KML Guard

GIS File Hosting Service

Requirement Specification <1.5>
February 20, 2019

Chelsea Marfil - 014400501
Anton Carrillo - 016216913
Christopher Greer - 016761912
Marc Marcelino - 014625700
Duong Pham - 013090075
Stakeholder/Project Visionary: Christopher Priebe

Requirement Specification Version History

Date	**Version**	**Description**	**Document Location**
February 18, 2019	1.0	Initial completed documentation by the software engineering team.	Requirement Specification <1.0>
February 20, 2019	1.5	Revisions made to documentation after comments and feedback received from G2SS stakeholder, Christopher Priebe. Major revisions include that the system is required to have two interfaces: the Web UI and API. DoD Agencies will access our system through our API rather than directly from our WebUI.	Current document.

Table of Contents

1. Executive Summary

1.1 System Context

KML Guard is a GIS (geographic information system) file hosting service with two main interfaces that will support a cross-domain file sharing requirement. The first interface is a user interface component allowing human users to manually upload individual KML files (and files of other types as well, but KML is the primary interest), along with basic metadata about the file being uploaded – who is uploading it and a brief description of the file. That information, along with the file itself, is accepted by the file upload GUI, and the file and its timestamped metadata is stored in the database. The server then also exposes a machine-to-machine web service API allowing other applications to query the server for information about KML files currently available, and to pull those files as a response to other API calls.

1.2 System Motivation - Problem to Solve

G2 Software Systems (G2SS) contracts with numerous DoD agencies including NORAD/US Northern Command. At NORAD, G2SS supports mission partners such as National Guard units, FEMA, state or local governments, etc., sharing GIS files, usually as KML, which provide amplifying information for the US DoD to provide support to those civilian authorities. In order to provide that support, the US DoD needs to engage their resources at the classified level, and so data provided by a state or county, for instance showing flood plain mapping data, needs to be transferred from an unclassified network to a classified one so that the DoD can engage some of their own mission systems to provide machinery, manpower, or other resources in support of civil authority.

1.3 Solutions Provided in this Document

We begin with an introduction giving context and an overview of the motivation for this software and how this software will solve the problem at hand. We will then provide a stakeholder model which gives an easy overview of the stakeholders involved in the project along with additional details about them such as their roles, goals and motivations, responsibilities and functions, their skills, and their level of priority. Then we will provide a goal model which is an overarching visual representation of our stakeholders' business goals which are broken down into usage goals and system goals. Then we will provide a system vision which illustrates the vision of the system and its core purpose and context. Following that, we will include a use case diagram and use case scenarios which will provide greater detail as to how the system and users will interact. Finally, we provide detailed requirements for our process, deployment, system constraints, and quality.

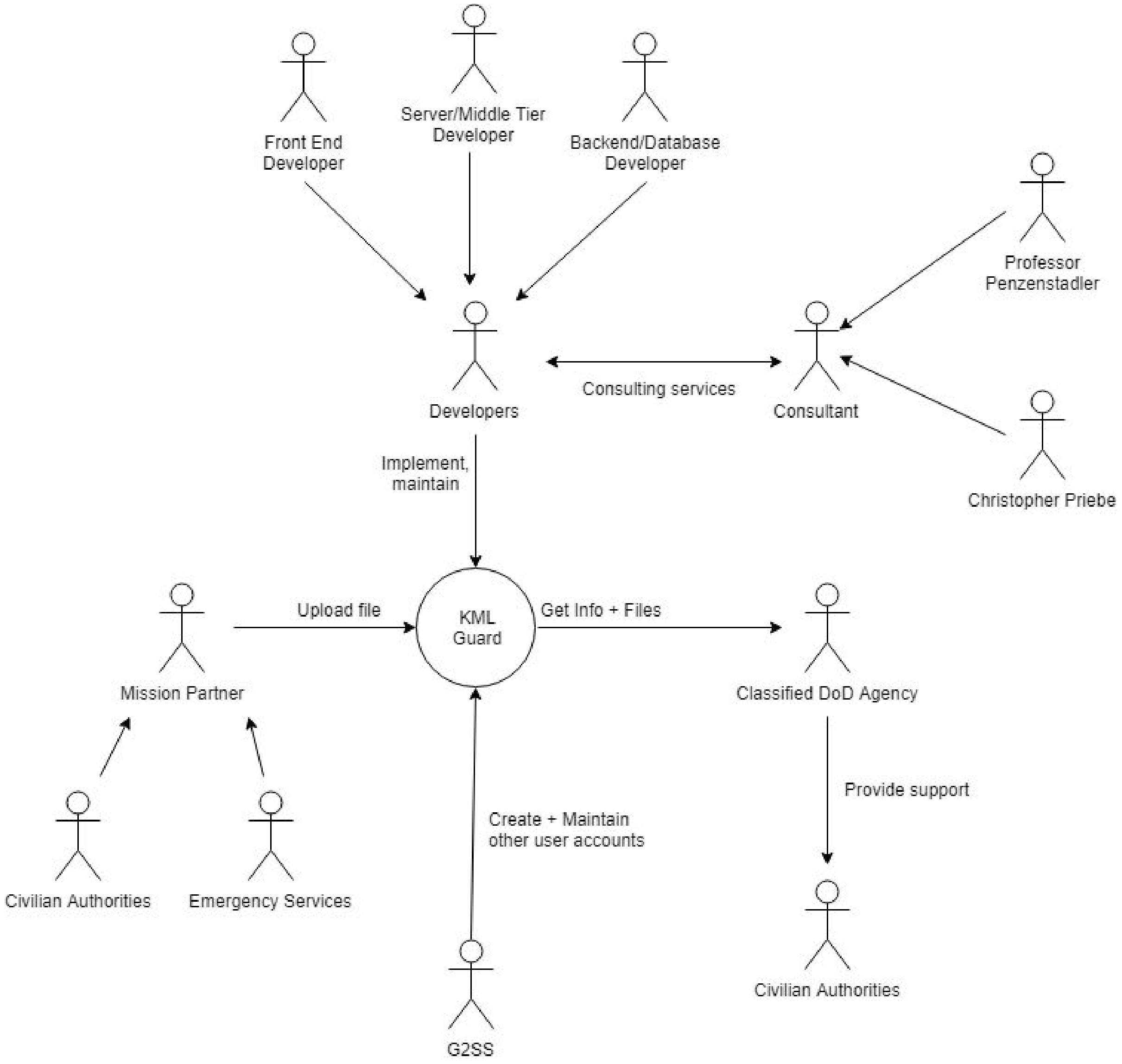
1.4 Impact - What This Helps, Why This Matters

The ultimate goal of this system is to support the movement of a file from an unclassified network via a guard to a classified one in order to provide the US DoD with crucial information needed to provide support to civilian authorities.

