

Computer Science and Engineering Department, University of  
Nevada, Reno

Lecture Goggles

Team #22, Logan Long, Nathan Yocum, Zachary Johnson

Dr. Sergiu Dascalu, Devrin Lee

Dr. Shamik Sengupta

February 22, 2018

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Abstract</b>	<b>3</b>
<b>Recent Project Changes</b>	<b>3</b>
<b>Technical Requirements Specification</b>	<b>3</b>
<b>Summary of changes in project specification</b>	<b>3</b>
<b>Updated technical requirements specification</b>	<b>3</b>
Functional Requirements	3
High Priority	3
Medium Priority	4
Low Priority	4
Non-Functional Requirements	4
Medium Priority	4
Low Priority	5
<b>Updated Use Case Modeling</b>	<b>5</b>
Detailed Use Cases	6
Updated Requirement Traceability Matrix	9
<b>Updated Design</b>	<b>10</b>
<b>Summary of Changes in Project Design</b>	<b>10</b>
Updated High-Level and Medium-Level Design	11
System level diagram	11
Program Units	12
Web Server	12
API Server	14
Data Structures	15
Account Info	15
Submissions	16
Subscriptions	16
Votes	16
Reports	17
<b>Detailed Design</b>	<b>18</b>
Login State Diagram	18
Admin Report Handling Activity Diagram	19
Create an Account Activity Diagram	20
Updated User Interface Design	20

<b>Updated Glossary</b>	<b>30</b>
<b>Engineering Standards and/or Technologies</b>	<b>31</b>
<b>Updated List of References</b>	<b>33</b>
<b>Contribution of Team Members</b>	<b>34</b>

# Abstract

The project proposed by Team 22 is a free, open-source, educational resource repository to help students gain a better understanding of school subjects. Lecture Goggles is meant to help resource sharing between students and professors. Initial implementation will be focused on a web application design using a back-end web server and database to quickly manipulate and store uploaded resources and user accounts. Lecture Goggles limits the administrator's interaction using a front-end design mimicking a tree structure that can be built upon as more resources and subjects are added.

## Recent Project Changes

Lecture Goggles has had minimal changes during the development of the project. React will still be used on the front end of the project with the use of Axios for GET and POST calls to the API. Django will continue to be used for the API but currently Team 22 is determining which web server will be chosen. Originally Apache was going to be used. The PostgreSQL database is being used for data storage and is the same as documented previously.

## Technical Requirements Specification

### Summary of changes in project specification

Currently no changes yet in the project specifications. As more parts are implemented there will need to be changes to some of the priority levels in the Functional Requirements section. The portion Team 22 is concerned about is the developer API. This portion is between medium and low priority. Otherwise everything is current.

### Updated technical requirements specification

- Functional Requirements

- High Priority

1. Lecture Goggles shall allow users to register for an account
2. Lecture Goggles shall allow users to log into their account

3. Lecture Goggles shall allow users to view posted content
4. Lecture Goggles shall allow users to vote on posted content
5. Lecture Goggles shall allow users to search for resources by topic
6. Lecture Goggles shall permit users to upload resources
7. Lecture Goggles shall permit all users to submit error reports
8. Lecture Goggles shall permit users to submit hyperlinks to resources
9. Lecture Goggles shall allow administrators to remove resources
10. Lecture Goggles shall allow all users to report inappropriate content

#### Medium Priority

11. Lecture Goggles shall allow users to subscribe to notifications for selected topics and uploaded resources.
12. Lecture Goggles shall support two-factor authentication
13. Lecture Goggles shall be able to accept donations
14. Lecture Goggles shall permit educators to have a verified user account
15. Lecture Goggles shall allow educator accounts to request removal of content

#### Low Priority

16. Lecture Goggles shall have a public facing API
17. Lecture Goggles shall have a mobile app

### ● Non-Functional Requirements

#### High Priority

1. Lecture Goggles shall be implemented using the Django web framework
2. Lecture Goggles shall be resistant to all attacks listed in the OWASP Top 10
3. Lecture Goggles shall allow users with a valid .edu email address to register for an account
4. Lecture Goggles shall be designed to be a responsive, mobile-first website
5. Lecture Goggles shall conform to WCAG 2.1 A levels of conformance

#### Medium Priority

6. Lecture Goggles shall be available for use 99% of the time during any 24 hour period
7. Lecture Goggles shall be a progressive web application

8. Lecture Goggles shall have a valid SSL certificate for all domains and subdomains
9. Lecture Goggles shall be GDPR compliant
10. Lecture Goggles shall be PCI-DSS compliant
11. Lecture Goggles shall use MongoDB to store locations of resources
12. Lecture Goggles shall conform to WCAG 2.1 AA levels of conformance

#### Low Priority

13. Lecture Goggles shall perform a search in no more than three seconds
14. Lecture Goggles shall not require users to transfer over 2MB of data on a fresh page load.
15. Lecture Goggles shall conform to WCAG 2.1 AAA levels of conformance

## Updated Use Case Modeling

In Fig. 5 an image of the Use Case Diagram displays the connections between actors and use cases. Each colored line represents an actor to keep track of interaction between the actors and use cases. To get a better description of each use case Fig 6. has a 2 to 4 sentence long description of each use case.

