 2.2 summarizes the purpose of the Geospatial Data Portal project, while Section 2.3 details the methods in which the project is being created and how the features fulfill the purpose. Section 2.4 covers the rules and practices for the design of the software on a visual level, and Section 2.5 details the equipment and software being used to create the visual design, backend, frontend, and documentation of the project. Finally, Section 2.6 details the original proposal to ascertain if the project has met the original requirements, and Section 2.7 details a conclusion of what's been completed on the project thus far.

2.2 Purpose

The Geospatial Data Portal project's purpose is to allow individuals and teams to access, view, and manipulate geospatial data in a remotely accessed database. Users will have the ability to upload python scripts which will let them modify the information in unique and useful ways, alongside being able to do so under a team where everyone can work cooperatively. This portal will help numerous fortune 500 companies and private users alike to view geospatial data which contains a large amount of information about the modern world. Examples of this information include site selection for construction, topographic analysis for mining and zoning, transportation modeling for large cities and suburbs alike, and incident mapping to find places that will experience disasters or to catalog where they previously occurred. These examples are just scratching the surface of possibility of how data can be used and viewed through the Geospatial Data Portal.

Who is doing the work?

2.3 Method

Team Undershrub began devising a solution for the Geospatial Data Portal project by laying out our initial vision of the product based on the project proposal provided by our client Blue Marble Geographics. We produced user stories based on this initial vision and informed by industry practices such as security by design. Our client provided guidance and clarification on these user stories in a Systems Requirement Review meeting. With these clarifications, we created a set of software requirements and sought agreement between our team and our client. Through this process, we identified several primary design requirements for a dynamic web portal, summarized as follows:

- The system must allow users to create, modify, and share GIS projects.
- The system must allow users to easily search for GIS projects and data.
- The system must support several GIS data formats.
- The system must allow users to create visualizations of GIS data using Python scripts.
- The system must allow users to run Python scripts without compromising the security of the overall system.
- The system must support storage usage tracking and payment thresholds.
- The system must utilize a modular design so that more features can be added later with ease.
- The system shall prioritize functionality and performance over appearance.

To meet these design requirements, the group decided to use several AWS services, such as EC2, RDS, and Lambda, to provide users with a secure way to run code and a scalable data storage environment. These services will also allow users to search for data quickly and offer website administrators an easy-to-use management system to make site control and moderation easier. Furthermore, the group decided to use a Django web server to communicate with AWS services and handle information flow between the frontend and backend resources. This choice will allow for a modular design because it contains a lot of tools and features that take some of the burdens off of the developers, making it so the deployment of new features will be quick. To render GIS projects, data visuals, and all other user interfaces, the group will use ReactJS for front-end development. ReactJS allows the group to break different user interface designs into simple components in an environment that focuses on runtime and performs better than other options such as angularJS. Furthermore, adding additional features will require less development time and a less steep learning curve than other options.

2.4 Design

Design Parameters and Constraints

- Client (Blue Marble Geographics) design matrix.
    - Colors:
        - Charcoal: background
        - Blue Marble: secondary background
        - Light Blue: primary buttons/actions
        - Sand: secondary buttons/actions
        - Earth: tertiary buttons/actions
- What items the Data Portal is designed to handle.
    - Data Portal must be capable of storing Metadata.
    - Not designed to store fully visualized data.
- Users must be able to navigate the portal without explanation as to what is where.
    - No tutorial needed for portal use.
- Must have a user profile.
    - With versions for personal view and non-personal view.
- Must have the capability to share and view shared projects.
    - Different share types including private (single or group) shares as well as public shares.
- Must have an accessible python editor.
    - Users should be able to edit/manipulate their code.

The design of the Geospatial Data Portal is constrained by technical, organizational, and procedural factors. Blue Marble Geographics enforces a Brand Guide which details requirements for colors, typography, and iconography. The Brand Guide specifies five colors that make up the Blue Marble Geographics official color palette: Light Blue (#4286BF), Blue Marble (#0E4D8C), Charcoal (#303036), Earth (#CDAF8A),

and Sand (#F2E1CC). Team Undershrub has placed additional constraints on the usage of these colors to simplify and standardize the design of user interfaces throughout the project.