2. Stakeholder Model

A stakeholder is a person or entity with an interest in our KML Guard system. This includes individuals, groups, or institutions having the responsibility for system requirements. In this section, we provide a diagram featuring stakeholders and their relationships to the system and other stakeholders. Next, we provide stakeholder descriptions and relationships which further describe some stakeholders and relationships in the diagram. Finally, our stakeholder matrix details stakeholders' motivations, responsibilities/functions, skills, and their priority.

2.1 Stakeholder Diagram



This diagram features stakeholders and their relationships to the KML Guard system and other stakeholders.

2.2 Stakeholder Descriptions and Relationships

This section describes some stakeholders and their relationships, both to the system and to other stakeholders, as depicted in the diagram.

2.2.1 G2SS

"G2 Software Systems, Inc. is a recognized leader in systems engineering and custom software development for government and corporate clients." - https://g2ss.com

G2SS is the main client for whom we are engineering and developing this system. The G2SS representative that developers report to for system requirements and vision is Christopher Priebe. G2SS will mainly create and maintain user accounts on the system.

2.2.2 Developers

This group of stakeholders includes front end, server/middle tier, and backend/database developers. This group are software engineers who document, implement, and maintain the system.

2.2.3 Web Frontend / UI / UX Developer(s)

A type of Developer. Develops the user interface and user experience of the system.

2.2.4 Server / Middle Tier Developer(s)

A type of Developer. Develops the business logic of the system.

2.2.5 Backend / Database Developer(s)

A type of Developer. Develops the database of the system.

2.2.6 Consultant(s) / Advisor(s) / Mentor(s)

Consultants to the developers include our Professor, Dr. Birgit Penzenstadler, and our project visionary and client, Christopher Priebe from G2SS. These stakeholders empower the team to work effectively, provide guidance to the team whether that be technical or involving system goals, provide project requirements, set deadlines, influence greater cooperation among team members, look over the team's work, and provide feedback.

2.2.7 Classified DoD Agency

Department of Defense agencies are coordinated and supervised by the US Department of Defense. They are established for specific purposes concerning national security and the United States Armed Forces. These stakeholders will be able to browse and download KML files from our system via our API on a classified network. With these files, they will be able to provide support to civilian authorities by engaging some of their own mission systems.

2.2.8 Mission Partners

Any agency, DoD or non-DoD, that has a role to play in supporting any authority, civilian or military, during any sort of mission. Mission partners can include National Guard Units, the Federal Emergency Management Agency (FEMA), and state or local governments (represented as “Civilian Authorities” and “Emergency Services” in our diagram). Mission partners upload KML files to our system via our WebUI.

2.2.9 Civilian Authorities

A type of mission partner. Any non-DoD local, state, or federal government that is in charge during an event or crisis. They can function as a mission partner during an event and upload files to our system. Classified DoD Agencies provide support to these stakeholders.

2.2.10 Emergency Services

A type of mission partner. They contribute to the overall well-being of civilians. They provide and report emergency information. They can function as a mission partner during an event and upload files to our system.

2.3 Stakeholder Matrix

Stakeholder Name	Motivations	Responsibilities/Functions	Skills	Priority
G2SS (Client)	1. To have a file management web system implemented by the team of developers. 2. To have this web system support a cross-domain file sharing requirement. 3. To excel at delivering superior software development solutions.	1. Provide the developer team with system requirements/features and overall system vision. 2. Make decisions on feature prioritization. 3. Act as a consultant, advisor, and mentor for the team. 4. Create and maintain user accounts on the system.	G2 excels at delivering superior software development solutions.	High
Web frontend/ UX/ UI developer(s)	1. Personal skill development. 2. Contribute to implementing a sophisticated system. 3. Relationships and networking.	1. Participate in team meetings. 2. Take notes and communicate information with the team. 3. Provide support. 4. Participate in documentation. 5. Complete assigned tasks by assigned deadlines. 6. Implement WebUI.	CSS, HTML5, JavaScript, JavaScript libraries, Angular, Trello, Draw.io or other diagramming tool, G (Google) Suite	High
Server/ Middle Tier Developer(s)	1. Personal skill development. 2. Contribute to implementing a sophisticated system. 3. Relationships and networking.	1. Participate in team meetings. 2. Take notes and communicate information with the team. 3. Provide support. 4. Participate in documentation. 5. Implement the business logic of the system. 6. Complete assigned tasks by assigned deadlines.	Node.JS, JavaScript, Trello, Draw.io or other diagramming tool, G (Google) Suite	High
Backend/ Database Developer(s)	1. Personal skill development. 2. Contribute to implementing a sophisticated system. 3. Relationships and networking.	1. Participate in team meetings. 2. Take notes and communicate information with the team. 3. Provide support. 4. Participate in documentation. 5. Complete assigned tasks by assigned deadlines. 6. Implement and maintain database.	MongoDB - NoSQL Database, Trello, Draw.io or other diagramming tool, G (Google)	High

			Suite	
Consultant(s) / Advisor(s) / Mentor(s)	1. Keep the project on task. 2. Keep the system goal clear. 3. Ensure that students are learning software engineering principles and processes.	1. Empower the team to work effectively. 2. Provide guidance to the team whether that be technical or involving system goals. 3. Provide project requirements. 4. Set deadlines. 5. Influence greater cooperation among team members. 6. Look over team's work and provide feedback.	Knowledge of software engineering principles.	Medium
Classified DoD Agency	1. Retrieve GIS files from an unclassified network via a guard to a classified one in order to engage their own mission systems to provide machinery, manpower, or other resources in support of civilian authorities.	1. Use our web service API to get access to our files and metadata.	Ability to implement an API in their software.	Medium
Mission Partners	1. Provide files that the DoD needs to engage their resources at the classified level.	1. Upload amplifying KML files for retrieval by DoD agencies.	Ability to upload files in a valid format and fill out forms.	Medium
Civilian Authorities	1. Receive support from DoD agencies during an event or crisis.			

3. Goal Model

3.1 Stakeholders and their Goals

This is a listing of some of our stakeholders' business, usage, and system goals. These goals are reflected in our Goal Diagram. Main usage goals are further detailed in our Usage Model.

