Software Design Documentation

BYU-Idaho Content Management System

Date: 11 June 2021

Written By: Spring 2021 Class of CS 364

Students: Jared Barney, Ian Blamires, Chris Brisco, Jonathan Dawson, Tyler DeFreitas, Josh Donaldson, David Doria, Weston Elsmore, Guilherme Faccinetto, Dylan Havens, Jack Leung, Hunter Livesay, Christian Longhurst, Tyler Lusk, Jarom Lybbert, Jessen Noble, Ulumasui Pe'a, Tanner Robinson, Michael Silski

Introduction (needs to be edited)

This document details the design specifications for a Content Management System to be deployed for Brigham Young University - Idaho (BYU-I). This Content Management System will be used by students, teachers, and external users to share and view content for school purposes under a variety of conditions.

System Overview (needs to be edited)

The Content Management Systems (CMS) will host content for users in one central location utilizing the following subsystems: Licensing, Features, Security, Syncing, Role-Based Access Control. The Licensing system will store metadata such as who created content, when content was created, and how often content is viewed among other information. The Features will handle compatibility between other platforms, servers, and media types as well as user interactions such as creating, editing, and uploading content among other operations. The Features will also include content organization and content notifications. The Security system will protect and backup content and sensitive information in compliance with Family Educational Rights and Privacy Act (FERPA) and Learning Tools Interoperability (LTI) standards. The Syncing feature will allow for the simultaneous upload of content while preventing race conditions. Role-Based Access control will give system administrators the ability to provide different role policies for various user types.

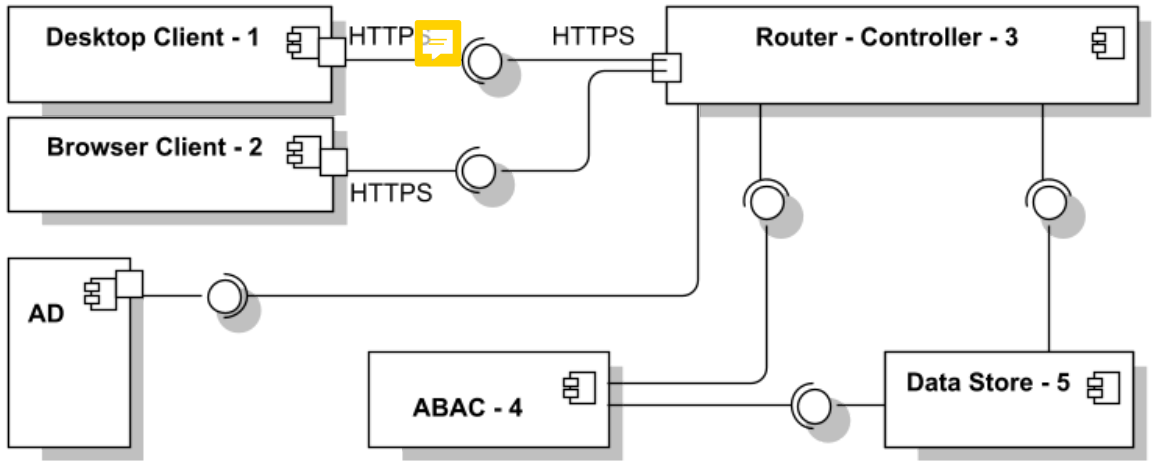
The CMS depends on several technologies that are not within the scope of the system. These technologies are the I-Learn Suite, web server, LTI, FERPA, and Microsoft Active Directory. The I-Learn Suite includes Canvas and the Authentication system among other University systems. A web server needs to be configured according to the requirements. Regardless of I-Learn, the CMS will still need to follow LTI principles. Microsoft Active Directory will assign a role to the user based on the role policies stated by BYU-I if the user is not already authenticated.

Viewpoints (needs to be filled out)

Component Diagram
Data Flow Diagram
Class Diagram
Flowchart
Entity Relationship Diagram

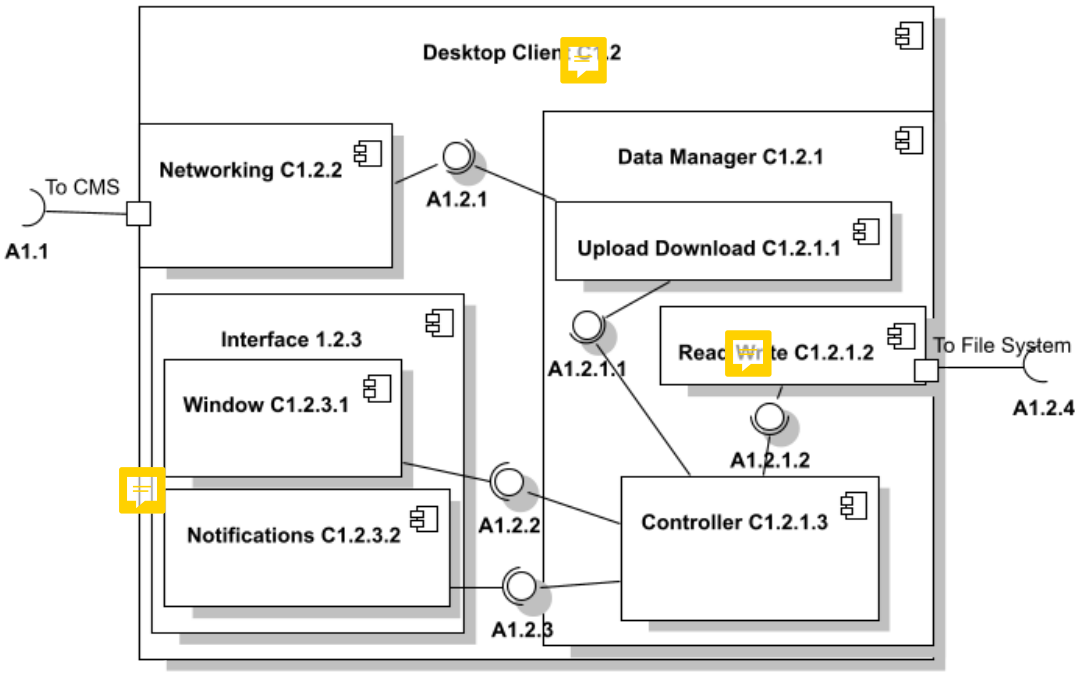
Design Views

System View



Name	System View	
Description	Details the highest level components that comprise or interact with the CMS.	
Design Concerns	Allows for users to store, manage, and share content through the CMS.	
Requirements	1.0 - 2.2 inclusive	
Elements	**Desktop Client** - 1 The user interface for desktops. Handles syncing between desktop and CMS.	**Attribute-Based Access Control (ABAC)** - 4 A system that controls user-access to content
	Browser Client - 2 The user interface for web browsers.	**Data Store** - 5 A repository for storing and managing collections of data.
	Router - Controller - 3 A function or group of functions that routes data to and from various components in the CMS.	**Microsoft Active Directory (AD)** The system that BYU-I uses for their Single Sign On(SSO). The CMS will need to interface with this for user-verification
Referenced By		
Viewpoint	UML 2.0 Component Diagram	