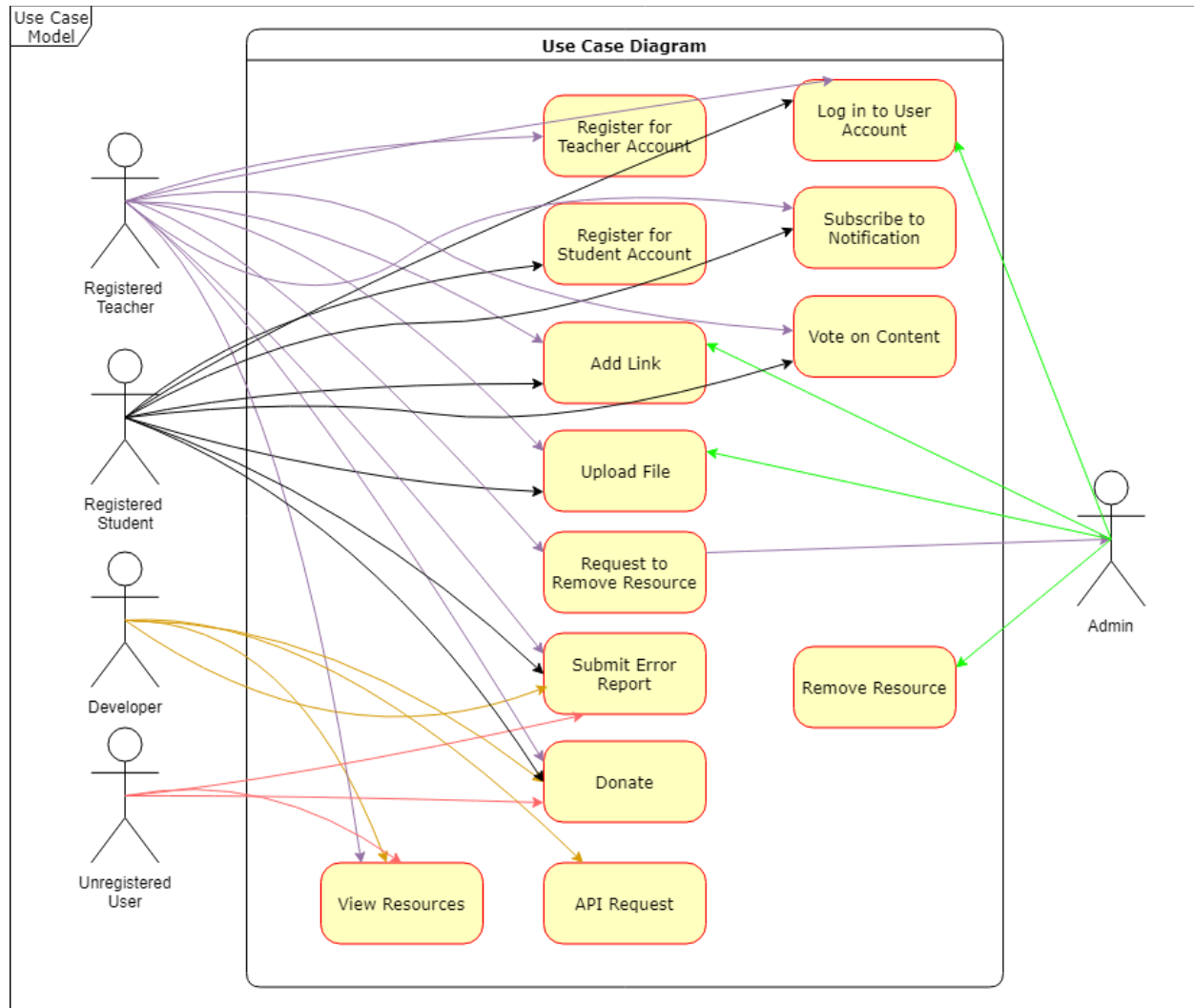


Fig. 1: Image of Lecture Goggles' Use Case Diagram. Each user points to Lecture Goggles' software features.

## Detailed Use Cases

**[UC1] Register for Teacher Account:** Using a verified teacher email, each teacher can register for an account. These accounts are given more privilege to report and request the removal of inappropriate material. These accounts will be free for all teachers.

**[UC2] Register for Student Account:** Using a verified student email, students are given the ability to add links, upload files, donate, and submit error reports. Only verified students can register and if banned will remove access from uploading resources. These accounts will be free for all students.

**[UC3]** Add Link: Using a popup box, all teacher and student accounts will have the access rights to add links to outside sources. These links will be compared to already available links, if the link is already posted, then it will inform the poster. If the link has not already been added, then a new post will be created.

**[UC4]** Upload File: All teacher and student accounts will have the access rights to upload files. Each uploaded file will require an accepted file format and description of the content of the file. Each file will be checked for malicious content to keep users safe.

**[UC5]** Request to Remove Resource: Each verified teacher account has the option to request removal of inappropriate links. Initial implementation will require the admin to review each request. After review, the admin has the choice to either remove or keep them available to users. Reported links will be frozen until review.

**[UC6]** Submit Error Report: All users will have access to submitting error reports. Since admins cannot know of all issues, having a good reporting system is key to the web pages success. These reports will go directly to admins for further review.

**[UC7]** Donate: There will be expenses to keep the web page up and running. Since Team 22 wants to keep the website free to every user, we will implement an optional donation section. Student accounts, teacher accounts, and anonymous users will be given the option to donate.

**[UC8]** API Request: Programmers will be given access to the website API. The API responds to programmers requests with values asked for. These values range from available topics to sub-content of the topics.

**[UC9]** View Resources: All users, whether registered for an account or not, have the ability to access all resources. This includes links and uploaded files. Although the platform requires an account to upload data, having access to resources will remain open to all users.

**[UC10]** Remove Resources: Only admin will have access to remove links and files. Typically after receiving a submitted report from a verified teacher account, the admin will review the complaint and remove the resource if needed. At any time admin can remove a link or file they deem inappropriate.

**[UC11]** Log in to user account: After teachers and students register for an account they will have access to log in. Once logged in they will have access as a teacher or student account.

**[UC12]** Subscribe to notifications: Registered users will have the ability to turn on notifications. These notifications, even if the webpage is not open, will be pushed to their desktop.



**[UC13]** *Vote on posted content:* Using a registered account, users can upvote or downvote webpage content. This will allow users to find good content faster.

Fig. 2: Detailed use case descriptors. Contains each use case and the detailed description of each use case.

## Updated Requirement Traceability Matrix

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13
FR1	X	X											
FR2											X		
FR3									X				
FR4													X
FR5									X				
FR6				X									
FR7						X							
FR8			X										
FR9										X			
FR10					X								
FR11												X	
FR12											X		
FR13							X						
FR14	X												
FR15					X								
FR16								X					
FR17	X	X	X	X	X	X	X		X		X	X	X

Fig. 3: Table demonstrating the relationship between use cases and functional requirements.

# Updated Design

## Summary of Changes in Project Design

Very little has been changed in our overall design, as most of our changes have been regarding implementation choices rather than overall design. The front end has had its look updated, while the other two servers have had no design changes. The overall architecture has also not changed, but Team 22 is looking into potentially switching over to Microsoft's managed services rather than using a virtual machine to accomplish the same job.