Design Approach

- Ease of use: reference the design constraint for how easy the portal must be to use.
- Simple visuals.
    - Professional look, clean and uncluttered.
- Easy to read.
    - Text must have high contrast with the background.
    - Users must be capable of focusing on specific page elements without being distracted by another element, ex: a graphic.

Design Problems and Unusual Issues

- Quantity of buttons in the navigation bar (nav bar).
    - Limited number of spaces (five) that can be used for the nav bar.
    - Currently being solved via the use of drop-down windows in place of normal buttons.

Design Notes

- Scope:
    - Data visualization is required for a completed product, but not required by the client for our work on the product itself.
        - As such the framework for it is required at the least.
- Python Editor:
    - Must be capable of using an array of useful libraries that help with handling Geospatial Data.

2.5 Equipment & Services

The equipment which we made use of during this project has been completely online with softwares such as; *Github* for development and version control, *Framer* for creating UI documents and tests, *Trello* for keeping track of development and assigned work, *Google Drive* for documentation, *Django's* Framework for web design, *JSReact* for easy to create interactive UI, *Amazon's Web Services* for databasing and storage and *Google's* Authentication services for login and security.

2.6 Analysis

Requirements:

- SRIPTING: Creating, uploading, editing, and executing Python scripts. (SRS fig.2)
- SITE ADMINS: Manage roles, view/edit/add/remove files, and receive notifications. (SRS fig.3)

- DATA MANAGEMENT: Upload, download, delete, and sort data. (SRS fig.4)
- NOTICES: System announcements, automated data cap alerts, admins view user databases to send warnings. (SRS fig.5)
- PROJECT SHARING (skip, duplicate): Share projects to specific users or get public access links. (SRS fig.6)
- GROUP MODERATORS: Message users, view group projects, access permissions, upload notifications, access API. (SRS fig.7)
- IT HELP: Send development questions/requests, request admin assistance. (SRS fig.8)
- PROJECT MANAGEMENT: Search project, tag project, manage project versions, "save as", have multiple projects. (SRS fig.9)
- PROJECT VERSIONS: Search project from list, open project, view change history, name version, restore to version, open previous version. (SRS fig.10)
- PROJECT SHARING: Share to group, share with specific user, share publicly, share with link (extends), send invitation notification, view shared project, view public project. (SRS fig.11)
- PROJECT MANAGEMENT (skip, duplicate): (SRS fig.12)
- METADATA: View metadata, search by metadata. (SRS fig.13)
- VISUALIZATION PROCESS: Handled externally. (SRS fig.14)

The Geospatial Data Portal has many requirements related to scripting, admin/moderator capabilities, project management, and file management.

Scripts can be created, uploaded, and edited through the data portal's Python editor. After, scripts are executed within cloud-hosted containers.

The requirements listed group moderators to be able to message users, view projects within the group, access permissions, receive upload notifications, and access an API.

The requirements listed site administrators to be able to manage roles, view/edit/add/remove any file, and toggle notifications.

The requirements state various means through which users can manage, access, and share their projects and files.

Users are required to be capable of managing projects via tags and versions. Versions themselves must be capable of receiving names, numberings, and potentially a restoration. If a version is restored it may be either as a new project or in replacement of an existing project. Tagging and version control are handled by the project owner and trusted individuals through the data portal UI. Site staff may also assist the project owner.

Projects are required to be searchable via both tags and version details. For the former of these two functions, this will exist as a filtering feature for the project search UI element. The latter of these two will exist for project owners, and trusted individuals, who are searching through a particular project internally. A lesser version of the version search will exist as a feature of the standard project search UI element, but its usefulness will be less than for someone looking at all versions of a project they own.

The data portal is required to provide functionality for sharing a project with others. Project sharing must be possible at an individual, group, and public level. The sharing feature will also allow for sharing a link to the project. All of these features should be handled by a UI element similar to typical online collaborative work products.

The data portal is required to provide the basic functionality of file management to project owners and their trusted individuals. File management includes removing, editing, and uploading. This feature will be available while within a project view.