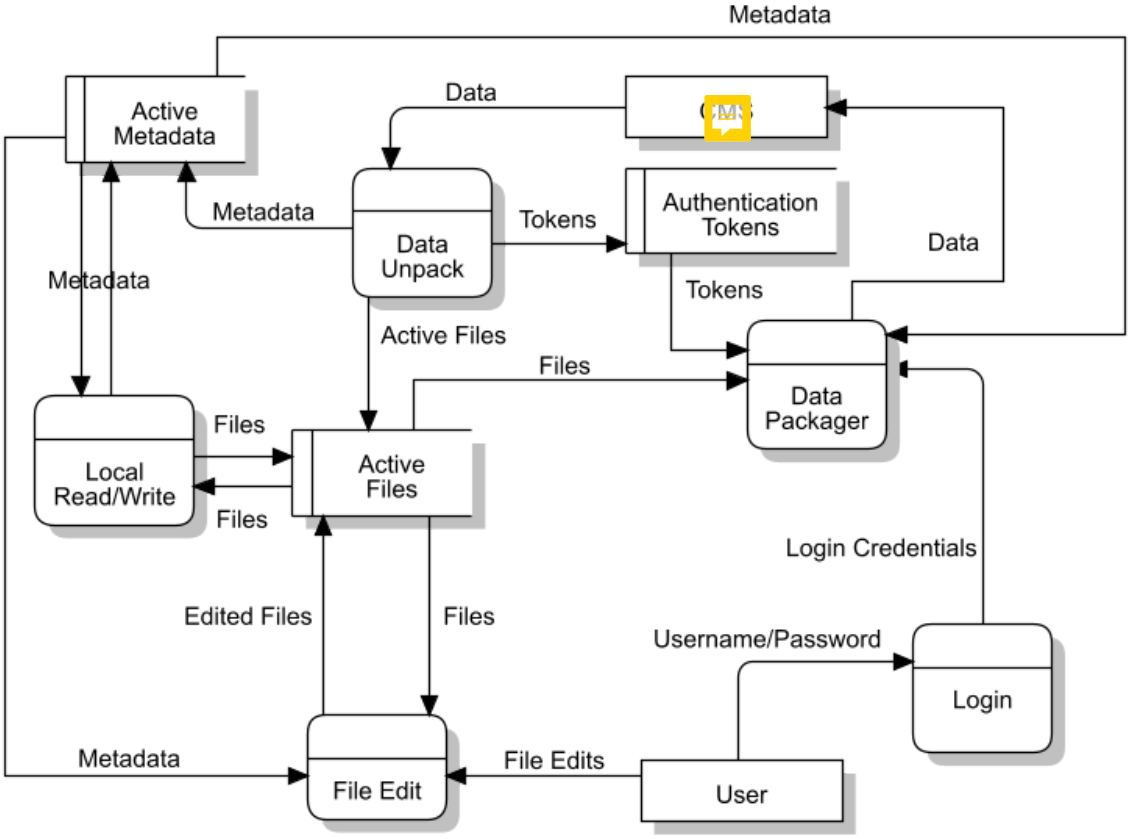
1 - Desktop Client Components



Name	View 1 - Desktop Client	
Description	Component diagram describing the structure of the CMS Desktop Client. This client runs on a personal machine and connects remotely to the main CMS server.	
Design Concerns	Allows users to create, edit, and save files to the CMS from a personal computer.	
Requirements	1.1.1, 1.1.2, 1.1.3, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 1.2.6, 1.2.7, 1.2.9, 1.2.11, 1.2.12, 1.2.13, 1.2.14, 1.2.15, 1.2.16, 1.2.17, 1.2.19, 1.3 Inclusive, 1.4.1, 1.4.4, 1.4.5, 1.4.6, 1.4.7, 1.4.8, 1.4.9, 2.1 - 2.2 Inclusive	
Elements	**Desktop Client** - C1.2 The user interface for desktops. Handles syncing between desktop and the CMS.	**Read Write** - C1.2.1.2 Deals with reading and writing files on the local disk.
	Data Manager - C1.2.1 Manages the general flow of data through the Desktop Client.	**Controller** - C1.2.1.3 Acts as the main hub for all data passing through the Desktop Client and determines where to send data given to it.
	Networking - C1.2.2 Handles sending and receiving data through the Internet.	**Window** - 1.2.3.1 Manages the GUI of the Desktop Client.

	Interface - C1.2.3 Handles the portions of the desktop client that interact directly with the user.	**Notifications** - C1.2.3.2 Handles the display of all notifications regarding the CMS.
	Upload Download - C1.2.1.1 Prepares and unpacks data for upload and download respectively.	
Referenced By	View	
Viewpoint	UML 2.0 Component Diagram	

