

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

November 19, 2018

Table of Contents

Table of Contents	1
Abstract	2
Introduction	2
High-level and medium-level design	3
System level diagram	3
Program Units	4
Web Server	4
API Server	6
Data Structures	7
Account Info	7
Submissions	8
Subscriptions	8
Votes	9
Reports	9
Detailed Design	10
Login State Diagram	10
Admin Report Handling Activity Diagram	11
Create an Account Activity Diagram	12
User Interface Design	12
Glossary Updates	23
Contributions of Team Members	26

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.

Introduction

Lecture Goggles is a web application that is built to store and share helpful resources. Resources can be HTTPS links or files provided by students or teachers. This will be limit file sizes to an appropriate size which has not been determined yet. To provide a safe environment for all users, each link will be checked for malicious content or inappropriate HTTPS forwards. The first implementation will limit resource contribution to authorized college students and college professor accounts. To authorize accounts, the application will use third-party SheerID software to get access to 180+ million student verified email addresses from the US and around the world.

The UI design behind Lecture Goggles is an expanding tree method. By using this method of design Lecture Goggles can continue to grow with little administration interaction. As users add new subjects and resources to the website, the website will grow to contain all academic subjects needed. Each resource link will afford a system to upvote and downvote. As demonstrated in websites like reddit.com, this seems highly successful at increasing visibility to useful additions.

Due to the size, scope, and dynamic capability of Lecture Goggles, there will need to be a web server to communicate with each users' requests. Django is a web framework that can be used as an intermediary between the user requests and database. Although Django can also be the web server for Lecture Goggles, for easier deployment and scalability Team 22 will be using Apache Web Servers to contain Django.

To contain and organize all the user account data, provided resources, and any other necessary data, a database will be needed. MongoDB, which is classified as a NoSQL

database program, was previously going to be used for initial implementation. Because Django does not support NoSQL out of the box, our group will need to use a PostgreSQL database instead. Django web framework will access the database directly when user requests are received then communicate pertinent information back to the user.

