CS 3604 Case Study Library III

Final Report

Theo Ouzhinski, William Denman, Katie Geibel, Thrilok Shivaraman

Multimedia, Hypertext, and Information Access (CS 4624)
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
May 11, 2023

Instructor: Dr. Edward A. Fox
Client: Dr. Dan Dunlap
# Table of Contents

1 **Introduction** .......... 1
   1.1 Existing Work .......... 1
   1.2 Approach .......... 1

2 **Requirements** .......... 2
   2.1 Functional Authentication .......... 2
   2.2 Metadata for Filtering .......... 2
   2.3 Student Interaction/Living Library .......... 2
   2.4 Resilience (and Recovering) .......... 3

3 **Design** .......... 3

4 **Implementation** .......... 6
   4.1 Original Plan: OIDC and VT Gateway .......... 6
   4.2 Current Plan: Google Authentication with AWS Cognito .......... 7
   4.3 Front-end Development .......... 7

5 **Future Work** .......... 8

6 **Testing, Evaluation, and Assessment** .......... 9
   6.1 Assessment Criteria .......... 9
   6.2 Pilot Testing .......... 10

7 **Developer’s Manual** .......... 10
   7.1 Prerequisites .......... 10
   7.2 AWS Resources .......... 11
      7.2.1 Amplify .......... 11
      7.2.2 Cognito .......... 12
      7.2.3 DynamoDB .......... 12
      7.2.4 AppSync .......... 12
      7.2.5 Lambda .......... 12
   7.3 Deployment from Scratch .......... 13
   7.4 Deployment through CloudFormation .......... 13
      7.4.1 Configuring Cognito .......... 14
   7.5 Supporting Procedures .......... 16
      7.5.1 Backup/Restore of Key User Data .......... 16

8 **Lessons Learned** .......... 16
   8.1 Timeline/Schedule .......... 17

9 **Acknowledgements** .......... 18

References .......... 19

A **Methodology Assignment** .......... 21
List of Figures

1. Screenshot of current sign in page ........................................ 4
2. Screenshot of addition to upload screen ................................ 5
3. Screenshot of existing like button and planned placement of edit button ........................................ 5
4. Screenshot of the configured identity provider (with secrets redacted) ........................................ 6
5. Screenshot of Network tab when currently authenticating to VT Gateway ........................................ 7
6. Screenshot of potential plan for the edit page ........................................ 9
7. Sample AWS CloudFormation template ........................................ 14
8. Screenshot of Configured AWS Cognito Identity Providers ........................................ 15
9. Screenshot of the Hosted UI Config ........................................ 15
10. Screenshot of the Hosted UI Page ........................................ 15
11. Our goals split into a mapping of tasks and subtasks ........................................ 22
12. Descriptions of service implementation ........................................ 23

List of Tables

1. List of Relevant AWS Services ........................................ 12
2. Team Roles ........................................ 17
3. Timeline of Group Activities ........................................ 17
Abstract

For a well rounded CS curriculum, it is important to teach both technical coding skills and real world ethical and professional wisdom. Therefore, all Virginia Tech computer science students are required to take CS 3604: Professionalism in Computing, coordinated by our client, Dr. Daniel Dunlap. This class explores the context of computing in society by having students research and present on hot button controversies, topics, or events in computer science. They are to pick a topic of their choice pertaining to the Internet, Artificial Intelligence, Intellectual Property, Commerce, or Privacy, and through thorough research and the learning from the class itself, the students construct a case study, consisting of both a presentation and report.

The Case Study Library website provides a platform through which these case studies can be viewed. Having been developed by two previous groups, the Library previously allowed for student case study upload, searching, and filtering by course topic. However, upload was through one admin account given to all students provided by the teacher. This meant once a student uploaded, they could not go back to edit their submission as there was no way to link users to uploads. Additionally, the interactivity of the website was limited.

We intended to focus on three overarching areas: implementing login functionality so that students can log in using their Virginia Tech accounts, creating the ability to add tags to archive submissions, and adding liking functionality. Enabling Virginia Tech Google account login allowed the application to link users with their uploads and thereby allows them to edit. Adding tags and liking would improve the interactivity of the site, by allowing users to filter by tags and sort by number of likes. In addition, it would enable future work to develop more advanced functionality such as recommendations and commenting.