## Updated High-Level and Medium-Level Design

### System level diagram

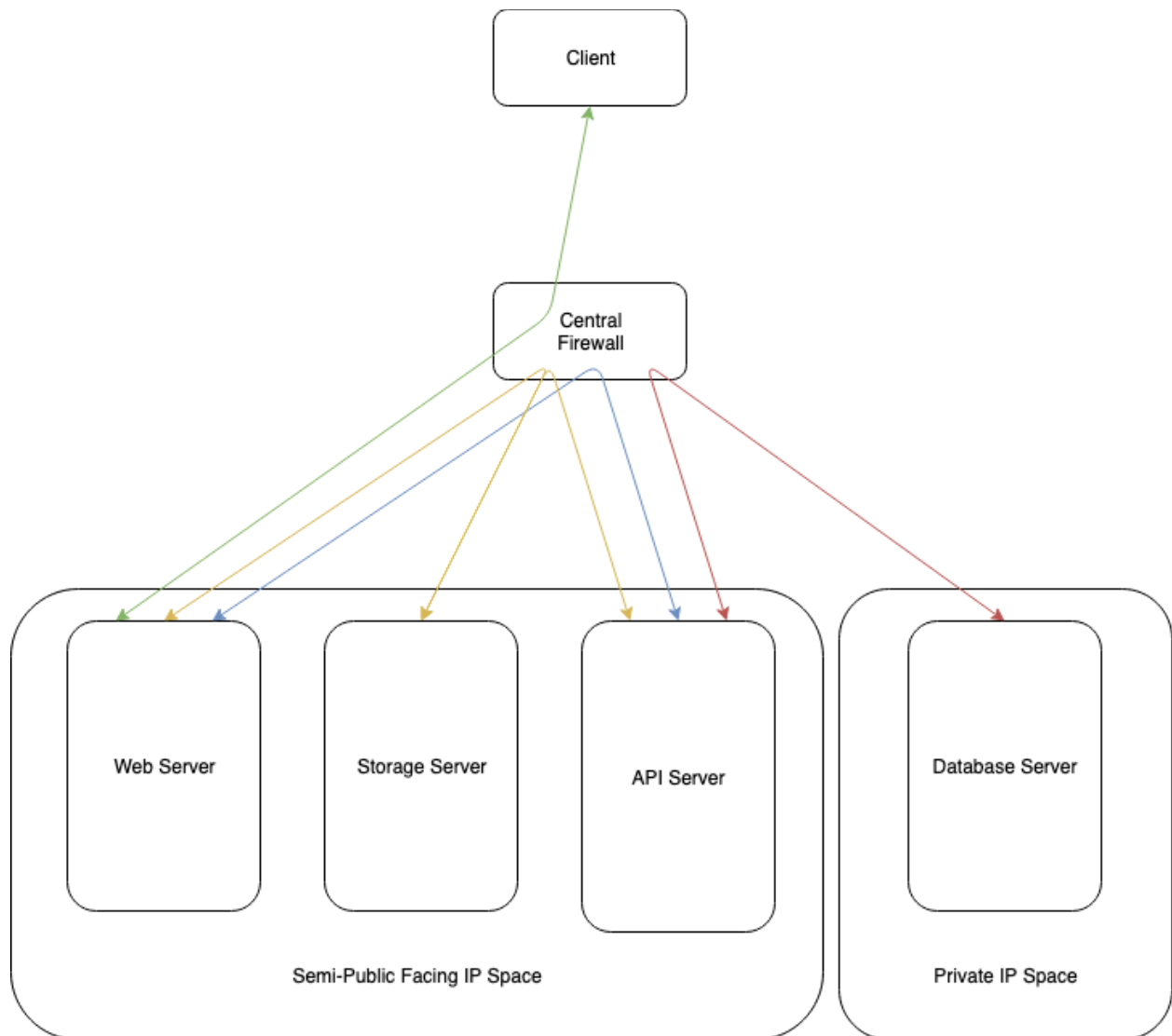


Figure 4: Lecture Goggles High-Level Architecture

Figure 4 shows the overall architecture of Lecture Goggles. In order to secure the application, all communication must move through a standalone firewall server. The firewall provides a barrier (logical or physical) between each server. Each server may only send requests to others if there is an arrow pointing at both servers in the diagram of the same color.

For any of the following communications to occur, they must pass through the firewall. The firewall implements a strict ruleset to prevent communication that does not follow the above pattern. This makes it extremely difficult for a malicious actor to compromise the entire application simply by infecting one server.

The firewall effectively creates “zones” on the internal network, allowing for any of the servers to be scaled vertically or horizontally without affecting the other servers.

The web server acts as the primary public server. It allows users to access the website, and perform basic tasks through the GUI. The storage server acts as a simple server to hold uploaded files, and serve them to the web server as appropriate. The API server is the core interfacing server, acting as a communication barrier between all external communication and the database server. It acts to ensure calls are made correctly, and that the valuable assets on the database server are protected. The database server is holds critical and sensitive data, and can only be communicated with through the API server. It sits in protected IP space to prevent direct attacks from malicious actors.

## Program Units

### Web Server

Table 1: Class information for the Web Server

Class	Description	Attributes	Methods
User	Base class for all user-related classes. “Virtual” class, so it cannot be initialized by itself.	FirstName LastName Email ProfilePicture Institution CreationDate APIKey Submissions AccountType	[Set/Get]Password() [Set/Get]Name() [Set/Get]Email() [Set/Get]ProfilePicture() [Set/Get]Institution() [Set/Get]CreationDate() [Set/Get]APIKey() [Set/Get]Locked() Vote() ReportSubmission() SubmitResource() GetSubmissions() DeleteAccount()
Student	Inherits from the User class. This		

	<p>class is the lowest on the totem pole of permissions, and has extremely limited functionality aside from basic website usage. This class does not currently introduce any new methods or attributes, but is separate from the User class to allow for adding more account types later on.</p>		
Instructor	<p>Inherits from the User class. Slightly above student, allows for priority service when requesting deletion of submissions. Allows for posting of “teacher-approved” content. Does not currently add new methods or attributes (for the same reasons as Student), as the account type dictates permissions.</p>		
Admin	<p>Inherits from the User class. Has the most permissions and can add or remove submissions at will. Can suspend/ban users as needed.</p>		<p>DeleteSubmission() BanUser() UnbanUser() LockUser() UnlockUser() [Set/Get]AccountType() ApproveUser()</p>