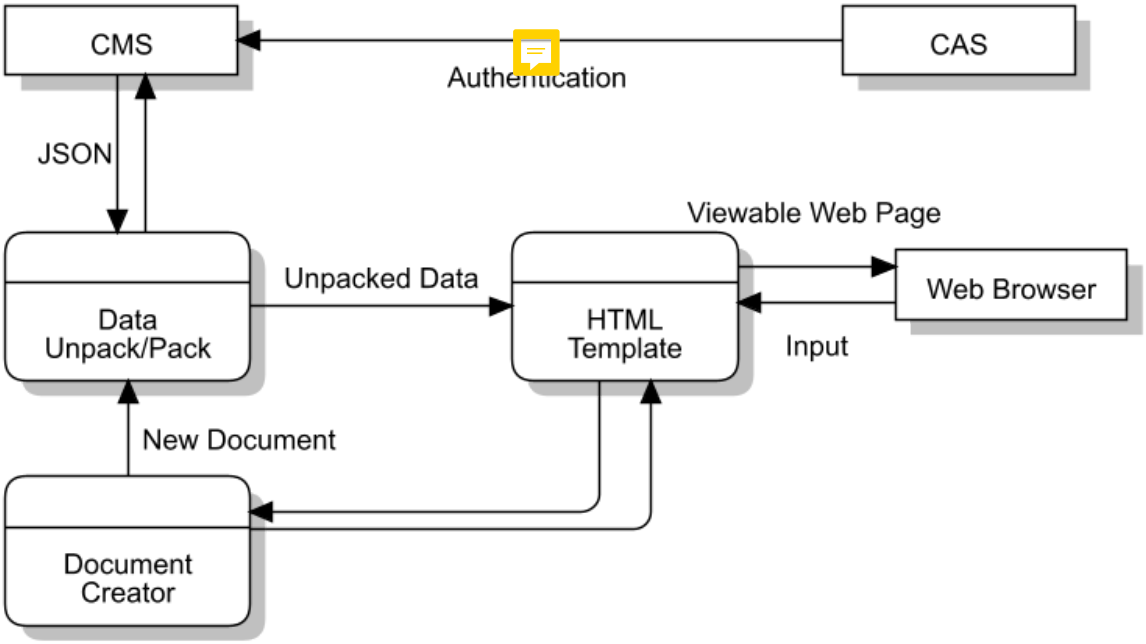
View 1.1 - Desktop Client



Name	View 1.1 - Desktop Client	
Description	A Data Flow Diagram showing the flow of data between the CMS and the Desktop Client.	
Design Concerns	Allows for users to store and manage data on their desktop by receiving data from the CMS.	
Requirements	1.1.1, 1.1.2, 1.1.3, 1.2.2, 1.2.3, 1.2.5, 1.2.9, 1.2.11, 1.2.12, 1.2.13, 1.2.19, 1.4.1, 1.4.4, 1.4.7, 1.4.8, 1.4.9, 1.5.3, 1.5.4, 1.5.11, 2.1 inclusive, 2.2.1, 2.2.3	
Elements	**Content Management System (CMS(**Link to View 1**)** - The system that handles how and	**Data Unpack** - A process that unpacks data and sends it to the appropriate and intended recipients.

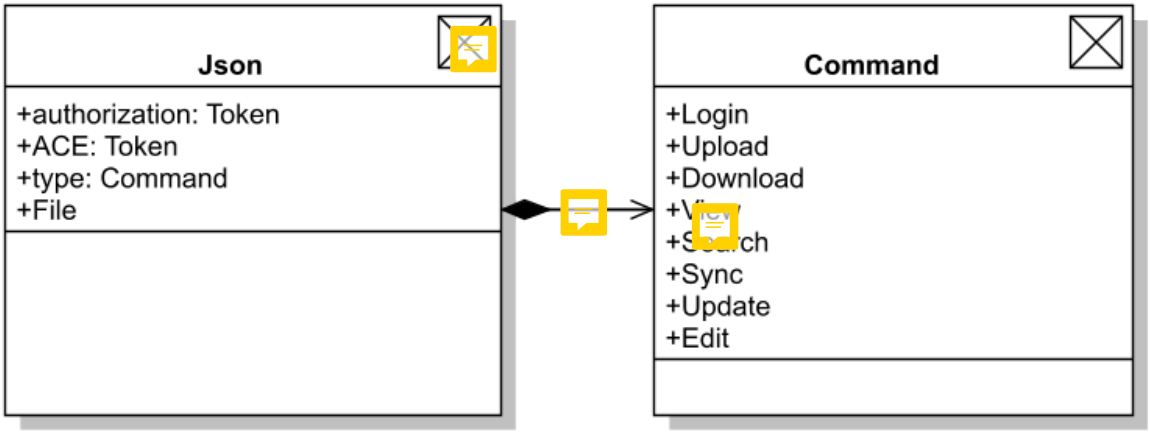
	where content is stored, viewed, uploaded, and edited.	
	Authentication Tokens - A data storage that stores tokens for authentication.	**Data Packager** - A process that packs data and sends it to the appropriate and intended recipients.
	Active Files - A data storage that receives files and stores them for the user to use.	**File Edit** - A process that will edit the files as the user sees fit.
	User - The actor that is using the product for its intended purpose.	**Login** - The process by which the user inputs their username and password to gain access to the CMS(Link to view 1).
	Local Read/Write - A process in which metadata or files can be read and/or written.	**Active Metadata** - A data storage that holds all metadata for the user to access.
Referenced By	View 1	
Viewpoint	UML 2.0 Data Flow Diagram	

2 - Browser Client DFD



Name	View 1 - Browser Client	
Description	Diagram displays the data flow between the web client and the CMS	
Design Concerns	Users need to interact with the system online	
Requirements	1.2.1, 1.2.3, 1.2.6, 1.2.7, 1.2.18 1.2.19	
Elements	**Content Management System (CMS)** The system that handles how and where content is stored, viewed, uploaded, and edited.	**CAS** Authentication for the CMS
	Data Packager Where the JSON gets unpacked and receives information from the document creator	**HTML Template** Where the unpacked data is organized from the data packager and document creator to the web browser
	Web Browser The client's interface online.	**Document Creator** A built-in document creation suite including a WYSIWYG editor
Referenced By	View 1	
Viewpoint	1.0 DF	