As we discovered, maintaining cloud infrastructure and interfacing with various authentication providers is no easy task and, thus, we were unable to complete all we set out to do. However, we did implement much of the foundational back-end work for centralized authentication and foundational front-end work for liking and metadata. Specifically, the front-end exists for students to select tags from a pre-populated list during case study upload and to view the like count. We also completed much needed code and infrastructure cleanup that will allow future developers to work on a cleaner codebase and cloud environment.
1 Introduction

CS 3604 (Professionalism in Computing) is a required class for all computer science students at Virginia Tech. Students in the class “study the ethical, social, and professional concerns of the computer science field through case studies of reliable, risk-free technologies, and systems that provide user friendly processes” Virginia Tech Department of Computer Science (2023). In order to ensure that graduating computer science majors understand the implications of the computing systems they build and use in a broader societal context, students in the class present on case studies from the real world and analyze them.

The case study library itself provides students in the class with a medium to store their case study work, along with providing them the ability to view prior class case studies to draw inspiration from. This library system currently is very limited, having only a single user to the system, and not allowing any features other than viewing the case study itself. The prior groups that have worked on this project have done a good job in creating the groundwork and scaffolding, allowing our current group to progress forward and add new features to the existing library.

1.1 Existing Work

Dr. Dan Dunlap, our client, is the principal instructor for the course, and wishes to maintain a “living library” of case study presentations and reports from prior offerings of the course. Students teams from CS 4624 in spring 2022 (Wieder et al. 2022) and fall 2022 (Bagbey, Betsill, Elgeoushy, & Setareh, 2022) have been working on the CS3604 Case Study Library, a serverless cloud application storing the recent case studies created by students in the CS 3604 class.

The Library (which is currently accessible at https://casestudies.cs.vt.edu) is based off of the Virginia Tech Digital Library Project (DLP) (Chen, 2023). So far, students are able to upload their case studies and their reports, and a user is able to search through the various case studies in the system. As it stands, the only functionality supported is to sort by collection, of which there are 6, and to search. We plan on expanding upon the existing work done by the previous students.

Prior to our work, the most recent software is hosted on GitHub (Bagbey, Betsill, & Setareh, 2022).

1.2 Approach

As described in section 2, one of the primary goals during this semester is setting up a functional centralized authentication system to enable students to upload submissions associated with individual PIDs. Due to uncertainty with the complexity of setting up authentication, developing advanced metadata functionality and increasing student interaction may not be completed.
2 Requirements

2.1 Functional Authentication

Presently, the Case Study Library functions on a single username and password that all students share. Any reliable functionality which would require interaction from unique students would require an authentication system which allows students to sign in with their own accounts. At first, this was intended to be done through OIDC (OpenID Foundation, 2011), but since this has proven to be a dead end this semester, Google authentication is the new focus.

Using Google federated authentication with Amazon Cognito is a well-documented process, which makes this more doable than using Virginia Tech’s custom gateway service. Since Virginia Tech emails all use the same unique domain name, this works the same way as VT CAS would otherwise have.

An extension of this requirement is allowing users to edit their submissions – this would be too unsafe to allow with only one account for the whole site, so a functional authentication system is the first step to allowing users to edit their own submissions.

Finally, there is no functionality present for logging out. This should be implemented as well.

2.2 Metadata for Filtering

Implementing functionality for attaching metadata (such as tags) to individual submissions as they’re uploaded allows for a more specific filtration system for the case study search functionality that is currently implemented.

Presently, the Case Study Library is implemented to (meaningfully) search only by one of six categories. Being able to define a set list of tags which can easily be added to in the future would allow for much more specific searching – as the Case Study Library approaches a thousand uploads, six categories isn’t enough to meet requirements for a functional search system.

Further, metadata could be used for drawing connections between submissions, allowing for a feed of related or recommended case studies to visit.

2.3 Student Interaction/Living Library

Currently, the Case Study Library feels more like a repository than a library. Adding functionality for user interaction or other ways for the Case Study Library to become a “living library” would enhance user experience greatly.

One use case for such an improvement would be in order to support a class assignment; perhaps students could look at past case studies in the library and comment on them, or add them to a list of “favorited” case studies, or “like” case studies which interest them.

Another use case would be the aforementioned recommendation system, which adapts to the case study that is currently being viewed, as well as possibly to user habits.