High-level and medium-level design

System level diagram

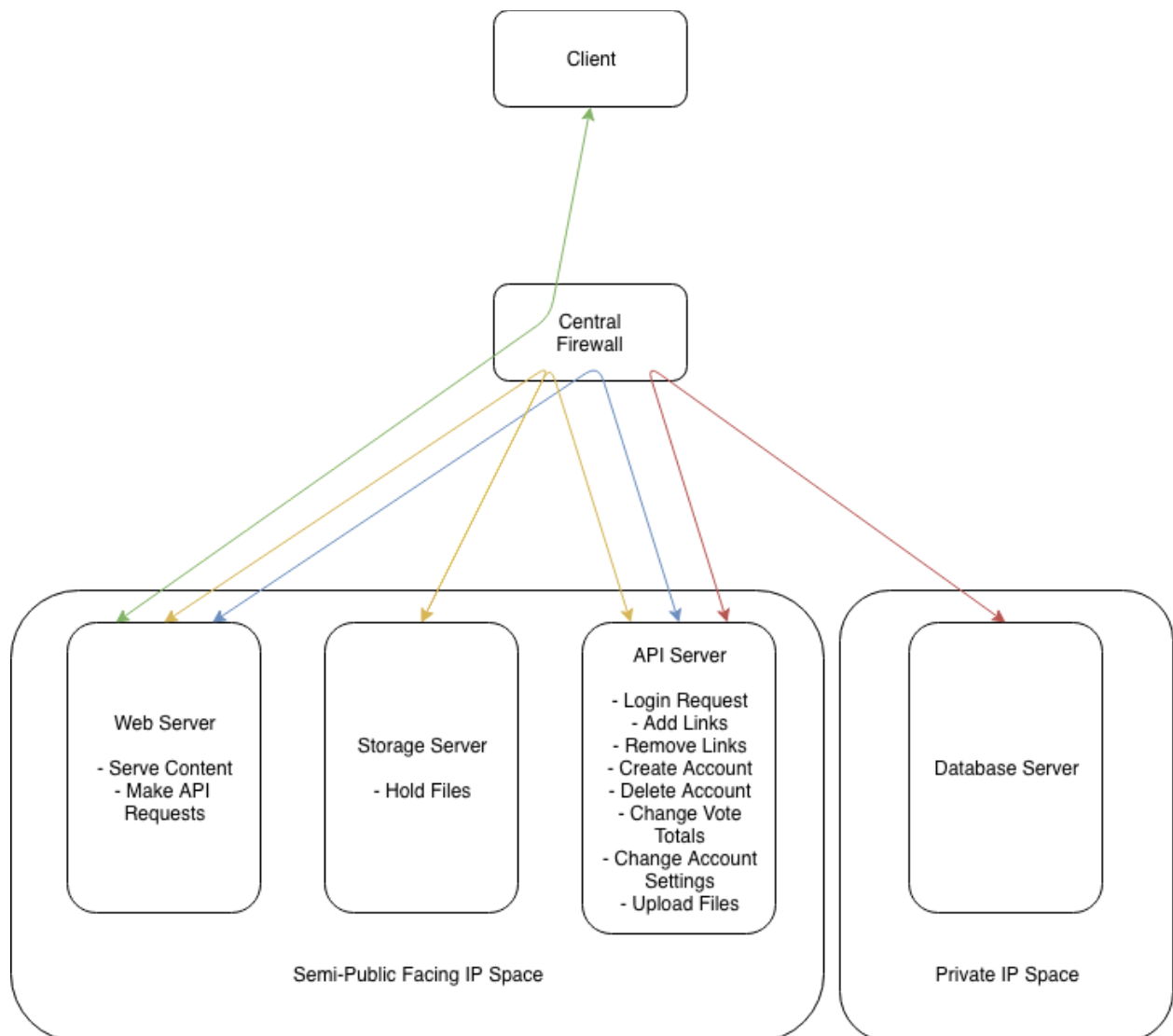


Figure 1: Lecture Goggles High-Level Architecture

Figure 1 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 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.		
Instructor	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.		

Admin	Inherits from the User class. Has the most permissions and can add or remove submissions at will. Can suspend/ban users as needed.		DeleteSubmission() BanUser() UnbanUser() LockUser() UnlockUser() [Set/Get]AccountType() ApproveUser() 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 private key to execute all	ListeningPort	UpdateSubmission()) AddSubmission() AddUser() DeleteUser() AddReport() RemoveReport() EditUser() AddVote() RemoveVote()

	commands.		
Database	Handles all communication to and from the database server residing in private IP space. Uses a separate public-private key pair to communicate with the database server.	ListeningPort	UpdateSubmission()) AddSubmission() AddUser() DeleteUser() AddReport() 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

Key	Value
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

Key	Value
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

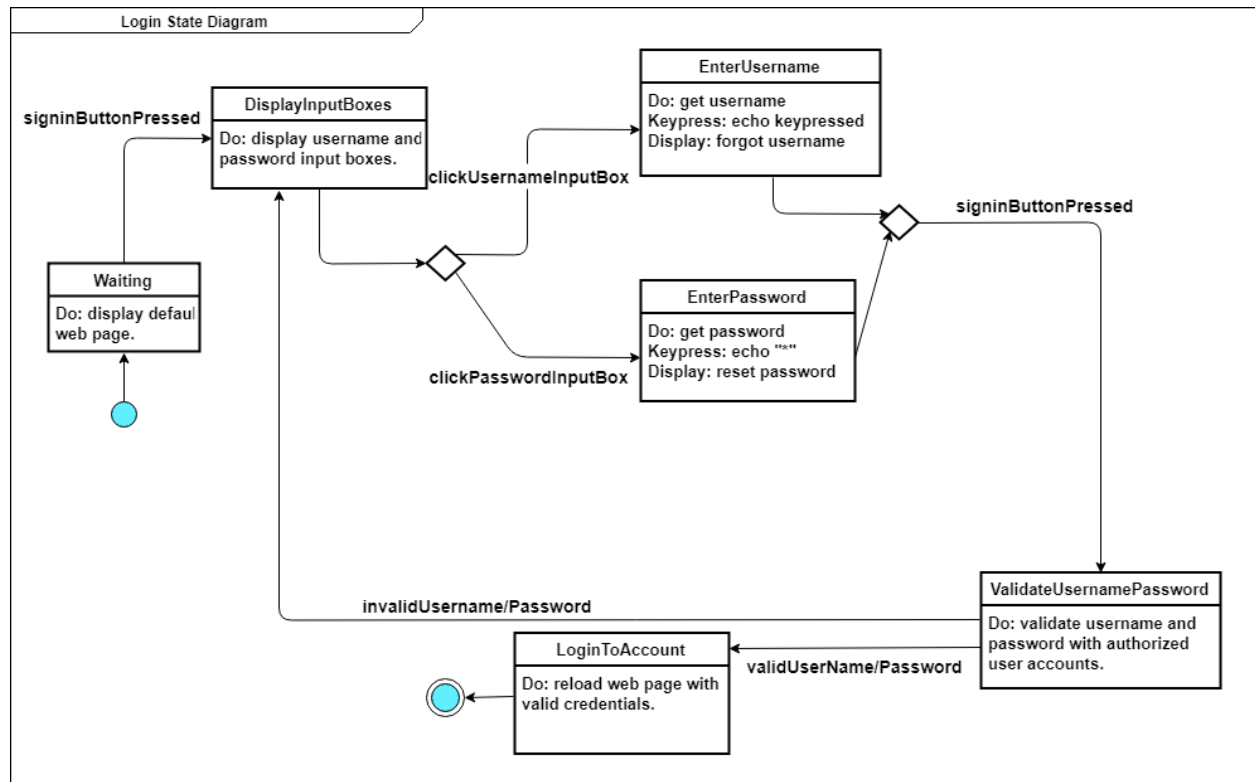


Fig 2. Login State Diagram which demonstrates a flow of events and actions for the login process.