The contents of a project can only be shared by sharing the project. This is a required feature that defines the purpose of the data portal compared to other collaborative work products.

The data portal is required to be capable of visualizing Geospatial Data. The task of creating this visualization is not currently required of this team (Undershrub). Providing the framework to implement this feature is however required, as data visualization is a core feature of the data portal.

2.7 Conclusions

Blue Marble Geographics has requested a Geospatial Data Portal website that allows users to upload, manipulate, visualize, and share geospatial data. The Geospatial Data Portal must support these features for Lidar, point cloud, and raster data files, which users must be able to upload. Manipulation and visualization of GIS data shall be accomplished via user-created Python scripts, which the system must allow users to upload and modify using an in-browser code editor. Users shall then be able to run their Python scripts, producing modifications and visualizations of their uploaded data files. Several data files, scripts, and descriptions make up a GIS project, which users may own or contribute to one or more of.

Team Undershrub has designed a system capable of supporting the requested and required features of a Geospatial Data Portal. Our system relies on a robust AWS cloud architecture to provide a fast, reliable, secure, and scalable application. Load balancing, autoscaling, and multiple availability zones are used to meet the storage

and computation requirements of the system at all times, as well as to ensure the near-constant availability of the system.

3.0 Addendum

3.1 References

1. Dr. Yoo, Terry. “COS397_22_Writing-gooder-Yoo-handouts” https://courses.maine.edu/d2l/le/content/179072/viewContent/6032605/View
2. System Requirements Specification
3. System Design Document
4. User Interface Design Document

3.2 Bibliography

1. Daly, Donald J. “Economics 2: EC2.” *Amazon*, CGA Canada Publications, 1987, https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc.
2. Dr. Yoo, Terry. “COS397_22_Writing-gooder-Yoo-handouts” https://courses.maine.edu/d2l/le/content/179072/viewContent/6032605/View
3. “How to Write an Executive Summary - Projects at Harvard.” *How to Write an Executive Summary*, Harvard Kennedy School Communications Program, https://projects.iq.harvard.edu/files/hks-communications-program/files/how_to_write_an_exex_summ_to_use_4_18_18.pdf.
4. MUSGRAVE, DAVID. “Lambda.” *Amazon*, EUROPA EDITIONS, 2022, https://aws.amazon.com/lambda/.
5. “MySQL 8.0 Reference Manual :: 1.2.1 What Is Mysql?” *MySQL*, https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.
6. “RDS: Amazon Relational Database Service.” *Amazon*, Research Defense Society, 2007, https://aws.amazon.com/rds/.

3.3 Acknowledgement

We would like to provide our deepest appreciation to Blue Marble Geographics as they provided us with the opportunity to work on the Geospatial Data Portal. Without their continued support and assistance we would not have been able to get to the point we’re currently at now, we hope to continue this project as it has been an enriching and fulfilling experience for us thus far.

In particular, we would like to thank Chief Technology Officer Victor Minor, Director of Product Management Sam Knight, Product Manager Katrina Schweikert, Quality Assurance Manager Stephanie Martini, and Product Manager Jeffery Hatzel for working closely with us during this development period.

3.4 Appendix

A. The System Requirements Specification document begins on page 21.
B. The System Design Document begins on page 100.
C. The User Interface Design Document begins on page 123.

3.4.1 Document Contributions

Grant Shotwell **Contributions**: 5%	- Created initial information for section 2.6. - Helped with the structure of the initial document.
Stephen Kaplan **Contributions**: 15%	- Edited document for grammar. - Added in appendix information and prior documentation bits. - Wrote section 2.7 (conclusion) - Assisted with section 1.5's nomenclature.
Devin Carter **Contributions**: 20%	- Section 2.4 Design information. - Section 2.5 Analysis tag/version management onwards. - Document review and comments.
David Sincyr III **Contributions**: 35%	- Created initial documentation. - Completed section 1.6 (summary) - Completed section 1.2 (preface) - Completed section 3.4 (appendix) - Completed section 2.3 (method) - Completed section 1.3 (figures) - Completed SRS for section 1.4 (tables) - Overall editing the document - Fixed many errors due to the conversion to different formats.
Anthony Jackson **Contributions**: 25%	- Completed part of section 1.4 → UIDD and SSD Tables. - Completed section 1.5 (nomenclature), - Assisted with editing section 1.6 (summary) - Completed section 2.1 (introduction) - Completed section 2.2 (purpose) - Completed section 2.5 (equipment & Services) - Filled in Bibliography with some of the needed references.