Ultimately, Dr. Dunlap seeks use cases for the Case Study Library outside of CS 3604, so any functionality beyond an authentication system and metadata should contribute to this...
2.4 Resilience (and Recovering)

An unforeseen requirement of this project which took up most of our time was boosting the resilience of the platform.

Part of this involves moving the GitHub repository (Ouzhinski, Denman, Geibel, & Shivaraman, 2023) for the Case Study Library to a new GitHub organization which can be managed by Dr. Dunlap instead of relying on keeping the repository within each successive team. It makes more sense for Dr. Dunlap to own the repository so that he can give access to new groups himself as needed.

The rest of the process involves rectifying mistakes that the previous groups made and making certain updates to the code. This included backing up the archive tables on DynamoDB, syncing the cloud state with the deployed website, updating JSON packages within the code, as well as changing the deployed version to not hard-code the name of the S3 bucket it pulls archives from.

The time consumed to address this requirement was so great that much of the other requirements will appear as future work as well here – it took a long time to re-engineer the website after these mistakes caused the site to go down for an extended period. For this reason, an extension of this requirement is making a detailed developer’s manual that will hopefully be of use to future groups in case they reach a blocker on the same scale that we did.

3 Design

For our design, we had to consider how the addition of student and administration authentication would change the website. Firstly, when uploading, users now must first be prompted to input their login information. Thus we created a login prompt that appears as soon as you click on the “Upload” tab. This way, there is no confusion that you must login to upload and users are prompted only when they need to be. Our design of this login prompt is shown in Figure 1.
Additionally, now that users will be able to be authenticated, they will be provided the option for editing their submission. On every archive page, an edit button is present underneath the case study thumbnail and above the share options. This can be seen in Figure 3.

We also began setting up for the addition of tags to case studies. For this we made changes to the Case Study Student Upload page. When filling in the upload page, there is now a drop-down list that can be configured to hold a list of tags. The students can choose the tags they would like to include from this list. This addition can be seen in Figure 2.
We have also begun the implementation of a liking feature. We have created and added a like button to each archive page that is above the share options and alongside the edit button. This button displays likes, but currently this like number is a static placeholder, as it is not set up with the back end. You can see our like button in Figure 3.

Figure 2: Screenshot of addition to upload screen

Figure 3: Screenshot of existing like button and planned placement of edit button
4 Implementation

The entirety of the Case Study Library is hosted on Amazon Web Services using serverless components managed by AWS Amplify. **AWS Amplify** ([Brandel & Amazon Web Services, 2023](#)) is AWS’s framework (and associated tooling) for enabling developers to deploy full-stack web apps without the need to manage servers. The serverless model means that VT only pays based on resources used (i.e., how much data is stored, how many users authenticate, how many queries are made, and how many people visit the website).

4.1 Original Plan: OIDC and VT Gateway

Our original plan of action was to utilize the single sign-on service managed by Virginia Tech, known as the Central Authentication Service (“CAS”). This service provides each VT student and member of the staff with a login (where their username is their university PID). The Virginia Tech Middleware group ([VT Middleware, 2023b](#)) manages all of the university’s identity and access management services, including VT Gateway. This service enables client websites to authenticate against the central sign-on service, through different protocols, including Open ID Connect.

Open ID Connect (OIDC) ([OpenID Foundation, 2011](#)) is a standard that “allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users”. In other words, with OIDC, users can be authenticated to web sites against some central identity provider without the web site having to store a directory of usernames and passwords.

In our case, we set up an OIDC client in the Library’s Cognito ([Amazon Web Services, 2023d](#)) user pool using the Virginia Tech identity provider. This was accomplished (in part) by following a guide provided by Virginia Tech’s Middleware group ([VT Middleware, 2023a](#)).

We configured the client (as described in [Figure 4](#)) using a client ID/secret obtained from the department.

![Figure 4: Screenshot of the configured identity provider (with secrets redacted)](#)

However, we were forced to abandon this approach to implementing authentication due
to the requirement that the OIDC redirect URI be a vt.edu domain. Due to this restriction, the redirect URI that we want does not match the registered redirect URI for the application. The redirect URI is part of the Cognito domain which is under an AWS domain.