Shown in Fig 2. 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

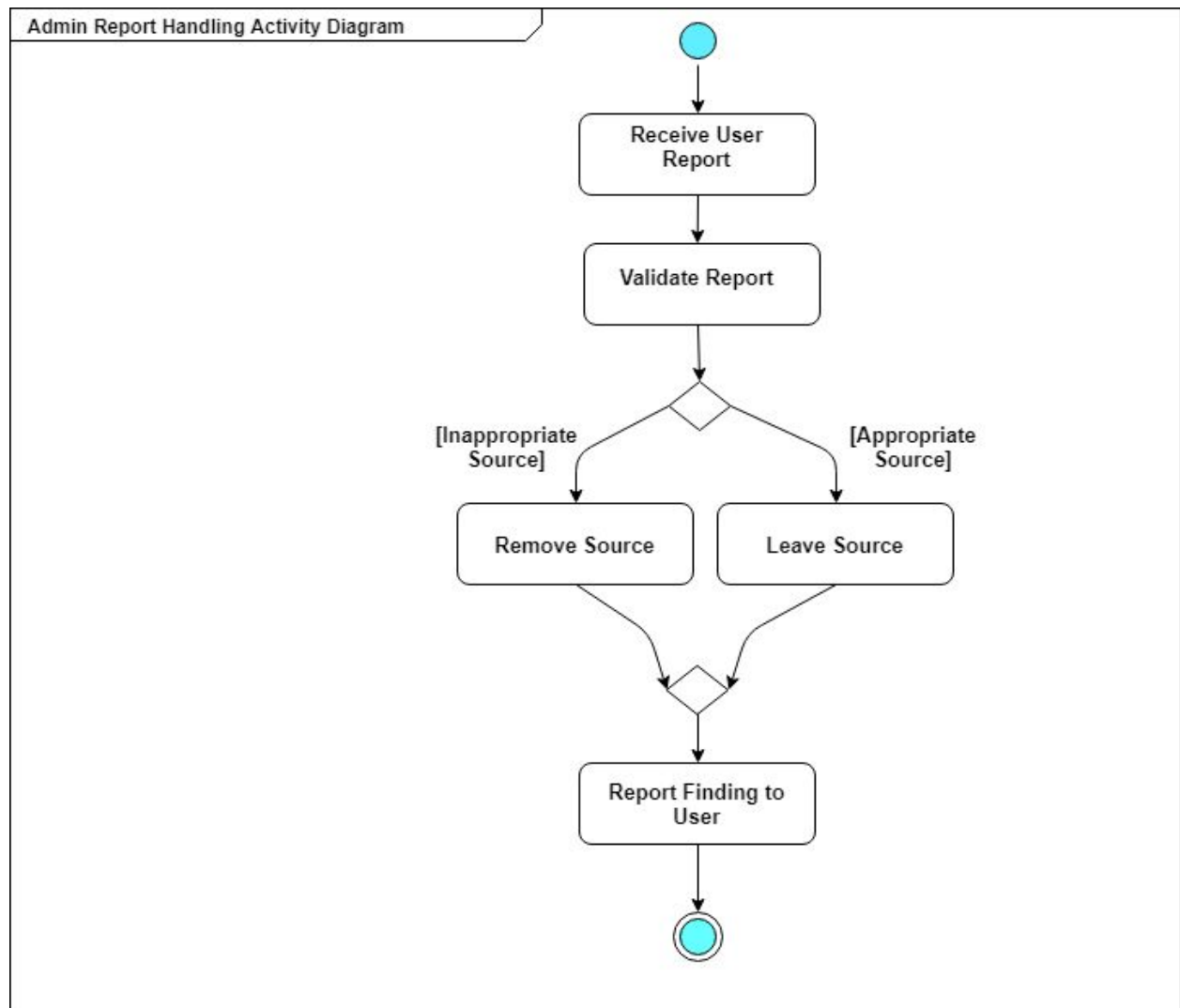


Fig 3. Admin Report Handling Activity Diagram which demonstrates the process in which inappropriate content will be evaluated.