Appendix A - System Requirements Specification

Name	Date	Reason for Changes	Version
All team members	10/4/2021	Initial Creation	1.0
Devin Carter	10/30/2021	Added Client changes	1.1
David Sincyr III	10/30/2021	Added missing info	1.2
David Sincyr III	12/9/2021	Added Dr. Yoo's changes	1.3

A.1. Introduction

The goal of this project is to provide a dynamic hostable web application for Blue Marble Geographics. This web portal will provide Blue Marble Geographics with a way to supply its clients with a web-based database that will allow its users or groups of users to manage, analyze, organize, characterize, render and visualize their geospatial data. Additionally, this web portal will enable Blue Marble Geographics system administrators to monitor each individual or group of users' data usage to charge their customers for using their service correctly. Lastly, this is a capstone project for Blue Marble Geographics, in partial fulfillment of the Computer Science BS degree for the University of Maine.

A.1.1 Purpose of This Document

The purpose of this document is to provide a detailed description of a dynamic web portal. This document will give an overall description of the web portal. Additionally, It will detail the purpose as well as the various features of the system to include the interfaces, what the system will do, and assumptions that are needed for the web portal.

The intended readership of this document is the project manager, the quality assurance manager, system developers, and system owner. Furthermore, this document is meant for all stakeholders to analyze the web portal prior to deployment to further their understanding of the system's functionalities.

A.1.2 References

1. Blue Marble Geographics. (2021). Global Mapper Pro (23.0) [Desktop application]. Blue Marble Geographics. https://www.bluemarblegeo.com/global-mapper-pro/
2. Lucid (2021). Lucid Visual Collaboration Suite [Online app]. Online. https://lucid.app/

A.1.3 Purpose of the Product

The dynamic web portal will allow its users to upload their geospatial data to an online database. Once in the database, they will have the ability to share, manipulate and visualize their data with others. Additionally, users will be able to automate this process by using python scripts through a built-in python script editor. Blue Marble Geographics does have a desktop application but does not currently have an online platform to be able to do this for their customer base, and there is currently space in the marketplace for this type of online tool to succeed.