			ApproveReport() DenyReport() DeleteAccount()
Submission	Can be created by any type of user. Placed in the submission table after creation.	ID Link Subject Topic Score UserID	[Set/Get]Link() [Set/Get]Subject() [Set/Get]Topic() [Set/Get]Score() [Set/Get]UserID()
Report	Can be created by any user. Can only be resolved by an administrator. Though technically an object, does not have methods other than its constructor and destructor.	User SubmissionID Reason Priority	[Set/Get]User() [Set/Get]SubmissionID() [Set/Get]Reason() [Set/Get]Priority()

## API Server

Table 2: Class information for the API server

Class	Description	Attributes	Methods
Developer	Handles all communication to and from the web server on the semi-public subnet, and all communication from external developers. Requires a <b>private key</b> to execute all commands.	ListeningPort	UpdateSubmission() ) AddSubmission() AddUser() DeleteUser() AddReport() RemoveReport() EditUser() AddVote() RemoveVote()
Database	Handles all communication to and from the database server residing in private IP space. Uses a	ListeningPort	UpdateSubmission() ) AddSubmission() AddUser() DeleteUser() AddReport()

	separate public-private key pair to communicate with the database server.		RemoveReport() EditUser() AddVote() RemoveVote()
Storage	Handles all communication to and from the storage server, is not encrypted.	ListeningPort	RetrieveHardLink()

## Data Structures

In this project, database tables will be used. They are described as follows:

### Account Info

Table 3: Columns for Account Info SQL Table

Key	Value
User ID (Primary Key)	Unique ID generated by the SQL program upon creation of row.
Email	Email selected by user
SHA3 Hashed + Salted Password	Hashed account password for login purposes
Salt	Randomly generated salt for password
First Name	First name as entered by user
Last Name	Last name as entered by user
Link to profile picture	Link to picture on storage server
Institution	Academic Institution
Creation Date	Date created (Unix Epoch Time)
API Key	Private API Key, NULL if not issued
Is Instructor?	Boolean to tell if the user is an instructor



Is Admin?	Boolean to tell if the user is an administrator
Is Locked?	Boolean to tell if the user is locked out of their account
Locked Time	Date and time of lockout (Unix Epoch Time)

## Submissions

Table 4: Columns for Submissions SQL Table

Key	Value
Submission ID (Primary Key)	Unique ID generated by SQL program
Link	Link to resource (internal or external)
Subject	Overall subject that the submission relates to
Topic	The topic within a subject that the submission relates to
Thumbs Up Count	Overall number of “thumbs up”
Thumbs Down Count	Overall number of “thumbs down”
User ID	Unique ID of the user who submitted the resource

## Subscriptions

Table 5: Columns for Subscriptions SQL Table

Key	Value
User ID	Unique ID of user who subscribed to a topic
Subject	Subject to subscribe to
Topic	Topic to subscribe to, NULL if no specific topic is desired

## Votes

Table 6: Columns for Votes SQL Table

<b>Key</b>	<b>Value</b>
User ID (Primary Key 1)	Unique ID of the user who votes on a submission
Submission ID (Primary Key 2)	Unique ID of the submission being voted on
Vote	Value representing the vote. 1 for “thumbs up”, -1 for “thumbs down”

## Reports

Table 7: Columns for Reports SQL Table

<b>Key</b>	<b>Value</b>
User ID (Primary Key 1)	Unique ID of the user who votes on a submission
Submission ID (Primary Key 2)	Unique ID of the submission being reported
Report Reason	Reason for reporting the submission
Is Priority?	True if submitted by an instructor, false if reported by a student

# Detailed Design

## Login State Diagram

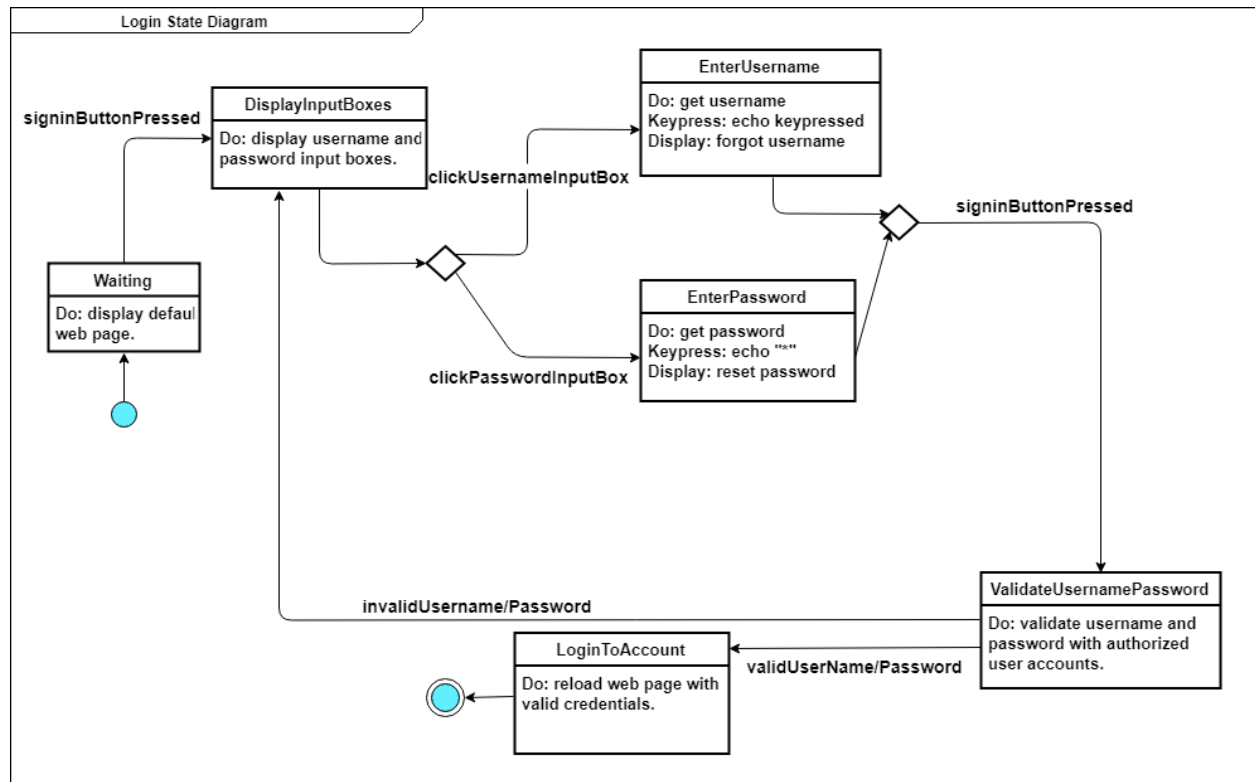


Fig 5. Login State Diagram which demonstrates a flow of events and actions for the login process.