Shown in Fig 3. 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

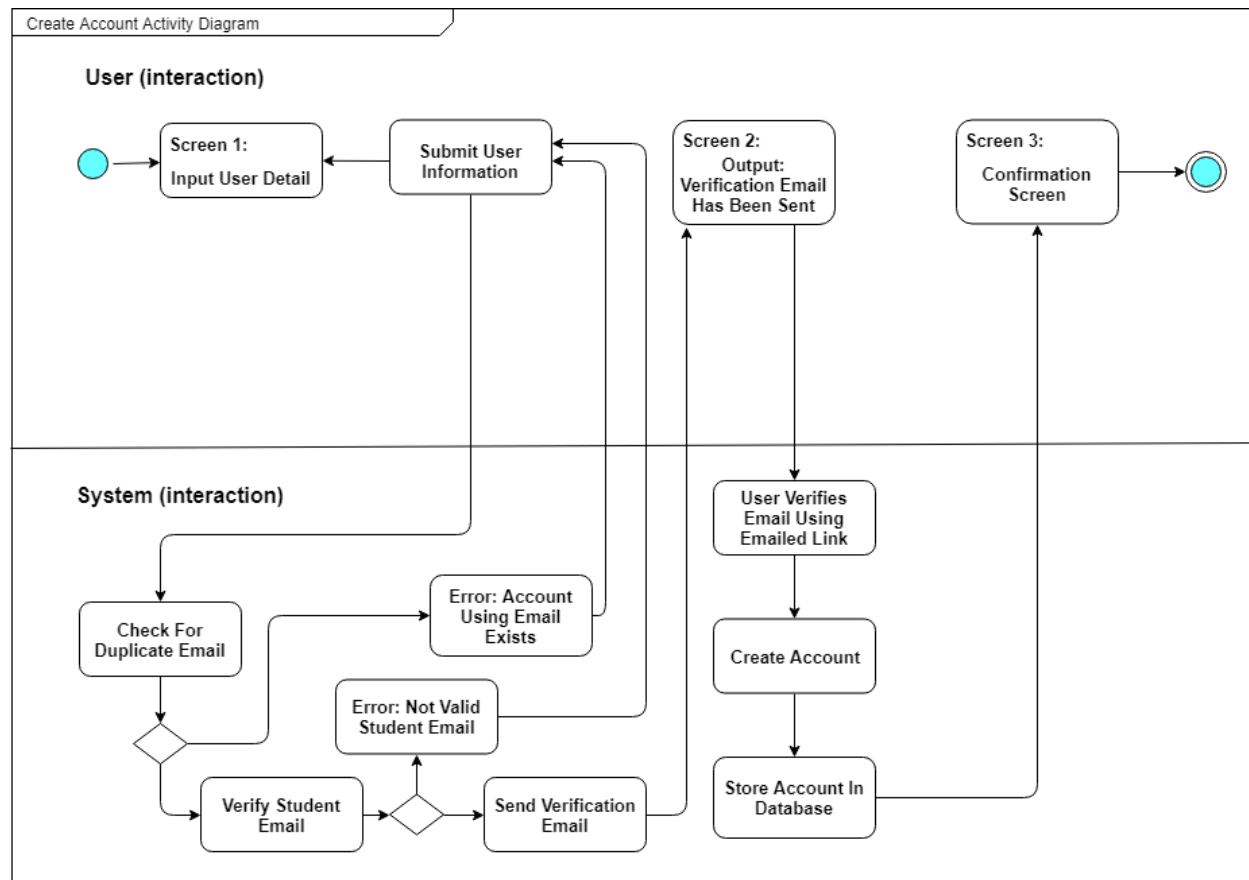
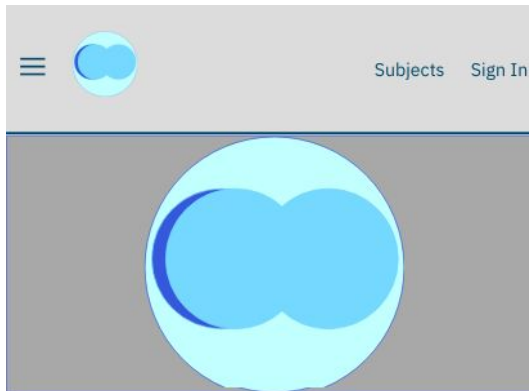


Fig 4. Create an Account Activity Diagram using system and user interaction sections.

Shown in Fig 4. 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.

User Interface Design

For the design of the user interface for this project, Team 22 took the wireframes from Project 2, and began implementing them in React as a nonfunctional prototype. Lecture Goggles is a web application designed to be used on mobile devices, tablets, computers, and anything else a web browser can run on. This is accomplished through a responsive, mobile first design.

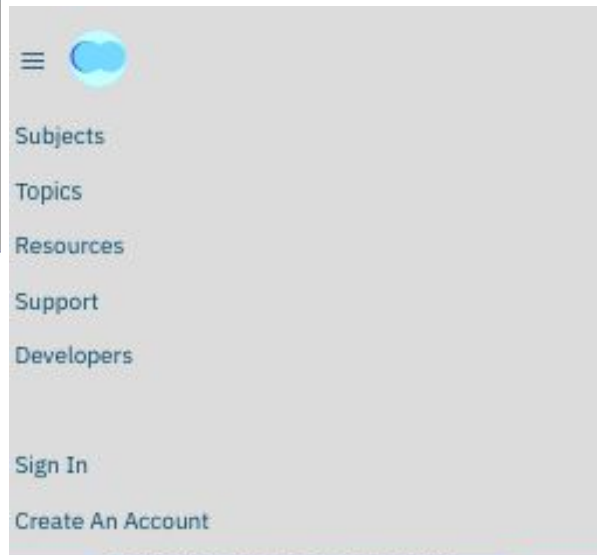


Lecture Goggles



Sign In

Create An
Account




Lecture Goggles!


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

Figure 5: Landing Page (while not signed in) with hamburger menu on mobile for navigation.

[Subjects](#) [Topics](#) [Resources](#) [Support](#) [Developers](#) [Sign In](#) [Create an Account](#)



Lecture Goggles

First Name

Last Name

Email

Confirm Email

Password

Confirm Password

Institution

Are you an instructor at your institution?