3.1.1 G2SS (Client)

1. <Business Goal> Provide and maintain a file management system to support a cross-domain file sharing requirement.
2. <Business Goal> Give system access to authenticated users.
3. <Business Goal> Provide access to validated GIS files from authorized users.
4. <Business Goal> Provide GIS files of operational specific zones affected by an emergency to later inform other external agencies.
5. <Usage Goal> Login. *(See Use Case Scenario 7 for further detail).*
6. <Usage Goal> Create and maintain user accounts. *(See Use Case Scenario 8 for further detail).*
7. <Usage Goal> Browse files. *(See Use Case Scenario 3 for further detail).*
8. <Usage Goal> Delete KML files. *(See Use Case Scenario 6 for further detail).*
9. <Usage Goal> System is easy to use.
10. <Usage Goal> View metadata of files. *(See Use Case Scenario 4 for further detail).*
11. <Usage Goal> Sort files. *(See Use Case Scenario 5 for further detail).*
12. <System Goal> [technical] Provide website management/maintenance/support.

3.1.2 Web frontend/UX/UI developer(s)

1. <System Goal> [technical] Implement a simple, easy to use WebUI and WebUX while conforming to the overall vision of the system.
2. <System Goal> [technical] Provide interface to upload KML file.
3. <System Goal> [technical] Provide interface to collect and store required metadata about file (who, what/why, when, etc.).
4. <System Goal> [technical] Provide interface to allow user to see status of previously uploaded files. Interface includes table view showing files and status, with 'Delete button'.
5. <System Goal> [technical] Provide username/password login form.
6. <System Goal> [technical] Provide website management/maintenance/support.
7. <System Goal> [technical] Limit amount of loading time between app screens by reducing bottlenecks down the stack that cause blocking.

8. <System Goal> [technical] Have an upper bound run-time on each transaction.

3.1.3 Server/Middle Tier Developer(s)

1. <System Goal> [technical] “Validate” uploaded files, recording validation status for each one. (KML validation (XML schema validation) - incorporate a validating XML parser.)
2. <System Goal> [technical] Enforce a max file size.
3. <System Goal> [technical] Expose clean REST web APIs for basic fetching by 3rd party application. REST API to query for list of available KML files (and their statuses), REST API to pull full KML file, and REST API to delete KML file (retain stored metadata, adding new metadata about date/time/user for file deletion operation) - same functionality as the ‘Delete’ button from the UI.
4. <System Goal> [technical] Implement username/password validation. Possibly hash passwords.
5. <System Goal> [technical] Provide website management/maintenance/support.
6. <System Goal> [technical] Optimize website content.
7. <System Goal> [technical] Have an upper bound run-time on each transaction.

3.1.4 Backend/DB Developer(s)

1. <System Goal> [technical] Store KML file along with required metadata about the file (who, what/why, when, etc.).
2. <System Goal> [technical] Record validation status for each file.
3. <System Goal> [technical] Handle file deletion which updates the stored metadata about the file.
4. <System Goal> [technical] Provide website management/maintenance/support.
5. <System Goal> [technical] Optimize website content: Strive to minimize bandwidth waste. Embrace asynchronous retrieval of data. Parse the KML files within linear time.
6. <System Goal> [technical] Have an upper bound run-time on each transaction.

3.1.5 Classified DoD Agencies

1. <Business Goal> Use of web service APIs to automate the retrieval of GIS files in order to provide support to civilian authorities.
2. <Usage Goal> Access files via our web service API on a classified network. *(See Use Case Scenario 9 for further detail).*
3. <Usage Goal> Browse files via our web service API on a classified network.
4. <Usage Goal> Download GIS files via our web service API on a classified network. *(See Use*

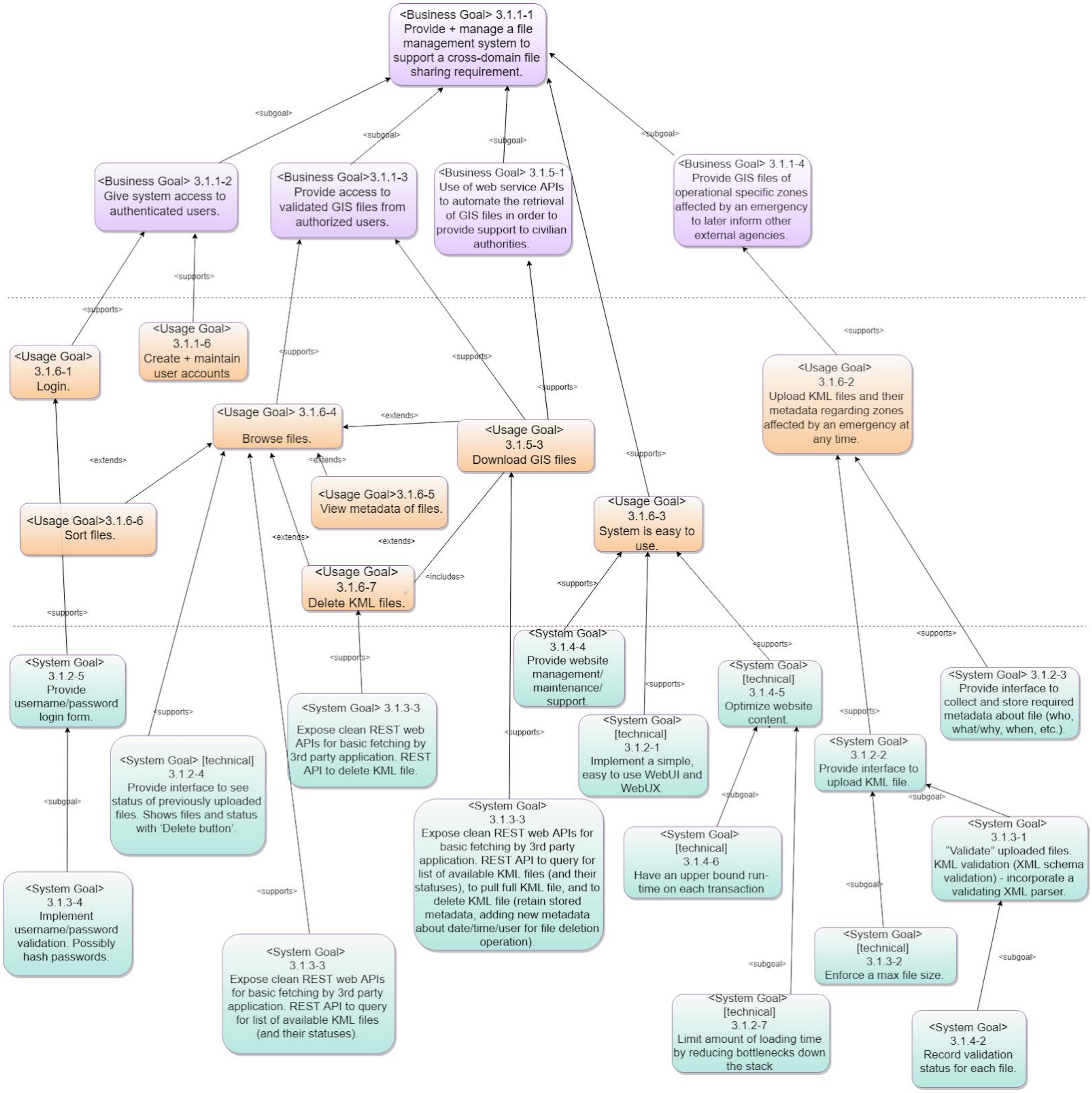
Case Scenario 2 for further detail).