As shown in the Figure 5 screenshot, the client makes an initial request to the Cognito domain’s /oauth2/authorize endpoint before redirecting to the /oauth2/authorize endpoint of the VT gateway service. As the redirect URI we request is different from the registered URI, the VT gateway redirects to the Cognito domain /oauth2/idpresponse endpoint with an error message.

If we were able to get the registered redirect URI to match the requested one, the user would instead see the VT login page. If successful in logging in to CAS, a token would be handed back after redirecting to the /oauth2/idpresponse of the Cognito domain.

Figure 5: Screenshot of Network tab when currently authenticating to VT Gateway

4.2 Current Plan: Google Authentication with AWS Cognito

Since we were unable to proceed with VT CAS authentication while using Amazon AWS, we decided to implement sign in through Google. The documentation on how to make Google authentication work with AWS Cognito is far more extensive, making it more approachable. This approach is possible, because to sign in to VT Google Accounts you must authenticate with CAS and the email addresses we have end in “vt.edu”. Thus, we can have users log in with their Google accounts and then check if the provided email address does in fact contain the correct domain. If it does not, the user will be immediately logged out. This leads to similar effects as if we had implemented CAS authentication. For this new approach, we have the federated identity on Cognito, and we have the page that will enable users to use Google authentication. The only missing piece to complete this authentication is to link the pages enabling Google authentication to the library itself, as the authentication wrapper for UploadSection.js does not include a link to said page, and we were unable to figure out how to do this in our time constraints.

4.3 Front-end Development

In order to add our additional front-end components, we worked with the files ArchivePage.js and UploadSection.js. In ArchivePage.js we implemented two UI button components, one for “Like Archive” and one for “Edit Archive” specifically. We utilized Ant Design (Ant Group, 2023) to get thumbs up and pencil and paper icons for these buttons to make them more visually appealing and user friendly. Above the like button we also display text to show the number of likes, currently with a placeholder number. To see the like button, see Figure 3.

The UploadSection.js is the component that implements the contents of the Upload Page. In this file we created a list of the tags, called METADATA_TAGS, that a student
can currently choose from, with a TODO statement noting how this list should later instead be linked to and reading from DynamoDB once that is implemented. Later in the file we implement the drop down menu mentioned earlier that allows students to view tagging options and choose for their case study. Currently, this menu holds the options that are listed in METADATA_TAGS and allows for selection. To see this drop-down addition, see Figure 2.

5 Future Work

Various authentication roadblocks and organization issues slowed our progress on this project, so there are a number of features that we have set up for but were not able to fully implement. We have done a good amount of work in the area of authentication, but in order to finish its implementation, one must figure out how to change the login screen to redirect to showing the providers (google in our case) that you can use to log in. Once this is completed, Virginia Tech accounts should have the ability to log in and create an archive. The next step with this is backend work to associate user uploads with particular usernames and store this information. Then, lastly, as users can now login, they will also need to be able to log out. Thus, the addition of a logout button must be implemented, likely to the case study upload page as this is where they log in.

A big step that will make a lot of the further additions relatively simple is adding new metadata fields to hold new information for each archive. This involves modifying the GraphQL [Amazon Web Services 2023a] schemas and the app itself. From there, the user who authored an case study, the tags the user associated with that case study, and the number of likes recorded for that case study will all be stored into the metadata of an archive. The ArchivePage.js file will have to be modified to account for these additional fields to store.

While our added edited button is present on all archive pages, it does not currently have functionality. In order to make it functional, an edit page should be created that the button will redirect the user to. Our plan was to make this page very similar to the student upload page, but to pre-load the fields contained in the currently existing archive so that the user could see what they had and change what they wanted. An example of how we imagined this to look is in Figure 6. It is also important that once functionality is implemented, that it is only functional if the user clicking edit is in fact the one who originally created the case study (or is admin). Thus, the button must first check what user is currently signed in before redirecting to the edit page.
As mentioned earlier, we began setting up for the introduction of tags to the site, but there are still a number of steps to complete this implementation. Once the tags are set up to be recorded in the metadata of each archive, it will be simple to make them then display on the archive page itself under where the like and edit buttons currently are. Additionally, our tag selection drop-down feature has to be linked to the backend where the list of possible tags is held so that it can display these options. Then the next step is being able to filter by tags. This will require changes to be done to the site.json file in AWS, as this file holds the search fields that are then loaded into the search page. One day, we hope that the use of tags might be extended to include user created hashtags that could automatically be added to the site and used for searching.