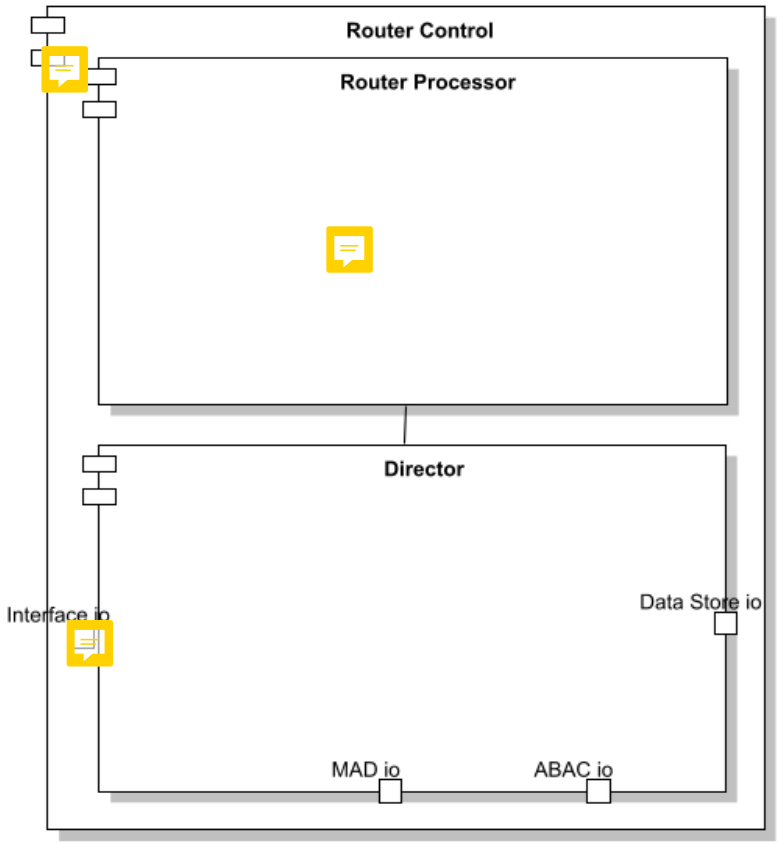
View 2.2 - JSON Schema



Name	View 2.1 - JSON Schema	
Description	Diagram detailing the information stored in JSON files. This method will be used between the controller (LINK TO VIEW #3) and the desktop(LINK TO VIEW) and web(LINK TO VIEW) clients.	
Design Concerns	Allows information transfer between components.	
Requirements	1.1.2, 1.1.3, 1.2.6, 1.4.1	
Elements	**JSON** The name and type of the file that will be passed between the interfaces and the controller (LINK TO VIEW #3)	**Authorization** A token that allows access to the CMS (LINK TO VIEW 1)through the active directory.
	ACE A token that defines the access level granted through the ABAC (LINK TO VIEW 4).	**Type** A command dictating the type of request being sent to the controller
	File The file that will be passed to or from the controller.	**Command** The types of commands that can be sent to the controller
	Login The command that sends a login request to the controller.	**Upload** The Command that dictates a certain document to be uploaded to the Data Store.
	Download A command that asks for a specific file to download.	**View** Asks for a link to view a document.
	Search A command that asks for a table of results to a query.	**Sync** A command for the desktop client to sync files to its directory inside the data store.
	Update A command that specifies updating the metadata of a file in the data store.	**Edit** A command asking for the ability to change certain elements of a document.
Referenced By	View 1, View 2	

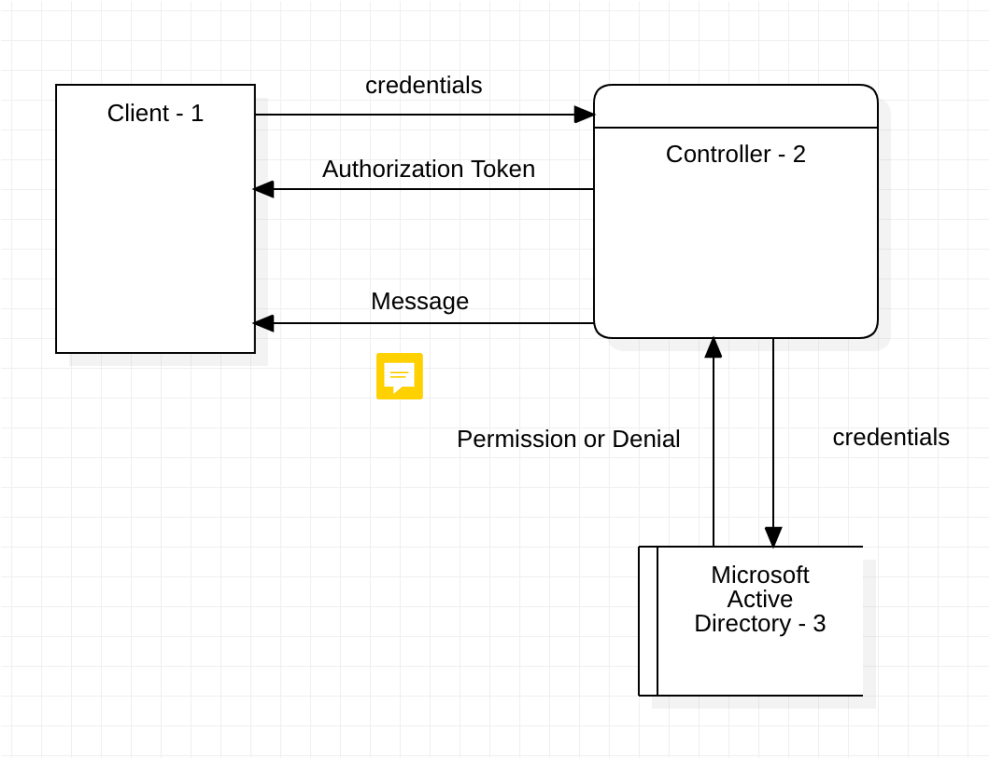
Viewpoint	UML 2.0 Class Diagram

3 - Router / Controller



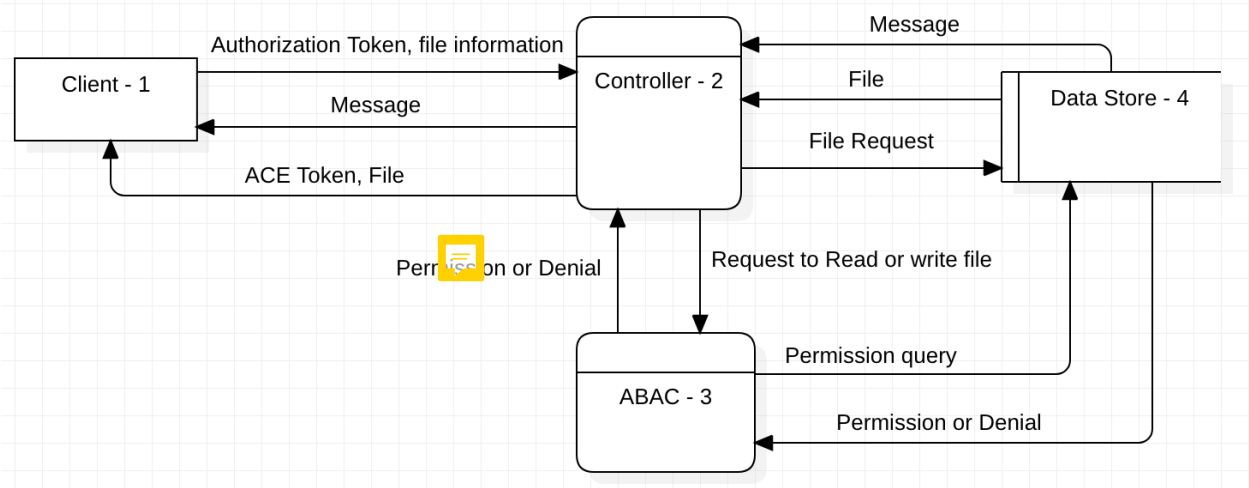
Name	View 3 - Router / Controller	
Description	Diagram detailing the workings of the router controller.	
Design Concerns	Routes information to where it needs to go.	
Requirements		
Elements	**Router Processor -** The processor that figures out what data packets go where.	**Director -** The component that sends off the data packets.
	Interface io - The input/output port for the interface.	**Date Store io -** The input/output port for the data store.
	MAD io - The input/output port for the MAD.	**ABAC io -** The input/output port for the ABAC.
Referenced By		
Viewpoint	UML 2.0 Component Diagram	