A.1.4 Product Scope

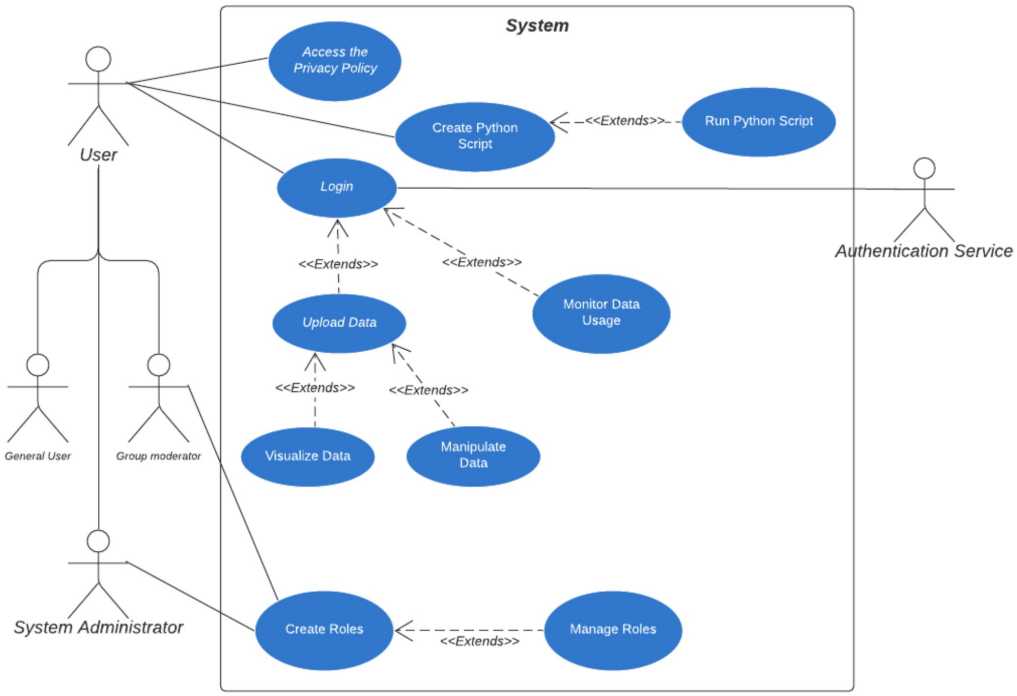

Figure 1: This UML diagram is a high level view of the whole system and its actors

This section identifies the boundary between the system under development and the outside world. That is, it identifies what is included in the system and what is not. Typically, a context diagram best describes the boundary. However, because the systems in this class are small, we will use a combination top-level use case and context diagram. In addition to referring the reader to the diagram, give a brief summary of how it illustrates the system's scope. Make sure to number the use cases in the diagram.

The dynamic web portal will be able to store, manipulate and provide an online medium for users. It will not be used as a form of social interaction between users outside of a group or between the Blue Marble Geographics company.

A.2. Functional Requirements

A.2.1 Use Case Specification

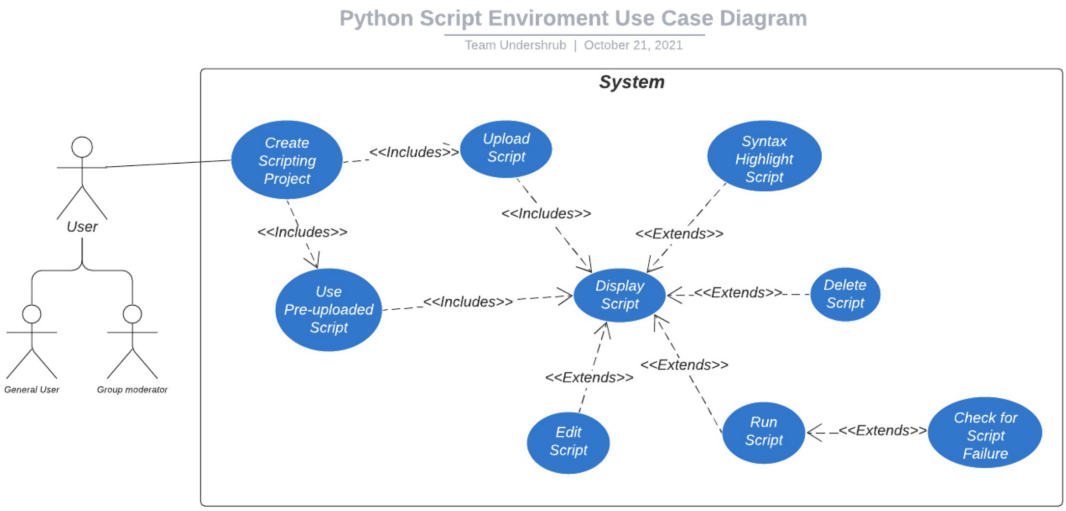

Figure 2: This details the how a user will interact with the python script environment

Number	1	
Name	Create Scripting Project	
Summary	A user creates a python script	
Priority	4	
Preconditions	A user needs to be authenticated	
Postconditions	A user creates a scripting project	
Primary Actor	Users with an account	
Secondary Actors	None	
Trigger	User clicks the create Python script button	
Main Scenario	**Step**	**Action**
	1	User is authenticated
	2	User clicks on create python script
	3	Python editor appears on screen
Extensions	**Step**	**Branching Action**
	2a	User is prompted to upload or use pre-uploaded script **:** 2a1. User clicks an option
Open Issues		