5. <Usage Goal> Delete KML files via our web service API on a classified network.
6. <Usage Goal> System is easy to use.

3.1.6 Mission Partners

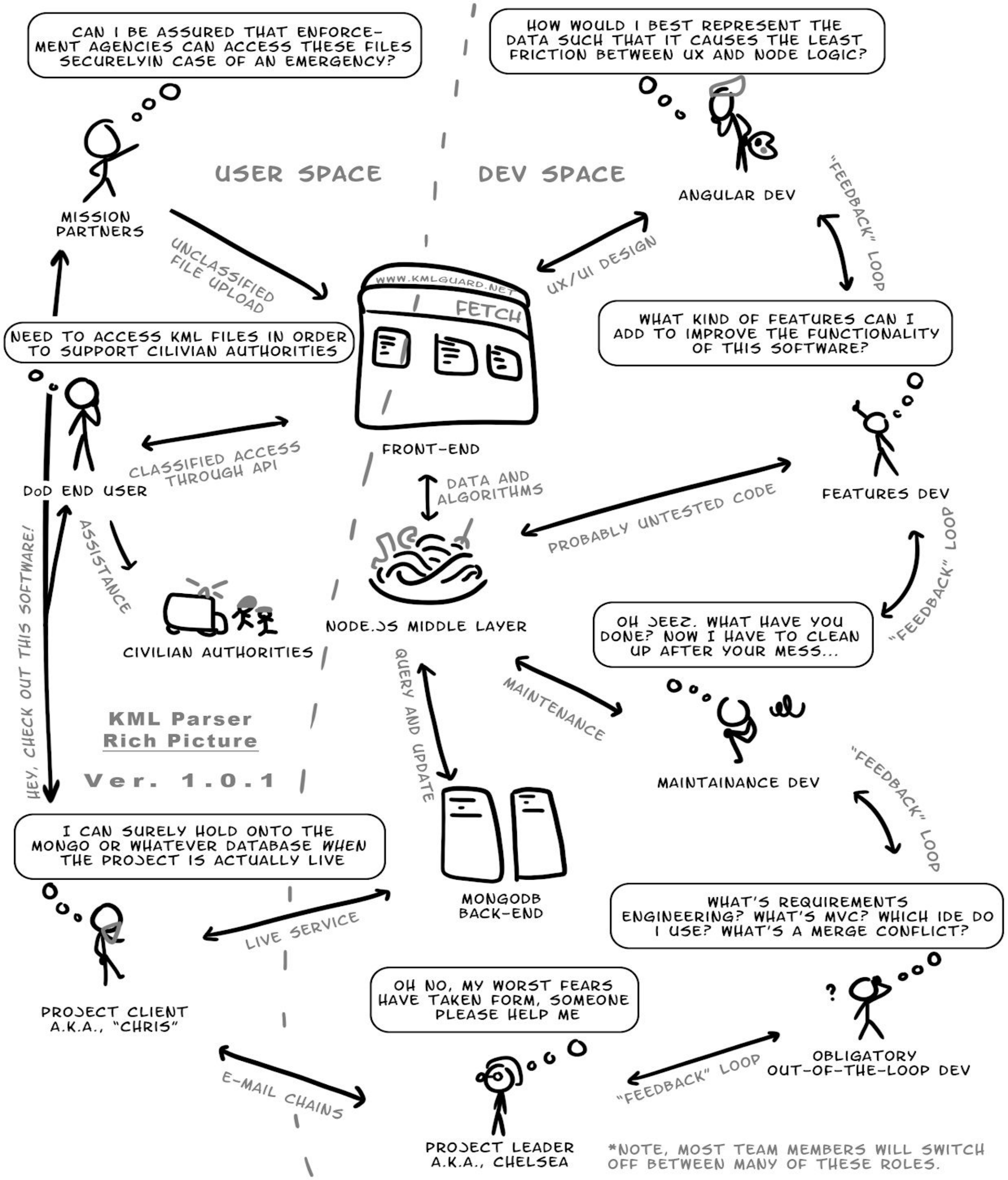
1. <Usage Goal> Login. *(See Use Case Scenario 7 for further detail).*
2. <Usage Goal> Upload KML files and their metadata regarding zones affected by an emergency at any time. *(See Use Case Scenario 1 for further detail).*
3. <Usage Goal> System is easy to use.
4. <Usage Goal> Browse files. *(See Use Case Scenario 3 for further detail).*
5. <Usage Goal> View metadata of files. *(See Use Case Scenario 4 for further detail).*
6. <Usage Goal> Sort files. *(See Use Case Scenario 5 for further detail).*
7. <Usage Goal> Delete KML files. *(See Use Case Scenario 6 for further detail).*