In order to make the like button fully functional as intended, it must be connected to the backend. Once each archive has a metadata field for number of likes, it will be a quick fix to change the static placeholder number that is displayed to instead display the saved number of likes for that archive. Additionally, once the like button is connected to this metadata, it will be another quick step to have it increment this total when clicked and communicate that back to the stored data. Further in the future, it would be ideal if the like button were adapted to also record which user clicked it so that a user cannot like it multiple times and they also have the option to unlike it. Additionally, a future hope that we had for this feature was to use number of likes (possibly alongside a user’s history) to create either a “Featured Case Studies” or a “For You” type queue that would display some of the most liked archives.

6 Testing, Evaluation, and Assessment

6.1 Assessment Criteria

Our initial aim for our assessment, before we ran into significant resilience problems, included goals with login functionality, the metadata implementation, and user interaction.

The primary new aim of our assessment is to ensure a functioning website in light of the array of problems we encountered this semester with development. A user should be able to
log in to the Case Study Library, upload case studies, and browse other case studies. This is the initial functionality of the website, and this should be preserved and restored as much as possible as development continues.

Regarding login, our assessment was that the website should prompt users to log in using OIDC – this has since been changed to using Google authentication as previously mentioned, but the idea is the same. Anyone with a Virginia Tech Google account ought to be able to log in with Google by parsing for the @vt.edu domain name. Users without this domain name should be prevented from logging in. This part of the assessment initially included storing the authenticated user in the metadata of uploaded archives, but this goal has since been postponed due to lost time.

The assessment for metadata functionality has been scaled back, but remains similar. The site should show a drop down on the Case Study Upload page which provides numerous tags pulled from a static list. Initially the assessment here included the backend implementation of storing tags with the archives, but again due to lost time, this goal has been postponed.

A ‘like’ button should also be present on each archive page along with the current number of likes this case study has. The current implementation lacks ‘like’ tracking currently – again, the backend implementation part of this assessment has been postponed. A number of likes should be displayed along with the like button nonetheless.

6.2 Pilot Testing

We have begun testing our various features by deploying our code to AWS. By deploying, we will be able to see if our UI components rendered correctly, and whether they function properly. So far, our button additions to the archive page appear as they should. We have also tested by deploying certain coding files to our own individual environments outside of the site. We have tested the additional tag drop-down for the upload page in this manner, and the functionality worked. We plan to continue deploying our various features and having different members in the group visit the website to test them. Additionally, the GitHub repository includes a Cypress testing framework (Lucas & Jaffre, 2022) that can be used for testing. Through pilot testing, we discovered that the upload functionality for the Library is not working properly. This is still the case at the time of writing, but it is being actively investigated.

7 Developer’s Manual

7.1 Prerequisites

To gain access to the AWS admin account, contact the client. Once you have access, you should be able to view all AWS resources you will need to edit (and deploy) the application. You also should clone the repository and cd into the directory.

In order to develop and deploy the Library, you must first install the amplify command-line interface (CLI) tool (Burden, 2021), run amplify configure, and follow the directions given. You should also install the aws CLI tool. Instructions can be found at https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html.
To obtain an AWS access key, visit https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/users/details/XXX?section=security_credentials (replacing XXX with the username of your administrative user). You should click “Create Access Key” and follow the steps to obtain both an access key ID and secret.

Securely store this access key and secret as this gives you full access to the account. To configure the aws CLI tool to use your credentials, enter your credentials when running `aws configure --profile [YOUR_PROFILE_NAME]`. Visit https://docs.aws.amazon.com/cli/latest/reference/configure/index.html for more information. This will store your credentials in the file at $HOME/.aws/credentials. You will need to use the --profile option whenever executing the aws tool.

The Library uses the us-east-1 AWS region.

Security Notes

- For security purposes, consider creating an IAM user for each individual developer of the Library.

- Consider using a tool such as “aws-vault” (https://github.com/99designs/aws-vault) to securely store your IAM credentials at rest.