Shown in Figure 5 is the Login State Diagram. The diagram displays the process in which each user would interact with logging into their accounts. Each time the user attempts to log in they will be required to follow a similar process. This process will be available to registered users only.

## Admin Report Handling Activity Diagram

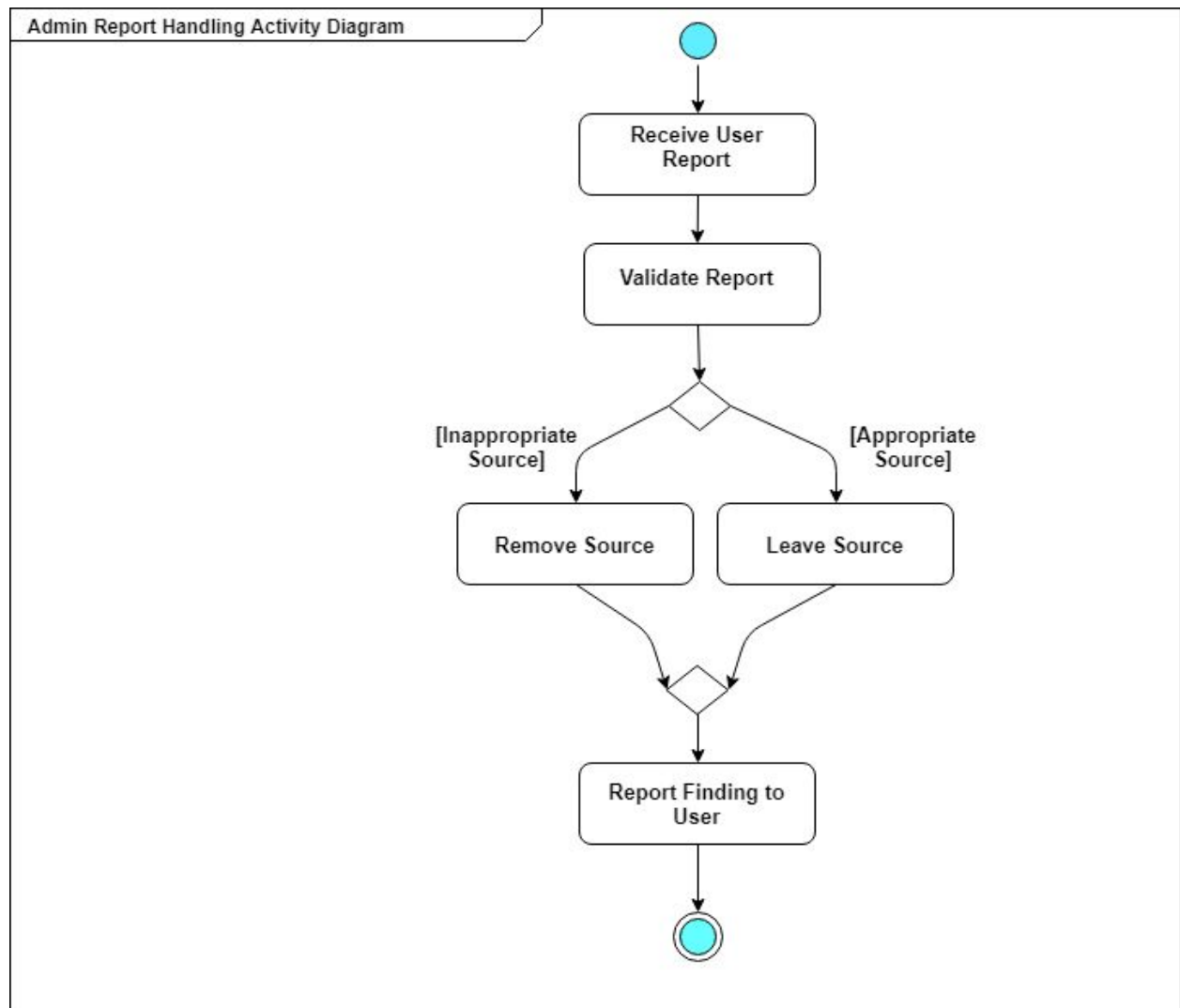


Fig 6. Admin Report Handling Activity Diagram which demonstrates the process in which inappropriate content will be evaluated.

Shown in Figure 6 Admin Report Handling Activity Diagram which demonstrates the process in which inappropriate content will be evaluated. Administration intervention will be required for each report for initial implementation. Future implementation will reduce administration overhead by running each link/upload through more thorough automated systems.