3.2 Goal Diagram


<Business Goal> 3.1.1-1 Provide + manage a file management system to support a cross-domain file sharing requirement.
<subgoal>
<subgoal>
<subgoal>
<subgoal>
<Business Goal> 3.1.1-2 Give system access to authenticated users.
<Business Goal>3.1.1-3 Provide access to validated GIS files from authorized users.
<Business Goal> 3.1.5-1 Use of web service APIs to automate the retrieval of GIS files in order to provide support to civilian authorities.
<Business Goal> 3.1.1-4 Provide GIS files of operational specific zones affected by an emergency to later inform other external agencies.
<supports>
<supports>
<Usage Goal> 3.1.6-1 Login.
<Usage Goal> 3.1.1-6 Create + maintain user accounts
<supports>
<supports>
<supports>
<supports>
<Usage Goal> 3.1.6-2 Upload KML files and their metadata regarding zones affected by an emergency at any time.
<Usage Goal> 3.1.6-4 Browse files.
<extends>
<Usage Goal> 3.1.5-3 Download GIS files
<extends>
<extends>
<supports>
<Usage Goal>3.1.6-6 Sort files.
<Usage Goal>3.1.6-5 View metadata of files.
<Usage Goal> 3.1.6-3 System is easy to use.
<extends>
<extends>
<Usage Goal> 3.1.6-7 Delete KML files.
<includes>
<supports>
<supports>
<supports>
<System Goal> 3.1.4-4 Provide website management/ maintenance/ support.
<supports>
<supports>
<System Goal> 3.1.2-5 Provide username/password login form.
<System Goal> 3.1.3-3 Expose clean REST web APIs for basic fetching by 3rd party application. REST API to delete KML file.
<supports>
<System Goal> [technical] 3.1.4-5 Optimize website content.
<System Goal> 3.1.2-3 Provide interface to collect and store required metadata about file (who, what/why, when, etc.).
<supports>
<System Goal> [technical] 3.1.2-1 Implement a simple, easy to use WebUI and WebUX.
<supports>
<System Goal> 3.1.2-2 Provide interface to upload KML file.
<System Goal> [technical] 3.1.2-4 Provide interface to see status of previously uploaded files. Shows files and status with 'Delete button'.
<supports>
<subgoal>
<subgoal>
<System Goal> 3.1.3-3 Expose clean REST web APIs for basic fetching by 3rd party application. REST API to query for list of available KML files (and their statuses), to pull full KML file, and to delete KML file (retain stored metadata, adding new metadata about date/time/user for file deletion operation).
<System Goal> [technical] 3.1.4-6 Have an upper bound run-time on each transaction
<System Goal> 3.1.3-1 "Validate" uploaded files. KML validation (XML schema validation) - incorporate a validating XML parser.
<subgoal>
<System Goal> 3.1.3-4 Implement username/password validation. Possibly hash passwords.
<supports>
<subgoal>
<System Goal> [technical] 3.1.3-2 Enforce a max file size.
<subgoal>
<System Goal> 3.1.3-3 Expose clean REST web APIs for basic fetching by 3rd party application. REST API to query for list of available KML files (and their statuses).
<subgoal>
<System Goal> [technical] 3.1.2-7 Limit amount of loading time by reducing bottlenecks down the stack
<System Goal> 3.1.4-2 Record validation status for each file.

Based on what is needed from our stakeholders, we have set up different goals and divided them into three different categories consisting of Business, Usage, and System. The way we have organized it, the top goal is of highest priority with the following goals stemming from it. Section numbers from [3.1 Stakeholders and their Goals](#) are denoted in each bubble.