If the app and backend already exist, in order to pull the backend environment from AWS:

1. Log in to the AWS console
2. Choose “AWS Amplify”
3. Click the app
4. Go to “Backend environments”
5. Find the “casedev” environment
6. Click “Edit backend”
7. Run the command that is listed: `amplify pull --appId [APP_ID] --envName [ENV_NAME]`

7.2 AWS Resources

As described\(^1\) in Table 1, the Library makes use of AWS-managed serverless solutions for file storage (i.e., S3 (Amazon Web Services [2023f])), databases (i.e., DynamoDB (Amazon Web Services [2023e])), app execution (i.e., AppSync (Amazon Web Services [2023g]) and Lambda), and user authentication (i.e., Cognito).

7.2.1 Amplify

Amplify (Amazon Web Services [2023c]) is the deployment framework the Library (and the DLP) uses. It does a lot of the leg work in managing the setup and dependencies of the web app (both frontend and backend API), databases, file stores, and user authentication.

There are two types of environments: hosting and backend. A hosting environment serves the frontend of the application (and handles redirection from https://casestudies

---

\(^1\)This table is constructed based on the CS 3604 Case Study Library II report at (Bagbey, Betsill, Elgeoushy, & Setareh, 2022).
### AWS Service | Resource Name | Link to Description
--- | --- | ---
Amplify | casestudies, hosting: dev, backend: dlpdev | subsubsection 7.2.1
Cognito | iawav2658176f3_userpool_658176f3-dlpdev | subsubsection 7.2.2
DynamoDB | Archive-dcfeiwidx5ezjiipllgulq45ue-dlpdev | subsubsection 7.2.3
Collection-dcfeiwidx5ezjiipllgulq45ue-dlpdev | Collection-dcfeiwidx5ezjiipllgulq45ue-dlpdev | subsubsection 7.2.3
Site-dcfeiwidx5ezjiipllgulq45ue-dlpdev | Site-dcfeiwidx5ezjiipllgulq45ue-dlpdev | subsubsection 7.2.3
S3 | collectionmap115006-dlpdev | subsubsection 7.2.3
AppSync | collectionarchives-dlpdev | subsubsection 7.2.4
Lambda | thumbnail-CreateThumbnail-BVxDrZJvihPE | subsubsection 7.2.5

Table 1: List of Relevant AWS Services

To the Amplify services), while the backend environment is a container of all the services needed for the app. In our case, this is the backend API (through AppSync), S3, DynamoDB, Lambda, and Cognito.

#### 7.2.2 Cognito

Cognito provides managed user authentication for AWS applications. The Case Study Library currently uses Cognito to provide authentication for the case study upload page and site administration pages.

We use a single user pool to handle authentication for all users to the Library. A user pool is a directory of users where users can log in directly (with a username and password stored on AWS) or through a third-party provider (e.g., Google, Facebook, Apple).

#### 7.2.3 DynamoDB

DynamoDB is the primary database for the Library. We have tables for the collections, archives, and site configuration data. This also includes paths on S3 buckets where the uploaded case studies (and their thumbnails) are stored.

S3 is Amazon’s file object store, i.e., you can store arbitrary files in this durable and performant file store. Each S3 bucket can contain many objects which are organized into keys. Case study files are stored under `public/casestudies/` and thumbnail images and other graphic content are stored under `public/sitecontent/image/default/`.

#### 7.2.4 AppSync

AppSync provides a managed GraphQL API which the app contacts in order to perform backend operations.

#### 7.2.5 Lambda

When a PDF or MP4 file is uploaded, the Library has a Lambda function that creates and uploads a thumbnail of the file.
7.3 Deployment from Scratch

**Note:** This section can be ignored if the Amplify app already exists. This section is solely meant if the application must be recreated.

We discovered a bug in Amplify that meant that performing a fresh deployment did not successfully conclude (see GitHub issue at [@peerhenry, 2020](https://github.com)). As described in the issue, to resolve this issue:

1. go to `amplify/backend/api/<api-name>/parameters.json` and add the key value pairs: "CreateAPIKey": 0 and "APIKeyExpirationEpoch": -1
2. execute `amplify push`. This will take some time and result in a significant amount of output. The command should complete successfully. If not, you have to debug.
3. in the file, set: "CreateAPIKey": 1
4. execute `amplify push`
5. in the file, remove "CreateAPIKey": 0 and "APIKeyExpirationEpoch": -1 from parameters.json
6. execute `amplify api update`
7. execute `amplify push`

The following environment variables must be set for the app:

1. `REACT_APP_REP_TYPE`: Default. This configures the collection type. The D must be capitalized.
2. `USER_DISABLE_TESTS`: true. This disables the test phase of the deploy process, saving time.