☐ Yes ☐ No

Continue

Cancel

Figure 6: Interface for signing up for an account.

The screenshot shows a web interface for 'Lecture Goggles'. At the top, there is a navigation bar with a hamburger menu icon, a logo, and links for 'Subjects' and 'Sign In'. Below the navigation bar is a large circular logo. The main content area is titled 'Lecture Goggles'. A red error message box is displayed, stating: 'ERROR: Incorrect data in: First Name, Last Name, Email, Confirm Email, Password, Confirm Password, and Institution'. Below the error message is a light blue form containing several input fields: 'First Name', 'Last Name', 'Email', 'Confirm Email', 'Password', 'Confirm Password', and 'Institution'. Each input field has a red border, indicating it is required. At the bottom of the form, there is a question: 'Are you an instructor at your institution?' with two radio buttons, 'Yes' and 'No'. Below the form are two buttons: 'Continue' and 'Cancel'.

Subjects Sign In

Lecture Goggles

ERROR: Incorrect data in: First Name, Last Name, Email, Confirm Email, Password, Confirm Password, and Institution

First Name
[]

Last Name
[]

Email
[]

Confirm Email
[]

Password
[]


Confirm Password
[]

Institution
[]

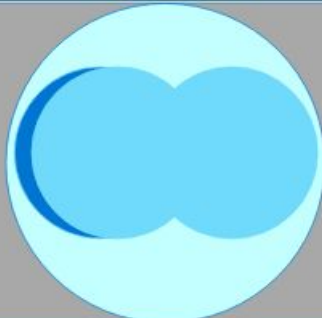
Are you an instructor at your institution?
☐ Yes ☐ No

Continue Cancel

Figure 6.1: Interface for wrong information provided when creating an account.



Subjects Topics Resources Support Developers Sign In Create an Account



Lecture Goggles

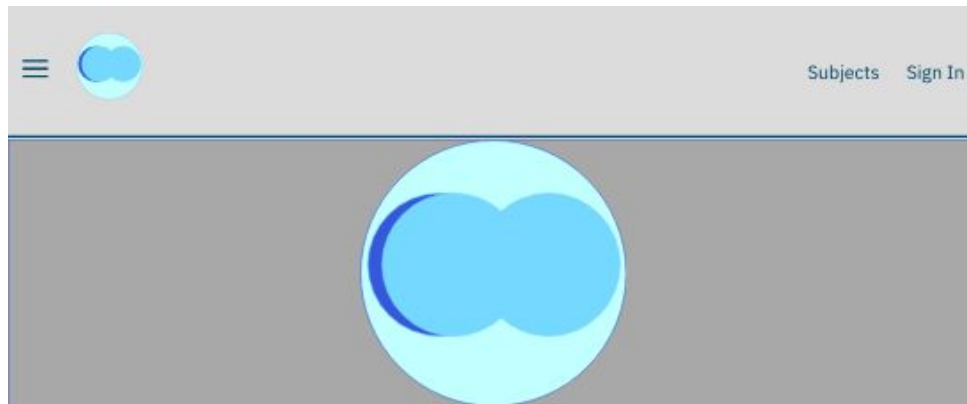
Sign In

Email

Password

[Forgot your password?](#)

Figure 7: Signing into an account.



Lecture Goggles

ERROR: The email or password you provided was incorrect.

Sign In

Email

Password

[Forgot your password?](#)

Figure 7.1: Error displayed when incorrect password or username is used.