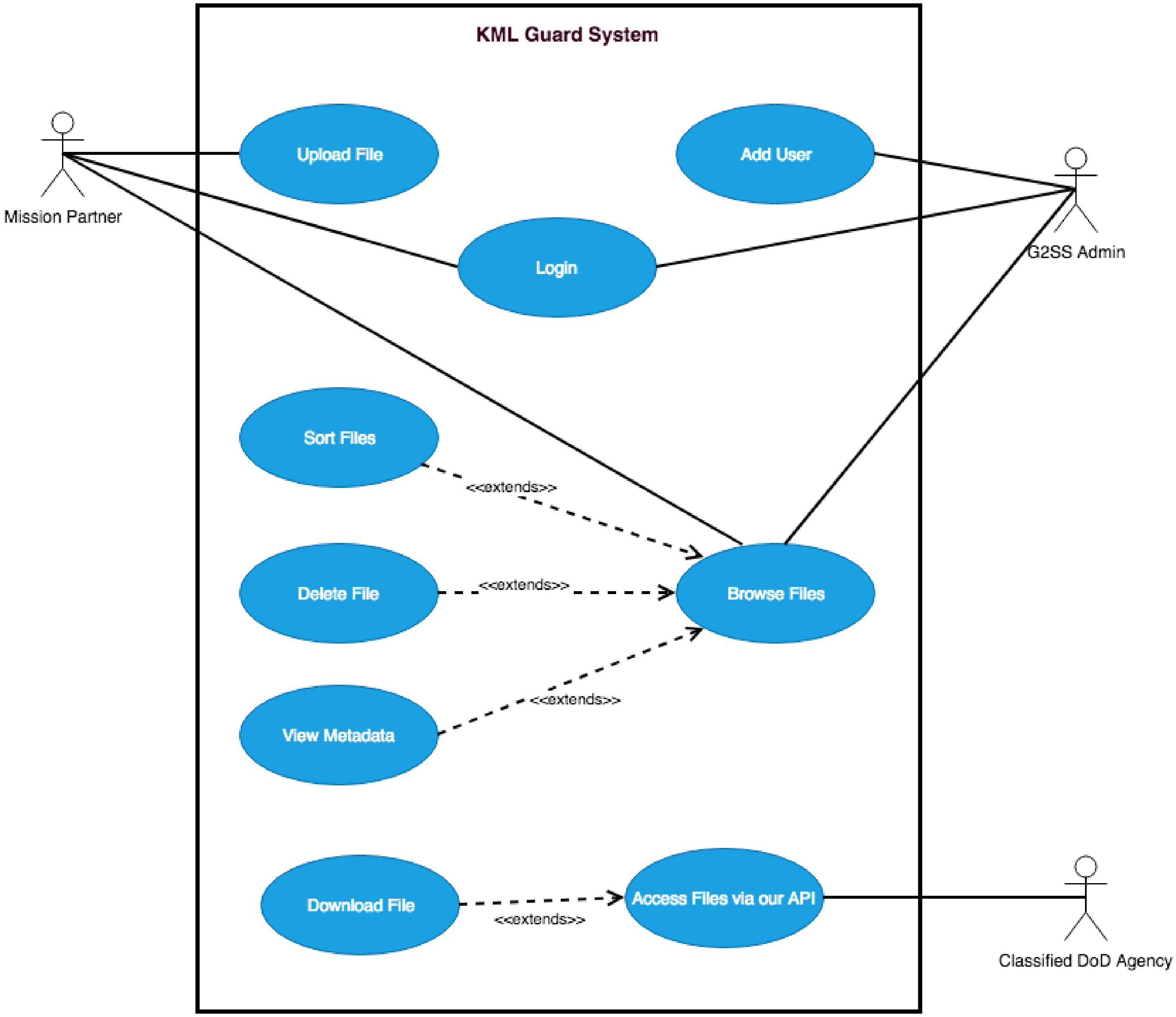
4. System Vision - Rich Picture


CAN I BE ASSURED THAT ENFORCE- MENT AGENCIES CAN ACCESS THESE FILES SECURELYIN CASE OF AN EMERGENCY?
HOW WOULD I BEST REPRESENT THE DATA SUCH THAT IT CAUSES THE LEAST FRICTION BETWEEN UX AND NODE LOGIC?
USER SPACE
DEV SPACE
MISSION PARTNERS
ANGULAR DEV
"FEEDBACK" LOOP
UNCLASSIFIED FILE UPLOAD
UX/UI DESIGN
WWW.KMLGUARD.NET
FETCH
WHAT KIND OF FEATURES CAN I ADD TO IMPROVE THE FUNCTIONALITY OF THIS SOFTWARE?
NEED TO ACCESS KML FILES IN ORDER TO SUPPORT CILIVIAN AUTHORITIES
CLASSIFIED ACCESS THROUGH API
FRONT-END
DoD END USER
DATA AND ALGORITHMS
FEATURES DEV
PROBABLY UNTESTED CODE
"FEEDBACK" LOOP
HEY, CHECK OUT THIS SOFTWARE!
ASSISTANCE
CIVILIAN AUTHORITIES
NODE.JS MIDDLE LAYER
OH JEEZ. WHAT HAVE YOU DONE? NOW I HAVE TO CLEAN UP AFTER YOUR MESS...
QUERY AND UPDATE
MAINTENANCE
KML Parser
Rich Picture
Ver. 1.0.1
MAINTAINANCE DEV
"FEEDBACK" LOOP
I CAN SURELY HOLD ONTO THE MONGO OR WHATEVER DATABASE WHEN THE PROJECT IS ACTUALLY LIVE
MONGODB BACK-END
WHAT'S REQUIREMENTS ENGINEERING? WHAT'S MVC? WHICH IDE DO I USE? WHAT'S A MERGE CONFLICT?
LIVE SERVICE
PROJECT CLIENT A.K.A., "CHRIS"
OH NO, MY WORST FEARS HAVE TAKEN FORM, SOMEONE PLEASE HELP ME
?
OBLIGATORY OUT-OF-THE-LOOP DEV
E-MAIL CHAINS
"FEEDBACK" LOOP
PROJECT LEADER A.K.A., CHELSEA
*NOTE, MOST TEAM MEMBERS WILL SWITCH OFF BETWEEN MANY OF THESE ROLES.

This picture represents the flow of concerns and actions taken by both developers and stakeholders as they act upon the system, whose components are identified and interact with each other.

5. Usage Model: Use Cases

5.1 Use Case Diagram



This diagram provides an overview of KML Guard's use cases and their actors. Use case scenarios for each bubble are detailed in the next section.

5.2 Use Case Scenarios

5.2.1 Use Case Scenario 1: Upload File via Our WebUI

USE CASE # 1	Upload File via Our WebUI	
Goal in Context	To inform DoD agencies of a local emergency or to provide DoD agencies with valuable information.	
Scope	File Sharing System	
Level	Primary Task	
Preconditions	User is logged in to the web application.	
Success End Condition	File is uploaded to the server and can be downloaded by other users.	
Failed End Condition	File is not uploaded to the server.	
Primary Actor	Mission Partners	
Secondary Actor	Emergency service	
Trigger	User clicks on upload link.	
DESCRIPTION	Step	Action
	1	User clicks on upload link.
	2	Application displays basic file upload form with simple meta-data table.
	3	User fills in the upload form.
	4	User clicks on upload button and selects a KML file to upload.
	5	User clicks on complete link.
	6	System validates KML file.
	7	System stores KML file and updates the application.
SUB-VARIATIONS	Step	Branching Action
	5	Condition: user enters invalid information. Action: application displays error message.

	6	Condition: KML file fails validation. Action: application notifies user.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.6-2).
Priority		Top priority
Performance		Three minutes
Frequency		Often
Channels to actors		Interactive
OPEN ISSUES		How to handle files that fail validation.
Due Date		May 2019

5.2.2 Use Case Scenario 2: Download File via Our API

USE CASE # 2	Download File via Our API	
Goal in Context	To obtain informative KML files.	
Scope	File Sharing System	
Level	Primary Task	
Preconditions	User’s 3rd party application invokes our API exposed by our Node.js server. User has embedded their Username/Password inside the HTTP headers of the HTTP "Get" request they initiate.	
Success End Condition	File is downloaded.	
Failed End Condition	File is not downloaded.	
Primary Actor	DoD agency	
Trigger	User invokes HTTP GET request to the server.	
DESCRIPTION	Step	Action
	1	User invokes HTTP GET request to the server.
	2	Server returns a list of available files.
	3	User does an HTTP GET request to get the KML file.

	4	System transfers the file to the user’s computer and updates meta-data with new download entry.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.5-3)
Priority		Top priority
Performance		Linear with the size of the file.
Frequency		Often
Channels to actors		Interactive
Due Date		May 2019

5.2.3 Use Case Scenario 3: Browse Files via our WebUI

USE CASE # 3	Browse Files via Our WebUI	
Goal in Context	To view the main table of information about the files in the system.	
Scope	File Sharing System	
Level	Primary Task	
Preconditions	User is logged in to the web application.	
Success End Condition	User sees table view showing files and status.	
Failed End Condition	User does not see table view showing files and status.	
Primary Actor	G2SS, Mission Partners	
Trigger	User successfully logs in to the application.	
DESCRIPTION	Step	Action
	1	User logs in to the application.
	2	Application displays table view showing files and their status.
	3	User scrolls through the table.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-6, 3.1.6-4)

Priority		Top priority
Performance		One minute (depending on the user)
Frequency		Very Often
Channels to actors		Interactive
Due Date		May 2019