## Create an Account Activity Diagram

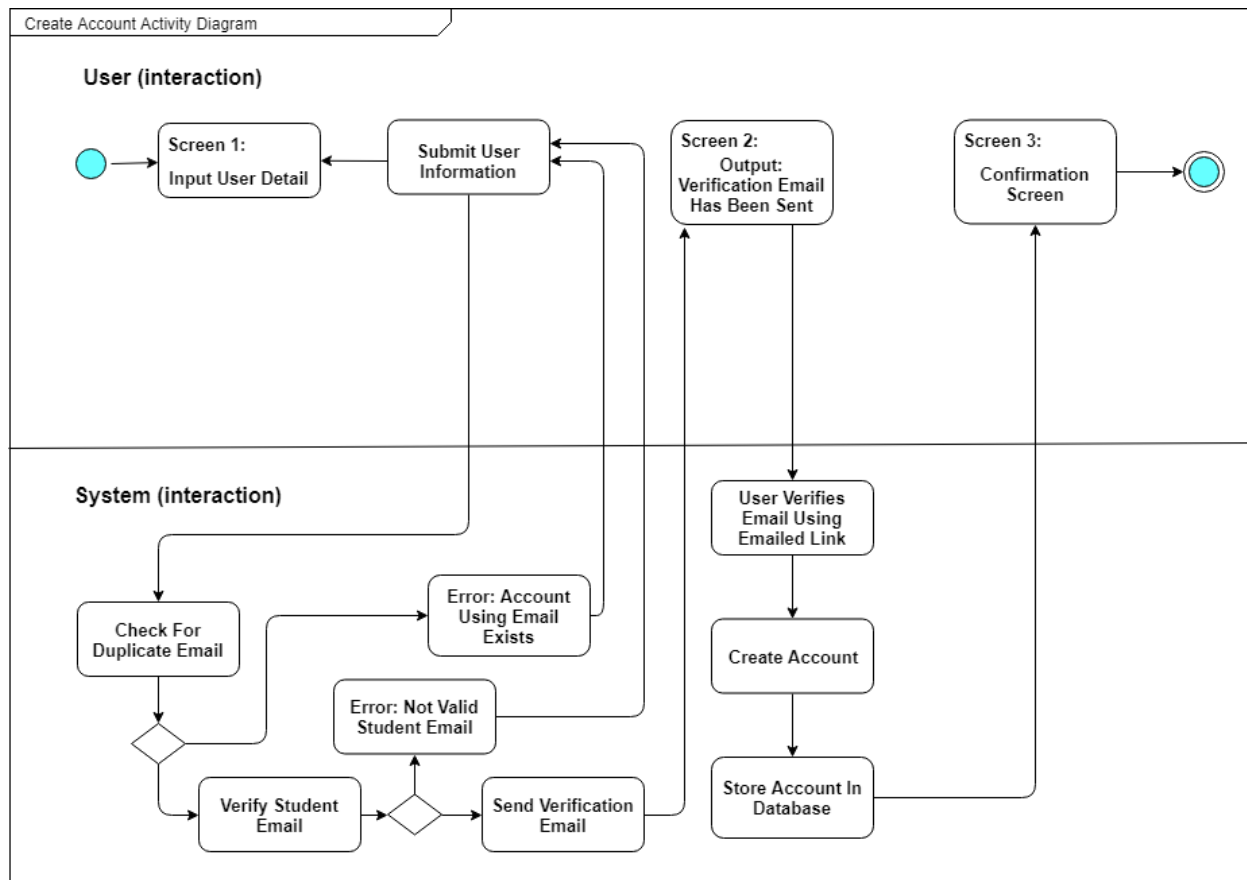


Fig 7. Create an Account Activity Diagram using system and user interaction sections.

Shown in Figure 7 Create an Account Activity Diagram is the interaction between system and user. Flowing from screen 1, 2, and 3 the user must follow steps to create an account. The system verifies each step along the process. This allows only accounts who have completed all tasks to be created.

## Updated User Interface Design

The frontend of Lecture Goggles is created using React with Styled Components for CSS-in-JS.



## Welcome!

Lecture Goggles is a free, open-source, educational resource repository to help students gain a better understanding of school subjects.

[SIGN IN](#)[CREATE AN ACCOUNT](#)

Fig 8. Landing page given user who is not signed in.



Sign In

Required


[Forgot your password?](#)

Continue

Cancel

Create An Account

Fig 9. Sign In page with error shown for invalid/empty email



First Name

Only alphabetical characters (A-Za-z) and hyphens accepted

Last Name

Only alphabetical characters (A-Za-z) and hyphens accepted

Email

Emails do not match

Confirm Email

Emails do not match

Password

Passwords do not match

Confirm Password

Passwords do not match

Institution

Required

Are you an instructor at your institution?

☐ Yes ☒ No

Continue


Cancel

Fig 10. Account creation with frontend validation. Continue button greys out when errors present in form.



Lecture Goggles

SubjectsTopicsResourcesSupportDevelopersSign InCreate an Account



First Name

nathan

Last Name

yocum

Email

nayocum@nevada.unr.edu

Confirm Email

nayocum@nevada.unr.edu

Password

•••••

Confirm Password

•••••

Institution

UNR

Are you an instructor at your institution?

☒ Yes ☐ No

Continue

Cancel

Fig 11. Account creation with valid input

Lecture Goggles

SubjectsTopicsResourcesSupportDevelopersSign InCreate an Account

ResourceSubjectTopic

Upload Resource

URL

https://github.com

Title

Github

Description

Github

Subject

TODO

Topic

TODO

CANCEL

SUBMIT

Fig 12. Uploading a resource with correct input. Tabs on top to switch to uploading subject and topic.

Lecture Goggles

Subjects Topics Resources Support Developers Sign In Create an Account

Resource Subject Topic

### Upload Resource

URL

8.8.8.8

Invalid URL

Title

DROP TABLE USERS; --

Titles can only contain alpha-numeric characters, hyphens, and spaces

Description

fdstfdstfdf

Subject

Accounting

Topic

TODO

CANCEL SUBMIT

Fig 13. Uploading resource with invalid URL demonstrating frontend validation errors.

Lecture Goggles

Subjects Topics Resources Support Developers Sign In Create an Account

Resource Subject Topic

### Upload Subject

Subject Name

CANCEL SUBMIT

Lecture Goggles

Subjects Topics Resources Support Developers Sign In Create an Account

Resource Subject Topic

### Upload Topic

Topic Name

Subject

Something

CANCEL SUBMIT

Fig 14 & Fig 15. Tabs for uploading topic and subject



## Developers

### Web API

Lecture Goggles has a API based on REST principles, which developers can utilize to create more applications to help students learn. Requests to the API can get, post, put, and request to delete resources. A public API is not currently available for Lecture Goggles, but it is a possible stretch goal for the future.

#### Read the documentation

[CS 426 Project Website](#)

[Front End Respository](#)

[API Respository](#)

#### API Methods

Coming soon...

## Source Code

Lecture Goggles is a fully open source applicaion. Read about how to contribute to the [frontend](#), the [API](#) or the [backend](#) of the website. Lecture Goggles is licensed under an [MIT](#) licence.

Fig 16. Updated wording for documentation page on website.



Fig 17: Updated Subject template. Changed from a card view to a list view for more information viewable at a glance.