View 3.1 - Authentication



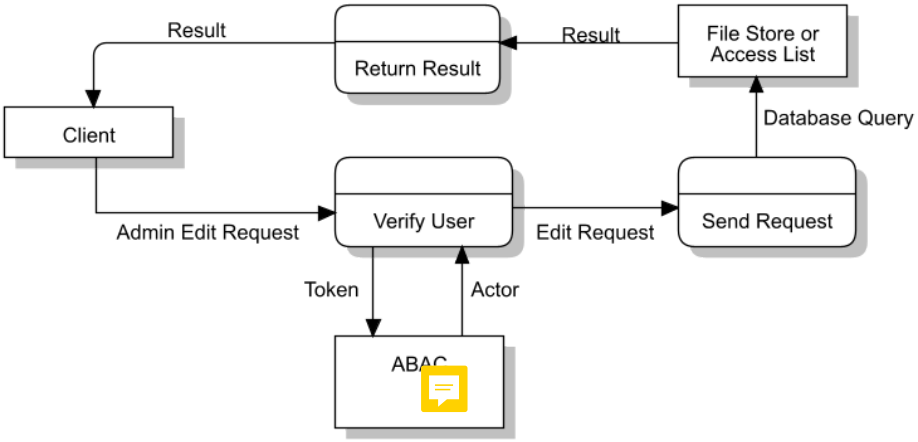
Name	View 3.1 - Authentication
Description	Diagram detailing the flow of data when a user attempts to gain authorization to access the system.
Design Concerns	Allows access to the system only to credentialed users.
Requirements	1.5.2
Elements	**Client - 1** The application used by users to access the database, may be the web or desktop version.
	Controller - 2 The logic and processes that interact with outside components on the client's behalf.
	Microsoft Active Directory - 3 The system which authorizes BYUI credentialed users.
Referenced By	
Viewpoint	Data Flow Diagram

View 3.2 - File Request



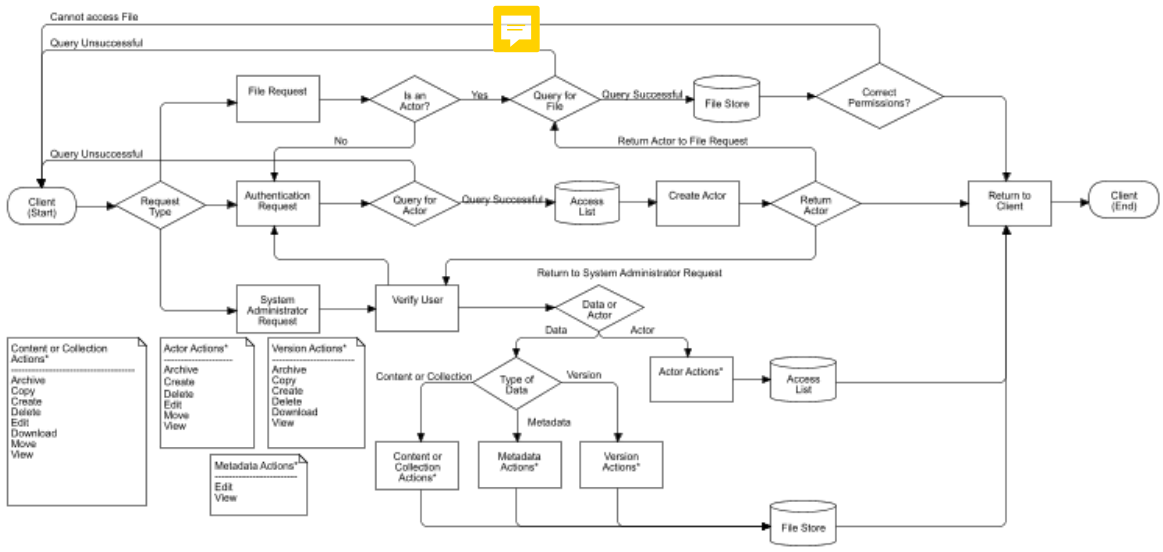
Name	View 3.2 - File Request
Description	This view represents the flow of data when the client requests access to a file and the system's response. The client will send their authorization token, along with file information and eventually either receive a invalid message (Because there was an issue with their request) or get an ACE token and access to the requested file.
Design Concerns	Allows credentialed users to access files on the system and view and edit them according to the permissions of the user.
Requirements	1.3.10, 1.4.1, 1.4.3, 1.4.8, 1.5.4 - 1.5.6, More new ABAC requirements?
Elements	**Client - 1** The application used by users to access the database, may be the web or desktop version.
	Controller - 2 The logic and processes that interact with outside components on the client's behalf.
	ABAC - 3 The logic and processes that help determine whether a user has permission to perform actions on a file.
	Data Store - 4 Where all of the data is stored for the application, including the ACL and files**.**
Referenced By	
Viewpoint	Data Flow Diagram

View 3.3 - System Administrator Request



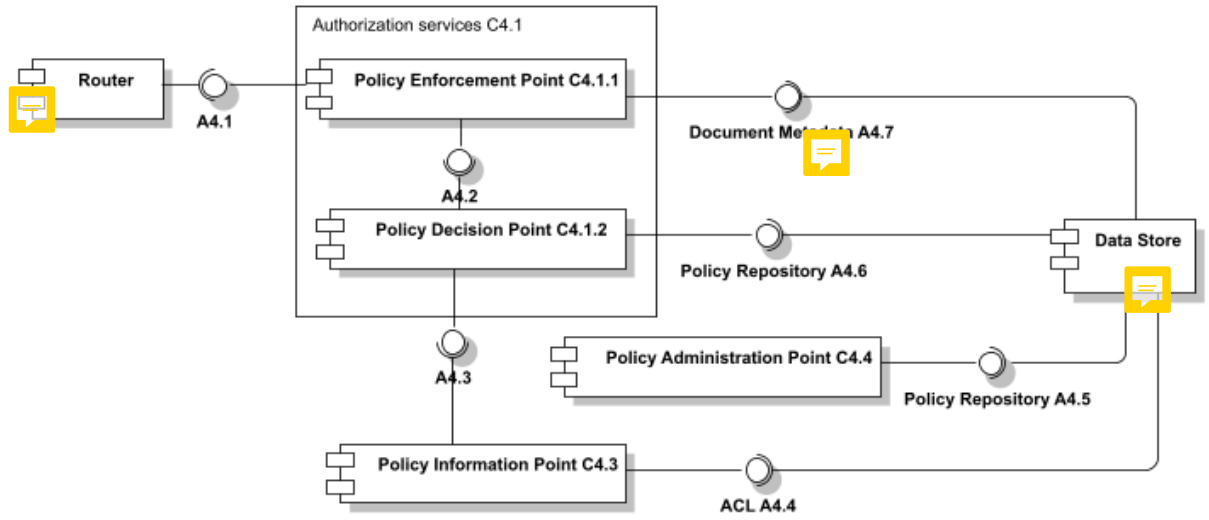
Name	View 3.3 - System Administrator Request	
Description	Details the flow of information during a System Administrator Request to the controller.	
Design Concerns	Handles System Administrator requests to either the Data Store or Access List	
Requirements	1.2.1 - 1.2.4, 1.2.9, 1.2.10, 1.2.13 - 1.2.17, 1.3.6 - 1.3.8, 1.5.6 - 1.5.7, 1.5.13	
Elements	**System Administrator Request** Routes a command from the client to the Data Store and back to the client to handle a System Administrator Request.	**Verify User Request** Verifies that the user is a System Administrator, and returns an actor.
	Attribute Based Access Control (ABAC) Verifies that the System Administrator is a System Administrator, and has permissions to perform the request.	**Send Request** Creates and sends a query to the Datastore (which encapsulates the File Store and the Access List).
	Return Result Receives the file, and returns it to the client.	**Client** The interface the request originated from.
Referenced By		
Viewpoint	UML 2.0 Data Flow Diagram	