Number	2	
Name	Use Pre-uploaded Script	
Summary	This option allows a user to load an already existing script	
Priority	4	
Preconditions	A python script must already exists	
Postconditions	The python script is loaded and displayed	
Primary Actor	A paid user	
Secondary Actors	Python Scripting Environment	
Trigger	The user selects use Pre-uploaded script	
Main Scenario	**Step**	**Action**
	1	The user selects the use pre-loaded script option
	2	The user selects the script they want to use
	3	The system loads the script
	4	The system displays the script
Extensions	**Step**	**Branching Action**
	1a	The user selects the script they want to use **:** 1a1. The user selects the script they want to use from a list
Open Issues		

Number	3	
Name	Upload Script	
Summary	The user uploads a script they wrote	
Priority	4	
Preconditions	The user clicks the upload script button after writing a script	
Postconditions	The users script is uploaded and save to the system	
Primary Actor	A paid user	
Secondary Actors	Python script environment	
Trigger	A user clicks the upload script button	
Main Scenario	**Step**	**Action**
	1	A user writes a script in the python script editor
	2	The user navigates to the upload script button
	3	The user clicks the upload script button
	4	The system uploads and saves the script to the database
Extensions	**Step**	**Branching Action**
Open Issues		

Number	4	
Name	Display Script	
Summary	This brings up all scripts owned by the user	
Priority	4	
Preconditions	A python script must be already created	
Postconditions	All scripts owned by the user will be displayed on screen	
Primary Actor	A paid user	
Secondary Actors	Python script environment	
Trigger	The user opens the Python script editor	
Main Scenario	**Step**	**Action**
	1	The user navigates to the python script editor
	2	the user clicks the python script editor button or menu option
	3	The python script editor window appears on screen
Extensions	**Step**	**Branching Action**
Open Issues		

Number	5	
Name	Syntax Highlight Script	
Summary	Specific portions of code written by the user is highlighted such as keywords	
Priority	5	
Preconditions	Text must be present in the python script editor	
Postconditions	Text will be highlighted	
Primary Actor	Paid user	
Secondary Actors	The user must be inside the python script editor environment and script editor	
Trigger	The user starts to type text	
Main Scenario	**Step**	**Action**
	1	The user creates or edits a script
	2	The user types in the script editor window
	3	The script editor syntax highlighter highlights pre-specified text
Extensions	**Step**	**Branching Action**
	1a	The user creates or edits a script **:** 1a1. Create Scripting Project 1a2. Edit Script
Open Issues		

Number	6	
Name	Edit Script	
Summary	Allows a user to be able to edit a script	
Priority	4	
Preconditions	An existing script is already present	
Postconditions	The selected script has been edited	
Primary Actor	A paid user	
Secondary Actors	Python script environment	
Trigger	A user selects a script and clicks the edit button	
Main Scenario	**Step**	**Action**
	1	The user selects an already available script from their list
	2	The user navigates to the edit script button
	3	The user clicks the edit script button
Extensions	**Step**	**Branching Action**
Open Issues		

Number	7	
Name	Delete Script	
Summary	The user deletes their script	
Priority	3	
Preconditions	The user is in the python editor environment and clicks the delete script button	
Postconditions	The users script is deleted	
Primary Actor	A paid user	
Secondary Actors	Python Script Editor	
Trigger	A user clicks the delete script button	
Main Scenario	**Step**	**Action**
	1	The user selects a python script
	2	The user clicks the delete script button
	3	The user receives confirmation that the script was deleted
Extensions	**Step**	**Branching Action**
Open Issues		

Number	8	
Name	Run Script	
Summary	Allows a user to run a python script	
Priority	4	
Preconditions	A user is in the python script environment	
Postconditions	A users script will have been run	
Primary Actor	Paid user	
Secondary Actors	Python Script environment	
Trigger	A user clicks the run script button	
Main Scenario	**Step**	**Action**
	1	A user creates a python script
	2	The user clicks on the run script button
	3	The user receives a result
Extensions	**Step**	**Branching Action**
	3a	The user receives a result **:** 3a1. Check for Script Failure
Open Issues		

Number	9	
Name	Check for Script Failure	
Summary	The system checks for the failure of a script due to syntax or other errors	
Priority	4	
Preconditions	The run script button is clicked	
Postconditions	Returns a result upon success or failure upon an error	
Primary Actor	Paid user	
Secondary Actors	Python Script environment	
Trigger	A user clicks the run script button	
Main Scenario	**Step**	**Action**
	1	A user clicks the run script button
	2	The python parser attempts to parse the code
	3	The result is returned
Extensions	**Step**	**Branching Action**
	3a	The result is returned**:** 3a1. The parser returns the result if no errors 3a2. The parser returns an exception and description if there is an error
Open Issues		