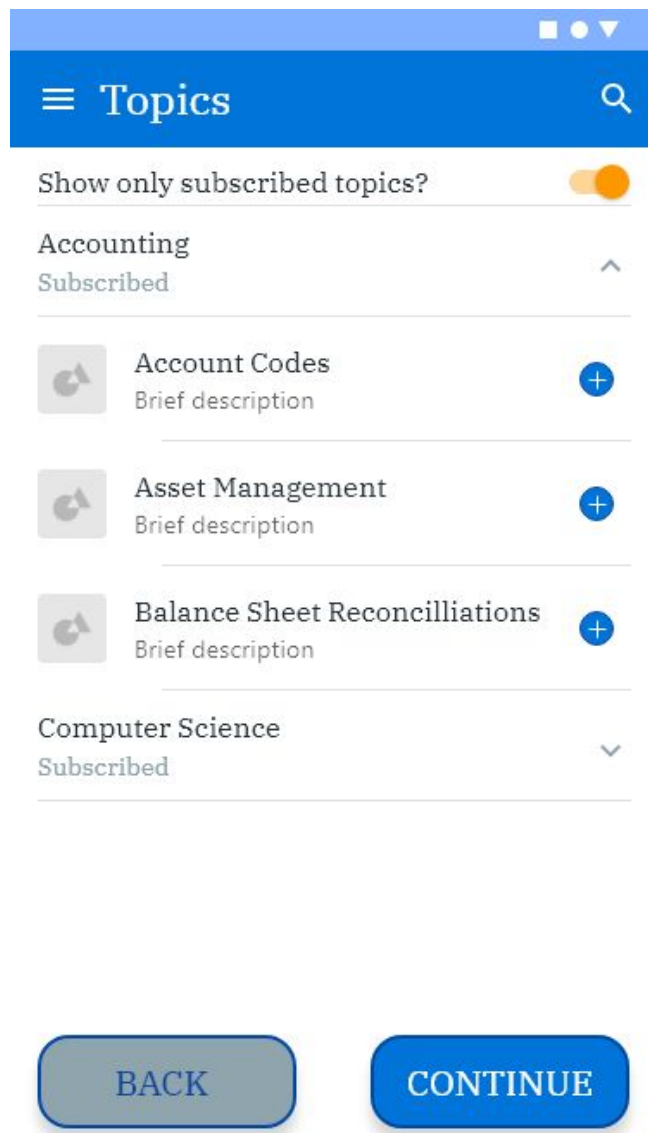


Fig 18. Wireframe of the topics view.

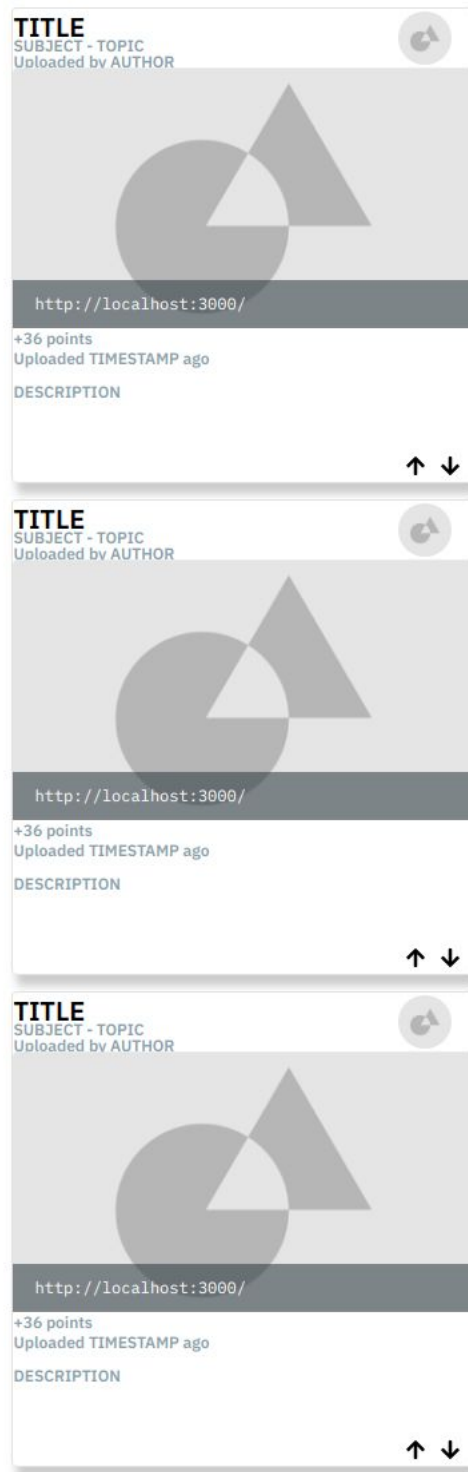


Fig 19. Template of the resources view.

# Updated Glossary

## 1. **Application Program Interface (API)**

An application program interface (API) allows developers to interface directly with a web application for the purpose of developing another application. APIs allow developers to interact directly with a website, such as retrieving information or uploading content.

## 2. **Backend Web Development**

Backend web development refers to working on the parts of a website that users do not directly interact with. This can include tools, servers, and databases that a developer creates to manage a website.

## 3. **Database**

A database is a set of data stored on a server. Databases hold important information, such as user accounts, hashed passwords, and other important information for websites to function.

## 4. **Framework**

A framework is a tool, or set of tools, developed to make development easier. For web development, common frameworks include React, Angular, and Vue for frontend and Rails, Express, and Django for backend. Frameworks automate parts of development and design and allow new design patterns to form.

## 5. **Frontend Web Development**

Frontend web development refers to the development of the user facing application. The front end of a website interacts directly with the web browser.

## 6. **Non-relational Database**

A non-relational database (often called a NoSQL database) is a database that does not follow the design paradigms of relational databases. They are better suited to handle unstructured data, scaling, and flexible models.

## 7. **Open Source Software**

Open source software is software constructed in a way that the source code is readable and modifiable. Software can be constructed to be different degrees of open source (fully open source and partially open source), but all open source software allows users to read the source and modify/change features based upon the license.

## 8. **Permissions**

User permissions refer to the features different users have access to. For example, a developer can access the API, but cannot deny resources from approval. Meanwhile, an admin may not have API access, but will be able to deny or approve resources.

## **9. Progressive Web Application**

A progressive web application (PWA) is an application that uses browser features to act similar to a native app on a platform. Using tools such as service workers, manifests, and caching, PWAs allow fast, reliable, and engaging experiences in web applications

## **10. Resource**

A resource, in context of the application, is a link or reference a user can follow to learn about a selected topic. Resources are a source of information which users can learn from.

## **11. Server**

A server manages the website, delivering the front end code to the browser and handling back end code and operations. Servers provide many services to users and developers such as sharing and modifying data.

## **12. Subject**

A subject is a collection of topics.

## **13. Topic**

A topic is the subject to which a resource discusses.

## **14. User Interface**

User interface refers to the parts of the website the user interacts with. The user interface directly affects the user experience, functionality of the software, and the usability of the software.