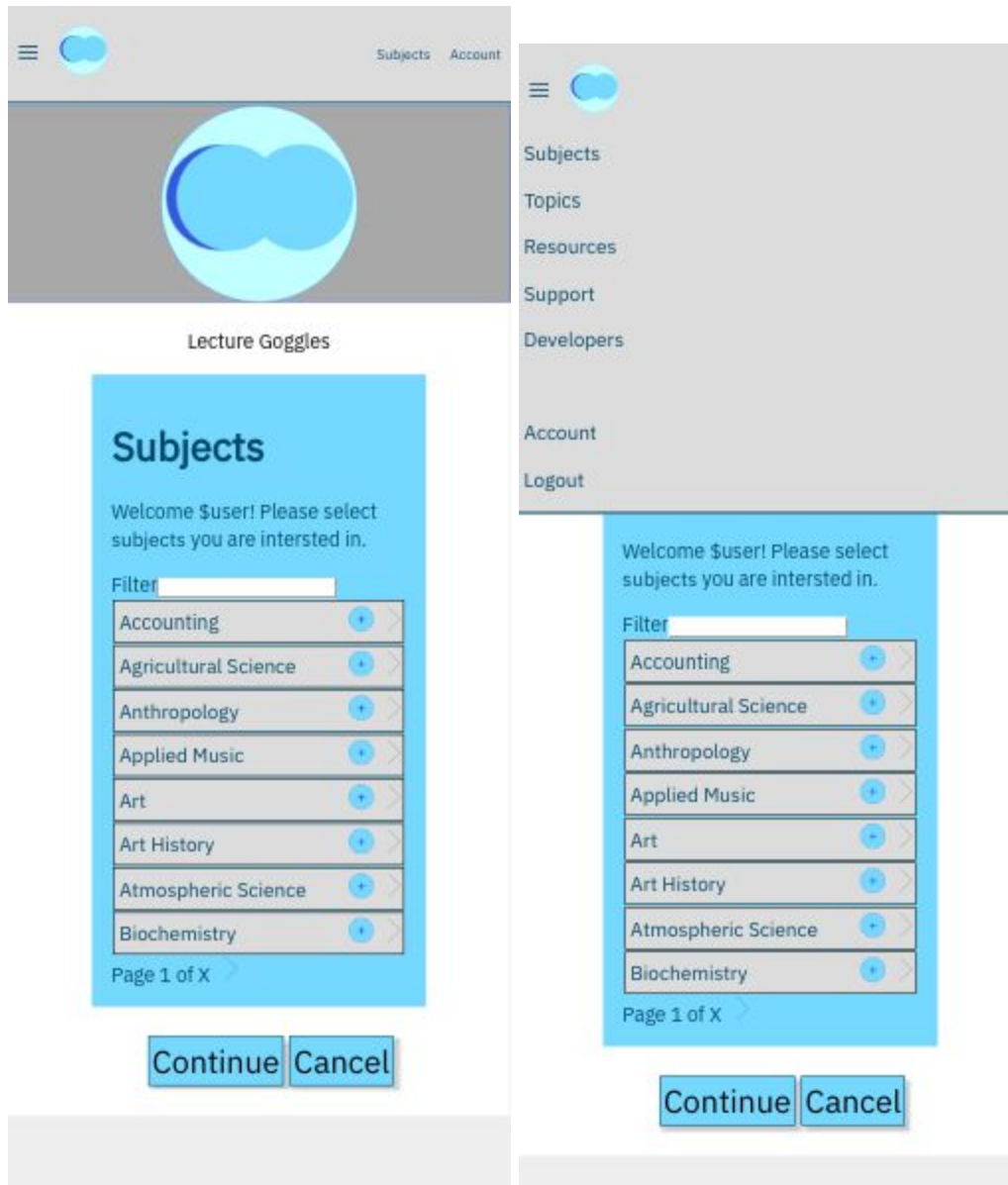


Figure 8: Browsing by subject with updated menu after sign in



Lecture Goggles

Topics

Welcome \$user! Please select Topics you are interested in.

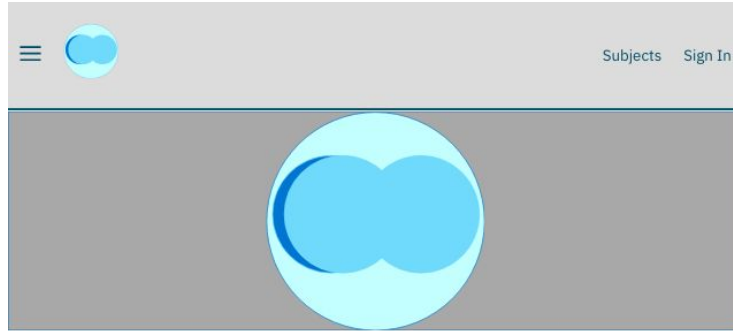
Filter

Topic: Assets Subject: Accounting	<input type="checkbox"/>
Topic: Lorem Subject: Accounting	<input type="checkbox"/>
Topic: Lorem Subject: Accounting	<input type="checkbox"/>
Topic: Lorem Subject: Accounting	<input type="checkbox"/>
Topic: Lorem Subject: Accounting	<input type="checkbox"/>
Topic: Lorem Subject: Accounting	<input type="checkbox"/>

Page 1 of X [>](#)

[Continue](#) [Cancel](#)

Figure 9: Browsing topics for a selected subject.



Lecture Goggles

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.

Getting an API account

Any registered user with a verified email can request an API key. Follow [this](#) link to register for a key.

Read the documentation

...

API Methods

...