View 3.4 - Router



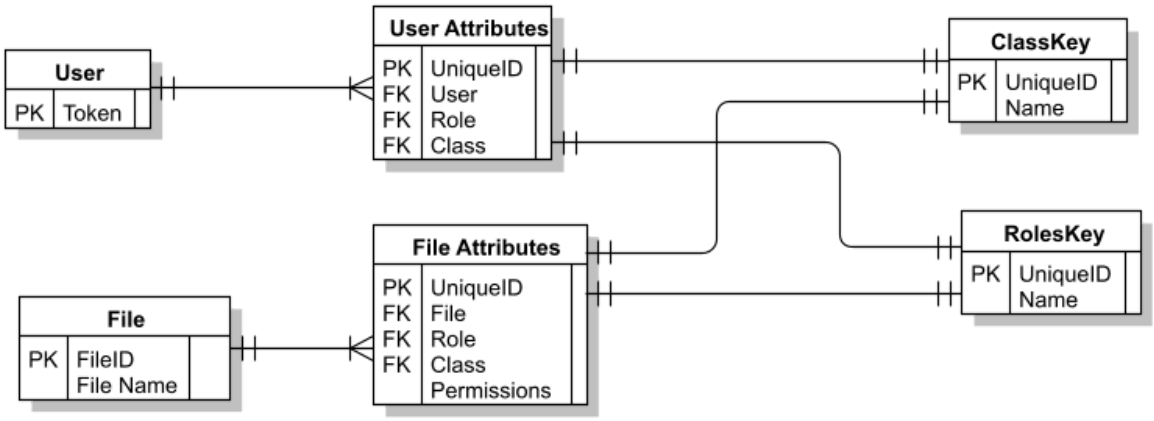
Name	View 3.4 - Router	
Description	A flowchart that contains a comprehensive view of what the router does given an input.	
Design Concerns	Routes commands and login tokens from a client to their intended locations, and returns the result of that request.	
Requirements	1.2.2 - 1.2.4, 1.2.9, 1.2.13, 1.3.3 - 1.3.8, 1.4.1 - 1.4.4, 1.4.9, 1.5.2, 1.5.4 - 1.5.7, 1.5.9, 1.5.13, 2.2.1, 2.2.2	
Elements	**Router** The system as a whole. It takes a command and a token from the client, and returns the result of that command.	**Request Type** **Remember to link to their DFDs** File Request Authentication Request System Administrator Request
	File Store Contains all of the file data, and is contained by the Data Store. This is where the actor's permissions are verified.	**Access List** Contains all of the actor data, and is contained by the Data Store. This is where the login tokens are verified. It returns an actor.
	Client The interface the request originated from.	
Referenced By		
Viewpoint	UML 2.0 Flowchart	

4 - ABAC



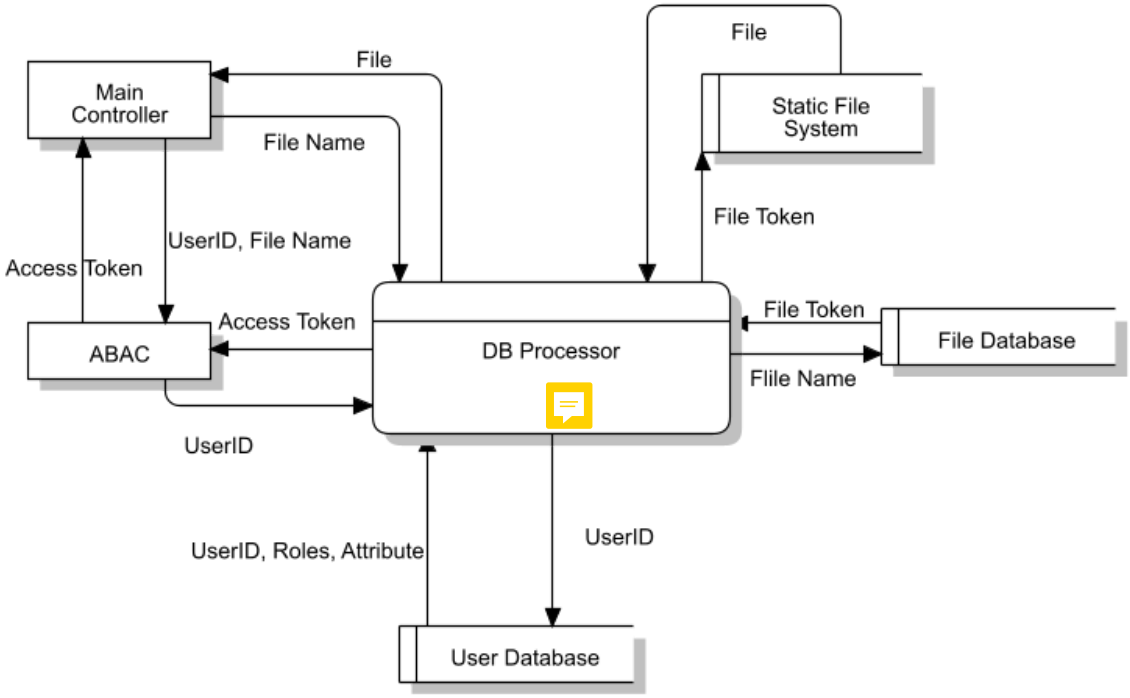
Name	View 4 - ABAC	
Description	Details of the components inside of the ABAC.	
Design Concerns	Enables only authorized users to access content based upon factors of what document they are trying to access and who the person is.	
Requirements	1.5	
Elements	**Authorization Services** - C4.1 A service to decide if an authenticated user has access to particular content.	**Policy Information Point (PIP)** - C4.3 Retrieves the required attributes and data needed by the PDP to make its decision.
	Policy Enforcement Point (PEP) - C4.1.1 Enforces policy decisions in response to a request from a subject requesting access to a protected object	**Policy Administration Point (PAP)** - C4.4 A user interface for creating, managing, testing, and debugging the digital policies and metapolicies. It will then store the policies in the appropriate repository.
	Policy Decision Point (PDP) - C4.1.2 Computes the access decisions based upon enacted digital policies. It will also handle conflicts in digital policies based on metapolicies.	
Referenced By		
Viewpoint	UML 2.0 Component Diagram	