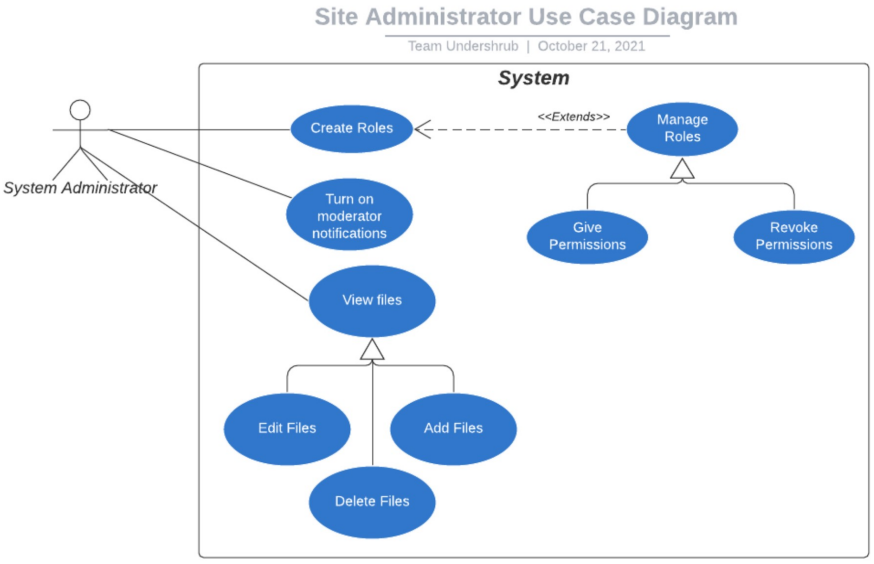

Figure 3: This depicts the options and how a system administrator will interact with the admin interface.

Number	10	
Name	Create Roles	
Summary	System Administrator creates one or more new roles.	
Priority	5	
Preconditions	System Administrator logs in.	
Postconditions	Successful creation of a role.	
Primary Actor	System Administrator.	
Secondary Actors	N/A	
Trigger	System Administrator attempts to create a new role.	
Main Scenario	**Step**	**Action**
	1	Attempt to create a new role.
	2	Assign title and permissions for the role.
	3	Assign roles to individuals.
Extensions	**Step**	**Branching Action**
	2a	< Using Pre-Existing Role > **:** Assign preset permissions.
Open Issues		

Number	11	
Name	Turn on Moderator notifications	
Summary	Notifications turned on to give notifications regarding moderation/administrative actions.	
Priority	2	
Preconditions	System Administrator logged in.	
Postconditions	System Administrator will be alerted to actions taken by other roles with administrative/moderation permissions.	
Primary Actor	System Administrator	
Secondary Actors	All roles with unique moderation/administrative permissions.	
Trigger	System Administrator attempts to turn on notifications for administrative actions.	
Main Scenario	**Step**	**Action**
	1	System Administrator attempts to turn on notifications for administrative actions.
	2	System Administrator selects types of actions to have notifications for.
	3	System Administrator selects groups to have notifications for.
Extensions	**Step**	**Branching Action**
	3a	< System Administrator Accepts All Notifications > **:** System Admin enables all notifications.
Open Issues		

Number	12	
Name	View Files	
Summary	System Administrator views files uploaded by other users.	
Priority	4	
Preconditions	Some amount of uploaded files exist.	
Postconditions	All files become viewable to the System Admin.	
Primary Actor	System Administrator	
Secondary Actors	General users & database.	
Trigger	System Admin attempts to check what files have been stored by other users.	
Main Scenario	**Step**	**Action**
	1	System Admin attempts to check the list of stored files.
	2	System retrieves a running list of files.
	3	View window displays all files to the System Admin.
Extensions	**Step**	**Branching Action**
	3a	< Narrow Search > **:** System Admin views a subsection of available files rather than the whole list.
Open Issues		