Sample applications

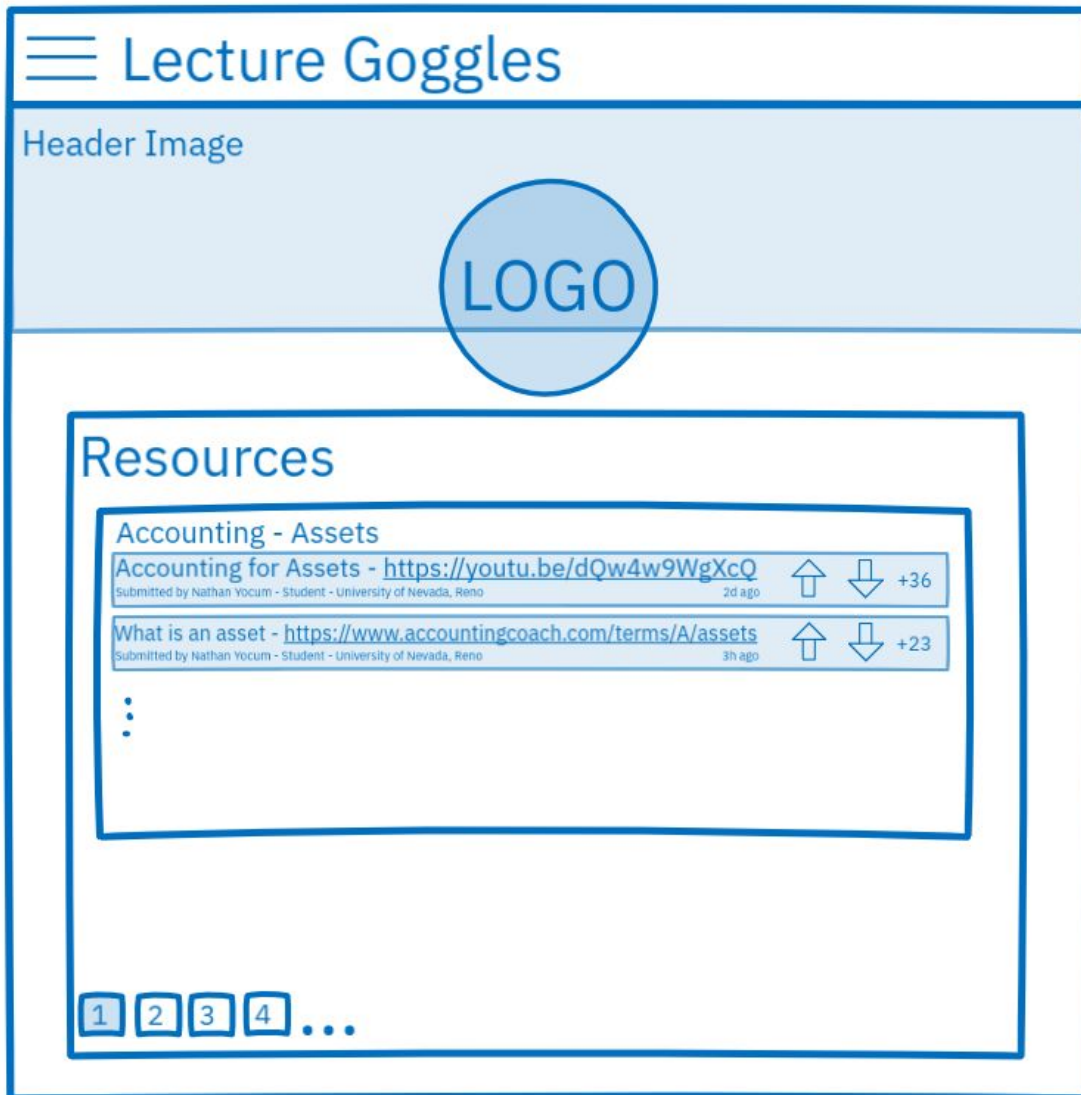
[Resource Randomizer](#)

...

Source Code

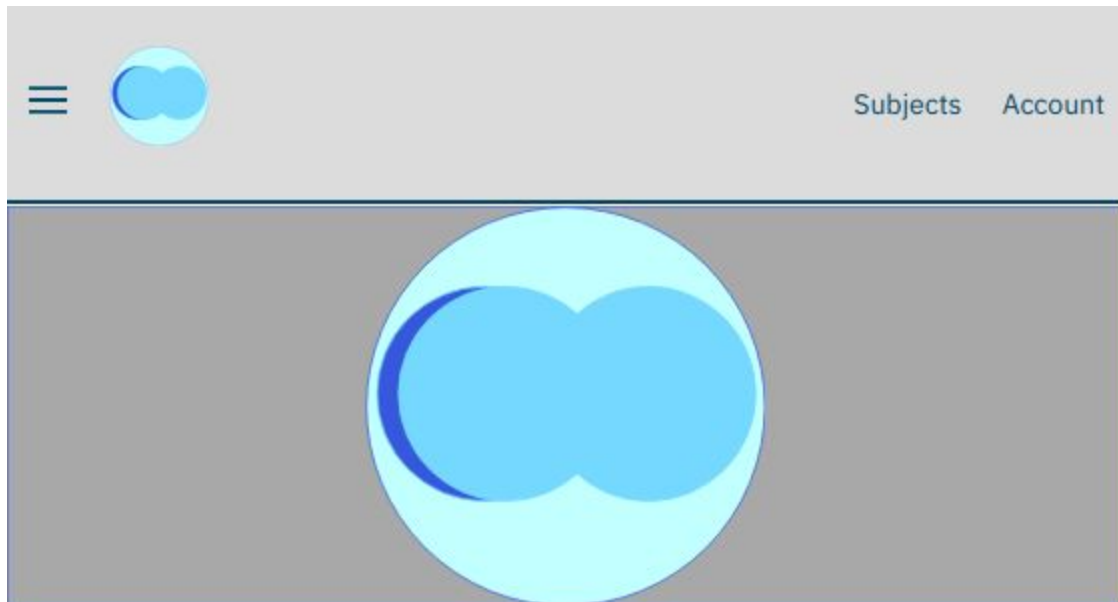
Lecture Goggles is a fully open source applicaion. Read about how to contribute to the [frontend](#) or the [backend](#) of the website. Lecture Goggles is licensed under an [MIT](#) licence.