View 4.1 - Database Access List



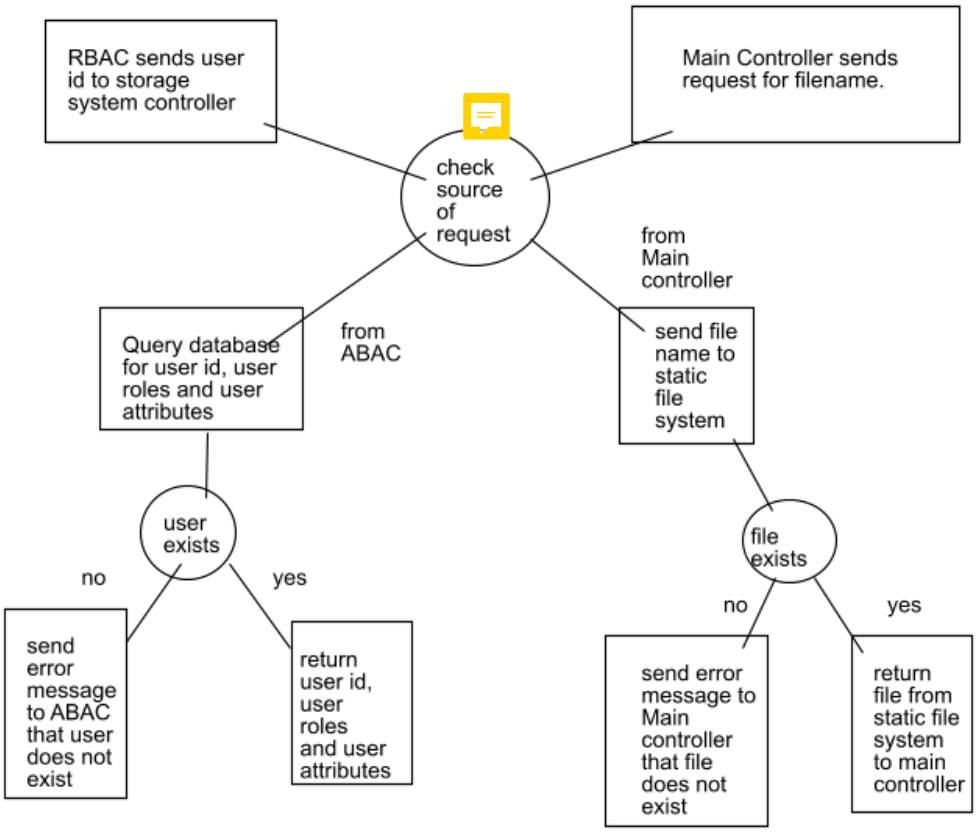
Name	View 4.1 - Database Access List	
Description	Relational database storing user and file access information	
Design Concerns	To make sure that users can't access files that they do not have permissions for.	
Requirements	1.2.3, 1.2.4, 1.2.9, 1.5.4, 1.5.5, 1.5.9	
Elements	**User -** Stores the users unique identifier	**User Attributes -** Stores a role and class attribute combination for a user
	File - Stores the file ID and name	**File Attributes -** Stores a role and class attribute combination for a file
	Class Key - A unique identifier for the class of each attribute	**RolesKey -** A unique Identifier for the role of each attribute
Referenced By	5.1	
Viewpoint	Entity Relationship Diagram	

5 - Data Store



Name	View 5 - Data Store	
Description	Data Store describes how data moves from one portion of the data store to another as well as how the data flow interacts with the other components of the system.	
Design Concerns	Provides a secure location for content to be stored and retrieved from.	
Requirements	1.1.1, 1.2.2, 1.2.3, 1.2.5	
Elements	**Controller- 5.1** Evaluates inputs given by ABAC and Main controller. Retrieves files or queries database to return information to Main controller and ABAC.	**Database-5.3** Relational database storing collection and file information.
	ABAC- 4 Attribute Based Access Control: authenticates users and allows access to content based on their attributes.	**Static File Store - 5.2** A repository for files that require no modification, processing, or generating, in order to be transferred to another end user.
Referenced By	View 1	
Viewpoint	Data Flow Diagram	

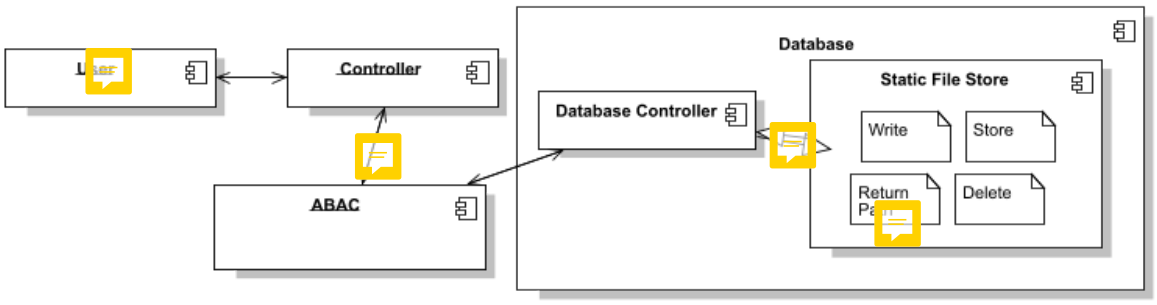
View 5.1 - Data Store Controller



Name	View 5.1 - Data Store Controller	
Description	The controller handles logic performed in the storage system. The controller will expect a user Id from the ABAC. The User Id provided from the ABAC will be used by the storage controller to query the database and returns the User id, user roles, and user attributes to the ABAC. The storage system controller will also handle file request from the main controller. If the controller will then send the file requested to the static file system. To be updated with file and user updating.	
Design Concerns	Controls data flow in the storage system. Security issues concerning file accessing.	
Requirements	1.1.1, 1.2.2, 1.2.3, 1.2.5	
Elements	**Main Controller** Contains data flow logic for the entire system.	**Database-5.3** Relational database storing collection and file information
	ABAC- 4 Attribute Based Access Control: authenticates users and allows access to content based on attributes.	**Static File Store - 5.2** A repository for files that require no modification, processing, or generating, in order to be transferred to another end user.
Referenced By	View 5	