Number	13	
Name	Edit Files	
Summary	System Admin edits files previously uploaded.	
Priority	4	
Preconditions	Some amount of files have been previously uploaded, and the System Admin is currently viewing a list of files.	
Postconditions	System Admin successfully edits a file.	
Primary Actor	System Administrator	
Secondary Actors	General users & database.	
Trigger	System Admin attempts to alter existing files.	
Main Scenario	**Step**	**Action**
	1	System Admin selects a specific file from the view window for uploaded files.
	2	System Admin selects to see the contents of the file.
	3	System retrieves the specific information, and contents, of the selected file.
	4	Contents of the file are displayed to System Admin in a view window.
	5	Using the view window the System Admin makes alterations to the file.
Extensions	**Step**	**Branching Action**
	5a	Content Additions **:** System admin adds/uploads contents to the file.
Open Issues		

Number	14	
Name	Delete Files	
Summary	System Admin deletes a previously uploaded file.	
Priority	4	
Preconditions	Some amount of files have been previously uploaded, and the System Admin is currently viewing a list of files.	
Postconditions	System Admin successfully deletes an existing file.	
Primary Actor	System Administrator	
Secondary Actors	General users & database.	
Trigger	System Admin attempts to delete an existing file.	
Main Scenario	**Step**	**Action**
	1	System Admin selects a specific file from the view window for uploaded files.
	2	System Admin selects to delete the file.
	3	System responds to the request for deletion with a confirmation request.
	4	System Admin confirms the request for file deletion.
	5	File is deleted from the system.
	6	System alerts the System Admin of a successful file deletion.
Extensions	**Step**	**Branching Action**
	1a	< Multi Selection > **:** The System Admin selects multiple files for deletion.
	1a2	< Multi Deletion > **:** System Admin goes through the same process for deleting a single file, but in this situation deletes multiple files.
	4a	< Deletion Cancellation > **:** The System Admin decides to cancel the request for deleting the selected file.
	4a2	< Cancellation Response > **:** The System alerts the System Admin that the file deletion was cancelled.
Open Issues		

Number	15	
Name	Add Files	
Summary	System Admin uploads a brand new file to the database.	
Priority	5	
Preconditions	System Admin has a file to upload.	
Postconditions	New file uploaded successfully.	
Primary Actor	System Administrator	
Secondary Actors	Database	
Trigger	System Admin attempts a file upload.	
Main Scenario	**Step**	**Action**
	1	The System Administrator chooses to upload a file.
	2	System prompts System Admin for the file they wish to upload.
	3	System Admin responds to the prompt by choosing a file.
	4	File upload occurs following Admin submission.
	5	System notifies the System Admin of a successful upload.
Extensions	**Step**	**Branching Action**
	3a	< Multiple Uploads > **:** System Admin chooses to upload multiple files at once.
	5a	< Failed Upload > **:** System notifies System Administrator that their upload failed.
Open Issues		

Number	16	
Name	Manage Roles	
Summary	System Administrator adjusts the state of other roles.	
Priority	5	
Preconditions	System Admin has already created a list of roles.	
Postconditions	Existing role is updated with new changes.	
Primary Actor	System Admin	
Secondary Actors		
Trigger	System Admin views the list of roles with intent to adjust one or more of the roles.	
Main Scenario	**Step**	**Action**
	1	System Administrator attempts to view the list of existing roles.
	2	System displays a running list of roles.
	3	System Administrator selects one or more roles to change.
	4	System displays the list of permissions that the selected role(s) has access to.
	5	The System Admin makes either an addition or removal of permissions.
	6	The System makes a request for confirmation on the change.
	7	The System Admin confirms the change.
	8	The role is updated with whatever change that was submitted.
Extensions	**Step**	**Branching Action**
	5a	< Multi-Changes > **:** System Admin makes changes to a number of selected roles all at once.
	7a	< Change Cancellation > **:** The System Admin decides to cancel the request for altering an existing role.
	7a2	< Cancellation Response > **:** The System alerts the System Admin that the role changes were cancelled.
Open Issues		