5.2.4 Use Case Scenario 4: View Metadata via our WebUI

USE CASE # 4	View Metadata via Our WebUI	
Goal in Context	To view details/metadata of an uploaded file.	
Scope	File Sharing System	
Level	Subfunction	
Preconditions	User is logged in to the web application.	
Success End Condition	User sees details of a chosen file.	
Failed End Condition	User does not see details of a chosen file.	
Primary Actor	G2SS, Mission Partners	
Trigger	User clicks on a row in the file table.	
DESCRIPTION	Step	Action
	1	User clicks on a row in the file table.
	2	Application displays details of the file.
	3	User clicks on the row again.
	4	Application returns to the file table.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-10, 3.1.6-5)
Priority		Medium priority
Performance		One second
Frequency		Very Often

Channels to actors		Interactive
Due Date		May 2019

5.2.5 Use Case Scenario 5: Sort Files via our WebUI

USE CASE # 5	Sort Files via Our WebUI	
Goal in Context	To re-sort data table by column header.	
Scope	File Sharing System	
Level	Subfunction	
Preconditions	User is logged in to the web application.	
Success End Condition	Data table is sorted by the column.	
Failed End Condition	Data table is not sorted by the column.	
Primary Actor	G2SS, Mission Partners	
Trigger	User clicks on a column header.	
DESCRIPTION	Step	Action
	1	User clicks on a column header.
	2	Application redisplays the data table sorted by that column header.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-11, 3.1.6-6)
Priority		Low priority
Performance		Two Seconds
Frequency		Very Often
Channels to actors		Interactive
Due Date		May 2019

5.2.6 Use Case Scenario 6: Delete File via our WebUI

USE CASE # 6	Delete File via Our WebUI	
Goal in Context	To delete a file in the server.	
Scope	File Sharing System	
Level	Subfunction	
Preconditions	User is logged in to the web application.	
Success End Condition	User successfully removes a file from the server.	
Failed End Condition	User cannot remove a file from the server.	
Primary Actor	G2SS, Mission Partner	
Trigger	User clicks the delete file button.	
DESCRIPTION	Step	Action
	1	User clicks the delete file button.
	2	Application prompts user for confirmation.
	3	User clicks yes.
	4	System deletes the file from the server and updates the file's metadata with deletion details.
	5	Application notifies the user that the file was successfully deleted.
EXTENSIONS	Step	Branching Action
	3a	Condition: User clicks no. Action: Application returns to step 1.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-8, 3.1.6-7).
Priority		Medium priority
Performance		Five Seconds
Frequency		Often

Channels to actors		Interactive
Due Date		May 2019

5.2.7 Use Case Scenario 7: Login via our WebUI

USE CASE # 7	Login via Our WebUI	
Goal in Context	To give system access to authenticated users.	
Scope	User Management	
Level	Subfunction	
Preconditions	User is a DoD agency with valid username and password and has access to the web application.	
Success End Condition	User successfully logs in.	
Failed End Condition	User cannot login.	
Primary Actor(s)	G2SS, Mission Partners	
Trigger	User goes to the URL.	
DESCRIPTION	Step	Action
	1	User goes to the URL.
	2	Application displays a login page.
	3	User types in a valid username and password.
	4	Application displays the the main table showing files and a separate widget to allow the upload of new files.
EXTENSIONS	Step	Branching Action
	3a	Condition: user clicks on forgot username/password link. Action: application displays the password recovery page. *Stretch goal.*
SUB-VARIATIONS	Step	Branching Action
	4	Condition: user enters invalid username or password.

		Action: application displays error message and prompts user to re-enter username and password.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-5, 3.1.6-1)
Priority		Top priority
Performance		Ten Second
Frequency		Very Often
Channels to actors		Interactive
Due Date		May 2019

5.2.8 Use Case Scenario 8: Add User via our WebUI

USE CASE # 8	Add User via Our WebUI	
Goal in Context	To allow G2SS administrators to add new users to the application.	
Scope	User Management	
Level	Subfunction	
Preconditions	User is a G2SS administrator and is logged in to the web application.	
Success End Condition	Administrator successfully adds a new user.	
Failed End Condition	Administrator cannot add a new user.	
Primary Actor	G2SS Administrator	
Trigger	Admin clicks the add new user button.	
DESCRIPTION	Step	Action
	1	Admin clicks the add new user button.
	2	Application displays a web form.
	3	Admin fills out the new user form and clicks submit.
	4	System add the new user to the database.

	5	Application returns to homepage.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.1-5)
Priority		Medium priority
Performance		One Minute
Frequency		Often
Channels to actors		Interactive
Due Date		May 2019

5.2.9 Use Case Scenario 9: Access Files via our API

USE CASE # 9	Access Files via our API	
Goal in Context	To obtain access to the files and corresponding metadata in our database.	
Scope	File Sharing System	
Level	Primary Task	
Preconditions	User's 3rd party application invokes our API exposed by our server. User has embedded their Username/Password inside the HTTP headers of the HTTP "Get" request they initiate.	
Success End Condition	User is able to access the data and files in our database.	
Failed End Condition	User is not able to access the data and files in our database.	
Primary Actor	DoD Agencies	
Trigger	User invokes our API on software running in their application.	
DESCRIPTION	Step	Action
	1	User invokes our API on software running in their application.
	2	User makes HTTP requests on our data.
RELATED INFORMATION		Featured as a usage goal in the Goal Model (3.1.5-2)
Priority		Top priority

Performance		One Minute
Frequency		Often
Channels to actors		Interactive
Due Date		May 2019

6. Detailed Requirements

6.1 Process Non-Functional Requirements

Process requirements are required characteristics of the process used to carry out this project. They deal with things like deadlines and deliverables.

6.1.1 Use an iterative software development process.

Process NFR	**Use an iterative software development process.**
Rationale	Software development is not a linear process. We will constantly need to make revisions or revisit requirements and features.
Satisfaction Criterion	All features and requirements can be revisited and fully revised or changed if needed.
Measurement	We will count how many times changes need to be made along the process, and how many of those changes were successfully made.
Risk	If we don't use an iterative development process, we will be inflexible to making changes when problems or conflicts arise.

6.1.2 Learn the technologies needed to implement the project.

Process NFR	**Learn the technologies needed to implement the project.**
Rationale	We'll only be able to implement the system if we know how to use the technologies.
Satisfaction Criterion	100% of the features are fully functional with the proposed technologies by the due date.
Measurement	Apply what we learn by implementing 1 or 2 system features per week.
Risk	The system won't be fully functional.