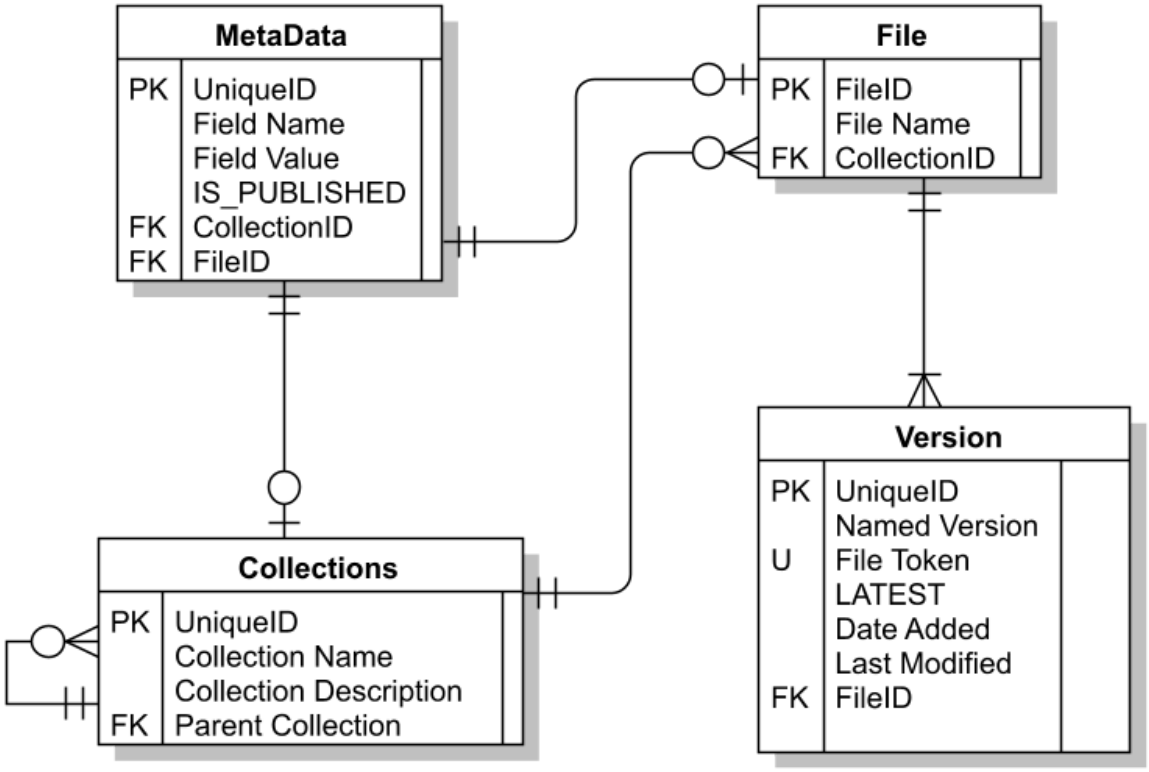
Viewpoint	Flowchart

View 5.2 - Static File Store



Name	View 5.2 - Static File Store	
Description	The portion of the Database component that reads, edit, and stores Static Files	
Design Concerns	The Static File Store will be controlled through specific User ABAC.	
Requirements	1.1.1, 1.2.1, 1.2.2, 1.2.5, 1.2.6, 1.2.7, 1.2.8, 1.2.11, 1.2.12, 1.2.13, 1.2.19, 1.3.1, 1.3.2, 1.3.3, 1.3.4, 1.3.5, 1.3.6, 1.3.7, 1.3.8, 1.3.9, 1.3.10, 1.3.11, 1.4.1, 1.4.2, 1.4.3, 1.4.6, 1.4.7, 1.4.9, 1.5.8, 1.5.11, 1.5.12, 1.5.13, 2.1.1, 2.2.1	
Elements	Store Static file information	Write collection data for file locations
	Delete path upon requests	Returns path for user interaction
Referenced By	C1.1.3.2	
Viewpoint	Component Diagram	

View 5.3 - Database Metadata



Name	View 5.3 - Database Metadata	
Description	Relational database storing collection and file information	
Design Concerns	Fulfills the need to have a quick reference to where the files are stored within the CMS	
Requirements	1.2.2, 1.2.5, 1.2.12, 1.2.13	
Elements	**Metadata -** Data about files and collections, including field name, field value, and which collection or file the metadata is referring to	**Collections -** A directory of files and/or collections
	File - Stores the file name, and reference to collection	**Version -** Refers to a specific instance of a file that can be found in the Static File Store
Referenced By	5.1	
Viewpoint	Entity Relationship Diagram	

Appendix

Traceability Matrix

(x if element fulfills requirement, blank if not)

Requirements	Component 1.2 Desktop Client	Component 1.3 Browser Client	Component 1.4 Controller - Routing	Component 1.5 RBAC	Component 1.6 Data Store	Component 1.6.3 Database
1.1.1	x	x	x	x	x	x
1.1.2			x	x		x
1.1.3			x	x		x
1.2.1	x	x			x	
1.2.2					x	
1.2.3			x	x		x
1.2.4			x	x		x
1.2.5					x	
1.2.6			x		x	
1.2.7			x		x	
1.2.8	x	x			x	
1.2.9				x		x
1.2.10			x			
1.2.11					x	x
1.2.12				x	x	x
1.2.13			x	x	x	
1.2.14	x	x				x
1.2.15	x	x				x
1.2.16	x	x		x		x
1.2.17						x
1.2.18		x	x			
1.2.19	x		x		x	
1.3.1					x	
1.3.2		x	x	x	x	
1.3.3		x	x	x	x	
1.3.4		x	x	x	x	
1.3.5					x	
1.3.6		x		x	x	
1.3.7					x	
1.3.8		x	x	x	x	
1.3.9					x	x

1.3.10					x	x
1.3.11					x	x
1.4.1		x	x		x	x
1.4.2	x		x		x	x
1.4.3	x	x	x	x	x	
1.4.4	x		x			x
1.4.5	x	x	x			
1.4.6	x		x		x	
1.4.7	x		x		x	
1.4.8	x	x	x			
1.4.9	x				x	x
1.5.1			x	x		
1.5.2			x	x		
1.5.3			x	x		
1.5.4		x	x	x		x
1.5.5		x	x	x		x
1.5.6		x	x	x		
1.5.7		x	x	x		
1.5.8			x	x	x	
1.5.9			x	x		
1.5.10				x		
1.5.11	x	x			x	x
1.5.12	x	x			x	x
1.5.13	x	x		x	x	
2.1.1	x	x	x		x	x
2.2.1	x				x	x
2.2.2	x	x	x		x	
2.2.3			x		x	