Figure 10: Developer page where users of the application can view information about the API, request a key, and read about the source code and documentation of the website.



Viewing Resources

Figure 11: Wireframe of viewing resources when viewing topic



Lecture Goggles

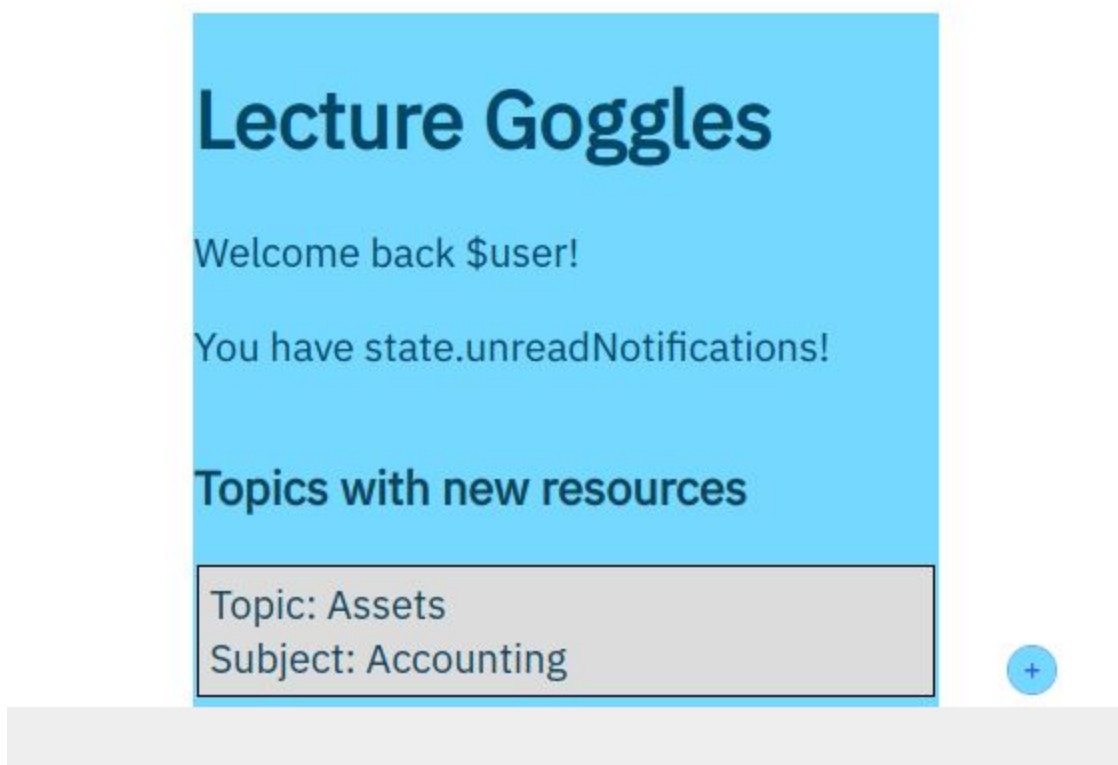


Figure 12: Home screen signed in

Glossary Updates

Updates highlighted

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. **Application Program Interface (API) Key**

A developer can obtain an API key to interface directly with a website. API keys can be public or secret. Secret keys usually allow users to authenticate directly with their account, while public keys usually allow users to access public data for the website.

3. **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.

4. **Cascading Style Sheets (CSS)**

A browser uses CSS to style pages and components in a web page.

5. **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.

6. **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.

7. **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.

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.

8. **Hypertext Markup Language (HTML)**

HTML is a markup language used to organize web pages.

9. **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.

10. 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.

11. 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.

12. React

React is a frontend framework which uses components to construct web pages. It utilizes JavaScript, HTML, and CSS often in one file.

13. Relational Database

A relational database organizes data in the form of tables, columns, and rows. Data in a relational database is structured so that items are identified by keys and relate with each other.

14. 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.

15. Salt

In order to store a password, a password must be hashed to protect users in the case of a database breach. A pure hash will still make common passwords easy to identify, causing passwords to leak. The solution to this is to insert a randomly generated string to the hash, making the hashed password unidentifiable.

16. 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.

17. State

React applications utilize state to display data it has. State is immutable to components other than by the component itself. State is a way for components to keep track of the data they maintain.

18. Subject

A subject is a collection of topics.

19. Topic

A topic is the subject to which a resource discusses.

20. 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.

21. 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.

22. Web Site

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

Contributions of Team Members

Zachary Johnson did the Abstract, Introduction, Detailed Design. Zachary worked on the project for approximately 11 hours.

Logan Long did the High-Level and Medium-Level design. Logan worked on the project for approximately 12 hours.

Nathan Yocum did the User Interface Design and Glossary Updates. Nathan worked on the project for approximately 12 hours.

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