## **15. Web Client**

The web client refers to the software users use to interface with websites. This is often a web browser (such as Chrome) but can also sometimes be an operating system or a web crawler. Clients display the application's front end.

**16. Web Site** A web site (or application) refers to a site or application hosted on the world wide web.

# Engineering Standards and/or Technologies

## **1. Django (Technology)**

- Django's REST framework is used for our web-based API. Zachary is creating a full API for use by the web server as well as 3rd party developers. It resides on a virtualized server in Azure, and is the only way edits can be made to the database.



## 2. React (Technology)

- React is an open source JavaScript framework maintained by Facebook. It uses a component based structure for its web pages. React's component based construction allows for easier conditional rendering, state management, and useful features such as higher order components and hooks. Lecture Goggles uses React 16.8 for the web server to create a front end for the users to interact with.

## 3. PostgreSQL (Technology)

- PostgreSQL is the software that powers our database server. It resides on a server in Azure accessible only by the API server, and holds all of the data for the website. The database is broken down into tables which hold all of the information in a categorized and easily searchable format. All API calls that request information are given results from the database server.

## 4. Azure Firewall (Technology)

- Azure's firewall is the central firewall that all traffic must travel through in order to use the website. It ensures that only specific machines have access to the database server, prevents malicious users from attacking the servers using unmonitored ports, and notifies the administrators of suspicious activity.

## 5. Hashed + Salted Passwords (Standard)

- User passwords are not stored, but rather the hashed values and salts are stored in the database, making it almost impossible to reverse engineer a users' password from the data stored on the database server. This prevents malicious actors from successfully breaking into other users' accounts using their passwords.

## 6. Web Content Accessibility Guidelines (WCAG) (Standard)

- WCAG provides a set of standards for accessibility on the web at three levels: A, AA, and AAA, where A is the most useful to achieve and AAA is the most difficult or least useful. Using standard web components and proper component construction allows all users of a website to access the content of web sites. WCAG is maintained by the Accessibility Guidelines Working Group to make websites accessible to all users. Lecture Goggles conforms to all type A accessibility guidelines but only some type AA and AAA.

## 7. ECMAScript (ECMA-262) (Standard)

- ECMAScript is the standard JavaScript is built to conform to. Modern ECMAScript adds additional features such as arrow functions, async/await, promises, and more. Lecture Goggles is coded using modern JavaScript conforming to ECMAScript 6 (or ES6). Since ECMAScript is a

frequently updated language standard, Lecture Goggles uses technologies (such as Babel) to support older versions of ECMAScript.

## Updated List of References

### Book:

- Orr, Dominic, et al. *Open Educational Resources: a Catalyst for Innovation*. OECD, 2015.
  - Orr et al. advocates for the usage and expansion of open education resources (OERs). This work explains how the availability of OERs can help expand the way students learn, allow teachers and students to interact in new ways, and facilitate learning at all levels. This work also explains how OER distributions models can be combined to become sustainable and useful to students.

### Articles:

- McShane, Michael Q. "Open Educational Resources." *Education Next* 17.1 (2017) *ProQuest*. Web. 2 Nov. 2018.
  - This work details the OER initiative and what efforts have been made to make it more available to students and educators. It discusses the efforts of the US Department of Education, and how ineffective its OER repository is. It also discusses the efforts put forward by the Department of Education to work with the many States to develop more OERs available to students and teachers alike.
- Christine L. Ferguson (2017) Open Educational Resources and Institutional Repositories, *Serials Review*, 43:1, 34-38, DOI: 10.1080/00987913.2016.1274219
  - This resource discusses how institutions catalog and retain educational information. It discusses the issues regarding the storage and usage of copyrighted materials. This work also discusses issues regarding versioning in educational repositories. This will be useful in identifying how to deal with potential duplicates or multiple versions of a submitted resource.
- Mouriño-García, Marcos, et al. "Cross-Repository Aggregation of Educational Resources." *Computers & Education*, Pergamon, 5 Oct. 2017, [www.sciencedirect.com/science/article/pii/S036013151730218X](http://www.sciencedirect.com/science/article/pii/S036013151730218X).
  - This work discusses the number of educational resources, their advantages and their drawbacks. These authors propose a system of classifying educational resources using machine learning algorithms and metadata. This method allows for efficient categorization with only minimal

human input or correction. Their system was able to categorize resources hosted on other systems.

#### Websites:

- “200 Free Kids Educational Resources: Video Lessons, Apps, Books, Websites & More.” *Open Culture*, [www.openculture.com/free\\_k-12\\_educational\\_resources](http://www.openculture.com/free_k-12_educational_resources).
  - This website has an enormous list of hyperlinks to free educational resources all around the internet. Though they are listed as K-12 resources, many remedial or introductory level college courses cover similar or identical content, making them excellent resources to use with Lecture Goggles
- Khan, Salman. “Khan Academy.” *Khan Academy*, Khan Academy, [www.khanacademy.org/](http://www.khanacademy.org/).
  - Khan Academy is an excellent resource for many topics, but especially math and science based topics. Khan Academy does not require an account to use their resources, so it is easy to locate them using hyperlinks. These videos, worksheets, and interactive problems will be excellent initial resources to add to Lecture Goggles.
- Fox, Carolyn. “A Guide to Free and Open Source Education.” *Opensource.com*, 2013, [opensource.com/education/13/4/guide-open-source-education](http://opensource.com/education/13/4/guide-open-source-education).
  - This page points to multiple large databases containing thousands of educational videos and courses. Some of the resources mentioned are designed to work in a similar way to Lecture Goggles, which will help us narrow down the specific functionalities that are well received, and which ones are not. The videos will be excellent to add to our initial release of Lecture Goggles.

## Contribution of Team Members

**Zachary Johnson** did the Abstract, Recent Project Changes, and Technical Requirements Specification. Zachary worked on the project for approximately 3 hours.

**Logan Long** did the Technical Requirements Specification, Requirement Traceability Matrix, Engineering Standards, and List of References. He also assisted in the Use Cases. Logan worked on the project for approximately 3 hours.

**Nathan Yocum** did the User Interface, Glossary, and added to the Technical Requirements Specification and Engineering Standards and Technologies. Nathan

worked on the report for about 4 hours and on the code for the user interface for about 15 hours.

All three members were involved with checking each other's work and removing errors.