6.1.3 Complete by due date.

Process NFR	**Complete by due date.**

Rationale	Stakeholders will not be happy if the system is incomplete by the date they expect it to be complete.
Satisfaction Criterion	All use cases and test cases are implemented by the due date.
Measurement	Testers are able to walk through each use case with only 1 or 2 complications.
Risk	Stakeholders will doubt our ability to implement this system.

6.2 Deployment Non-Functional Requirements

Deployment requirements are the demands for deployment of the project.

6.2.1 Deploy a web application that runs as a Node.JS package.

Deployment NFR	**Deploy a web application that runs as a Node.JS package**
Rationale	Supply the files needed to run the web application (Portability)
Satisfaction Criterion	Systems runs successfully to end user’s browser.
Measurement	Web application is able to run and show the contents using a Node.JS environment.
Risk	No functionality will be presented to the user.

6.2.2 Users with minimum training shall be able to use the system

Deployment NFR	**Users with minimum training shall be able to use the system**
Rationale	Ensure system is user-friendly and requires minimum training to be able to operate the web application.
Satisfaction Criterion	Inexperienced users successfully operate the web application.
Measurement	Feedback from users will be logged to be later analyzed

Risk	No functionality will be presented to the user.

6.2.3 Modified data in the database should be able to update to all users

Deployment NFR	**Modified data in the database should be able to update to all users**
Rationale	Facilitate integrity and concurrent processing of data and retrieval (e.g., KML files, metadata)
Satisfaction Criterion	Data is able to be stored, updated and retrieved for use by stakeholders and in displaying files specific to a stakeholder’s account.
Measurement	After successful login, a stakeholder is able to store KML files, preview their metadata, update and retrieve them from the database. The required actions are executed at the backend.
Risk	Stakeholders from unclassified systems cannot count on their data being available to classified systems.

6.3 System Constraints Non-Functional Requirements

System constraints are system-related restrictions that don’t necessarily result from functional goals. These could include operating systems and languages needed for the project.

6.3.1 System usability in different machines

System Constraint NFR	**System usability in different machines.**
Rationale	The web application must be able to function as a Node.JS web application regardless of the machine used.
Satisfaction Criterion	A stakeholder is able to unzip the application and launch the Node.JS package in their machine
Measurement	Detection of performance and utility to measure the actual operation (including the amount of time it to execute processes).
Risk	Any execution of the web application in a non-Node.JS environment imposes a security risk on the entire system.

6.3.2 Require user login.

System Constraint NFR	Require user login.
Rationale	Stakeholders should be able to manage only their own account with its own data.
Satisfaction Criterion	The stakeholder will be logged in, presenting them with the main application screen with data specific to their account.
Measurement	Record log of successful and failed login attempts.
Risk	Stakeholders may not trust application to keep their data safe.

6.3.3 Use Angular, Node.js, and Mongo.

System Constraint NFR	Use Angular, Node.js, Mongo.
Rationale	Provides modern, in-demand frameworks for creating applications on the web which broaden the developers’ skill sets.
Satisfaction Criterion	Continuous delivery of stellar functionality, security, and stability for the application.
Measurement	Github pull requests and waffle.io issue tracker resolutions.
Risk	Unfamiliarity with frameworks may slow down code output and quality.

6.3.4 The system must run on most browsers.

System Constraint NFR	The system must run on most browsers.
Rationale	Users will be able to access the system on any browser convenient to them.
Satisfaction Criterion	The web app is fully functional on Mozilla, Internet Explorer, Chrome, and Safari.
Measurement	Test that the web app is fully functional on Mozilla, Internet Explorer, Chrome, and Safari.
Risk	Users may not want to use the system if it does not run on their browser of choice.

6.3.5 Accessibility - Web application must follow accessibility guidelines.

System Constraint NFR	**Accessibility - Web application must follow accessibility guidelines.**
Rationale	Makes the web app accessible to all users.
Satisfaction Criterion	The web page satisfies W3C Accessibility Standards.
Measurement	Pass the web accessibility evaluation tool by WAVE.
Risk	If the web page does not follow accessibility guidelines, some users will not be able to use the web application.

6.4 Quality Non-Functional Requirements

Quality requirements are desired quality characteristics of the system.

6.4.1 Performance - Upload process should take less than one minute.

Quality NFR	**Performance - Upload process should take less than one minute.**
Rationale	Makes it more likely that users will be satisfied with the experience and will continue to use the system.
Satisfaction Criterion	80% of new users could complete the upload form within a minute.
Measurement	Have 20 new users fill out the upload form and track the time it takes for each user to complete the form.
Risk	If the upload process takes too long, users may reject the system.

6.4.2 Availability - The web application must be available at all times.

Quality NFR	**Availability - The web application must be available at all times.**
Rationale	Users will be frustrated if they cannot access the

	system when they need it. Users must be able to upload or retrieve files and their metadata at any time for their convenience.
Satisfaction Criterion	All users can access the web application at any time unless a maintenance is scheduled.
Measurement	Compare the server downtime with the agreed amount of time that the server should be available. The measured server downtime should be equal to the agreed amount of time that the server should be available.
Risk	Users will be frustrated that they cannot upload KML files during an emergency. Agencies will not be able to respond in time.

6.4.3 Usability - The system must be easy to use.

Quality NFR	**Usability - the system must be easy to use.**
Rationale	Makes it more likely that users will be satisfied with the experience and will continue to use the system.
Satisfaction Criterion	90% of users know how to navigate through and use all the system’s features after an initial introduction/description of the features.
Measurement	Have a group of people use every feature of the system. We will record how many people went through the whole system process without help. We will then calculate (# of testers who did not need help/total # of testers) to calculate the percentage.
Risk	If the system is not easy to use, users will not want to use the system.

6.4.4 Maintainability - Code must be clean and readable for maintainability.

Quality NFR	**Maintainability - Code must be clean and readable for maintainability.**
Rationale	Makes fixes to code easier to approach and understandable when different people work on it.
Satisfaction Criterion	Code readers/reviewers have no confusions when reading and working with our code.

Measurement	Code readers/reviewers only ask 1 or 2 questions when reading through our code.
Risk	Time wasted trying to understand, reverse engineer, or change the code.

6.4.5 Performance - Code must be functionally stable and easily recoverable from error.

Quality NFR	**Performance - Code must be functionally stable and easily-recoverable from error.**
Rationale	Improves user experience, reduces complications, and ensures there are no roadblocks.
Satisfaction Criterion	User experiences little to no errors during use
Measurement	Bug test for potential or known errors and work to resolve them
Risk	User is unable to proceed; External support will need to be called in to troubleshoot.