The following redirect must exist:

1. Source address:
   ```html
   <[^.]+>$|\.(?!css|gif|ico|jpg|js|png|txt|svg|woff|ttf)$)([^.]*)/$>
   ```
2. Target address: `/index.html`
3. Redirect type: 200 (Rewrite)

7.4 Deployment through CloudFormation

Under the hood, Amplify uses a service called CloudFormation (Amazon Web Services, 2023) in order to manage deploying the web app stack and supporting dependencies. CloudFormation is known as an “infrastructure-as-code” tool, i.e., a tool meant to ensure that the AWS infrastructure reaches a particular state. Understanding the underlying CloudFormation code was an important part of ensuring that we could actually deploy the app. Figure 7 shows a portion of a sample CloudFormation template.
7.4.1 Configuring Cognito

The AWS Cognito documentation is quite disjointed and seems to focus on social sign-in via Google, Facebook, etc. rather than custom OIDC providers. It turns out that configuring Google as an identity provider provides a good basis for custom OIDC providers (Amazon Web Services, 2023b).

Cognito is organized into “user pools”. The name of our user pool begins with `iawav` and ends with `casedev`. The identity providers are registered under the “Sign-in Experience” tab under the “Federated identity provider sign-in”.

To register the Cognito user pool domain, visit the “App Integration” tab, click “Create Cognito domain”, and follow the directions. Once the user pool domain is registered and the identity providers are registered, find the appropriate Cognito client and visit the “Hosted UI” section. You can then set the allowed callback URL as `https://casestudies.cs.vt.edu/case-study-upload`, with an authorization code grant, OpenID Connect scopes of `email/openid`, and the two identity providers (see Figure 9). The “View Hosted UI” link connects with an Cognito-hosted page (Figure 10).

We followed the Google setup instructions in order to set up the Google OAuth app and identity provider. We then have two identity providers set up (shown in Figure 8).
Figure 8: Screenshot of Configured AWS Cognito Identity Providers

Figure 9: Screenshot of the Hosted UI Config

Figure 10: Screenshot of the Hosted UI Page
7.5 Supporting Procedures

7.5.1 Backup/Restore of Key User Data

In certain cases, you may need to transfer data from one S3 bucket to another. This can be accomplished with the command:

```
aws --profile cs4624 --region us-east-1 s3 sync \
    s3://collectionmap115006-dlpdev s3://backup-casestudy-20230301
```

8 Lessons Learned

This project involved a significant amount of working with backend cloud services which many team members did not have experience with. These backend services, such as AWS Cognito and DynamoDB, were somewhat difficult to understand and work with. Over the course of the semester, we’ve become well-acquainted with these services and how they interact with each other to support the website’s functionality, but there have still been AWS-related challenges that, even with our months of gained experience, we have not been able to surmount. To help manage the workload, we have assigned each member of our team with different roles (see Table 2).

**Understanding the setup.** The first month of our project involved getting acquainted with the pre-existing system and figuring out where each of the different points of interest were for our expansion of the current system. One major lesson learned was the difficulty of jumping on a project that others had already made significant progress on. Our first major issue was migrating the project to a new organization, and then making changes on gitlab that would be reflected on the final page. The reasoning behind this was to allow us to have the ability to add people to this project after we leave.

We also discovered that modification to the prior AWS setup was needed to be made, which resulted in down time for the actual website during the semester. Since we were forced to re-engineer the website, the downtime took very long to correct and repair in order to promote the future resilience of the library. We discovered that we could deploy our code to other branches of the GitLab page and then deploy those branches separate from the Case Study Library’s official URL, which was very useful for testing our changes before pushing them to the live build.

Another major pitfall and lesson learned was the use of OIDC. We discovered that the bureaucratic work and setup in order to be able to utilize OIDC was simply too much, so we decided to swap over to Google to finish our authentication, which we speak more about in the Requirements section.

A major problem was the resilience of the previous system. As discussed, the migration was very difficult from one group to another due to the large amount of missing work and confusing documentation from the other manuals, which was likely due to the time crunch they had when setting up the page. We found that we had to re-learn how the previous groups had done their set up in order to figure out how to update and make it more future proof for any groups that would end up having to work on this library in the future. The time consumed to fix this requirement made it so that we were forced to reassess our goals,
causing us to lose out on many features that we had hoped to accomplish over the course of this project. Unfortunately this was unforeseen and could not have been predicted. However, we did learn how to properly reassess and regroup after realizing our initial goals would be too problematic to achieve in the term that we had.

<table>
<thead>
<tr>
<th>Individual</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will</td>
<td>Project Lead</td>
</tr>
<tr>
<td>Theo</td>
<td>Technical and Cloud Lead</td>
</tr>
<tr>
<td>Thrilok</td>
<td>Report Lead, Distinguished Secretary, and Co-Frontend Lead</td>
</tr>
<tr>
<td>Katie</td>
<td>Presentation Lead and Co-Frontend Lead</td>
</tr>
</tbody>
</table>

Table 2: Team Roles

8.1 Timeline/Schedule

Table 3 describes the work done during the semester.

<table>
<thead>
<tr>
<th>Time Period</th>
<th>Work Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 16 - February 1</td>
<td>Project launch&lt;br&gt;AWS and Case Study Library Credentials obtained&lt;br&gt;Weekly meeting times with Dr. Dunlap established&lt;br&gt;Base requirements set and approval obtained from Dr. Dunlap</td>
</tr>
<tr>
<td>February 1 - February 28</td>
<td>Moved code to new GitHub Organization for future-proofing&lt;br&gt;Diagramming how planned components will appear on website&lt;br&gt;Functionality for logout implemented&lt;br&gt;Beginning redeploy process&lt;br&gt;Main use cases within CS 3604 identified&lt;br&gt;Potential use cases outside of CS 3604 identified&lt;br&gt;OIDC ID and Secret obtained</td>
</tr>
<tr>
<td>March 1 - April 10</td>
<td>Begun and completed work on Like button&lt;br&gt;Worked on adding metadata tagging&lt;br&gt;Worked on edit functionality</td>
</tr>
<tr>
<td>April 11 - May 3</td>
<td>Finalized documentation and report&lt;br&gt;Set up Google application</td>
</tr>
</tbody>
</table>

Table 3: Timeline of Group Activities
9 Acknowledgements

We would like to first thank our client, Dr. Dan Dunlap (dunlapd@vt.edu), for his constant support and encouragement throughout this project. We would also like to thank the Virginia Tech Computer Science Department for funding this digital library for the benefit of all CS students. In addition, we thank Chris Arnold for getting us access to the VT Gateway application.

We also are indebted to the developers of the VT Digital Library Platform, including Dr. Yinlin Chen (ylchen@vt.edu), without whom this project would not have been possible.
References


A Methodology Assignment

The goals of each of the types of users that our system is to support include the following:

- Students should be able to upload submissions associated with their PID.
- Uploaders should be able to add tags to their submissions.
- Current CS3604 Students
  - Should be able to log into their own account using the CAS instead of all students using the same authentication.
  - Should be able to add tags to their uploads to the Case Study Library.
  - Should be able to “like” other users’ submissions, or otherwise interact with other submissions.
  - Should be able to filter their searches: by author, by using the tags that students used in their uploads, as well as by when the submissions were uploaded.
- CS3604 Instructors/TAs
  - Want the Case Study Library to be more usable in a class setting. Possibilities include:
    * Drawing connections between submissions
    * Allowing for more student interaction, e.g., where an assignment could be created wherein the students use the Case Study Library
- Past CS3604 Students
  - Should be able to find their own submissions (search-by-author functionality, if they didn’t submit anonymously)
  - Should be able to find case studies related to their own case studies (possibly through a recommended list of related items when looking at a case study)

We split up our goals into tasks and subtasks that are reproduced in Figure 11.
Figure 11: Our goals split into a mapping of tasks and subtasks

Our service implementations are described in Figure 12.
Our goals can be detailed as workflows that yield them, as follows:

- **Goal 1:** Students are able to upload submissions associated with their PID
  Workflow 1 = Service 1A + Service 1B + Service 1C

- **Goal 2:** Presentations must be sorted by tags
  Workflow 2 = Service 2A + Service 2B + Service 2C

- **Goal 3:** Allow a user to see recommended presentations
  Workflow 3 = Service 3A + Service 3B + Service 3C