CS4624 Projects Discovery Portal

Final Report
Spring 2024

Prepared by
Shayne Perryman
Adam Kanaan
Mark Fuentes
Maddie Gonzalez
Faizan Moshin

Client:
Dr. Mohammad Farag

CS4624 Multimedia, Hypertext, and Information Access
Virginia Tech, Blacksburg, VA 24061
May 1, 2024

Table of Contents

This page is intentionally blank

Abstract

CS4624 is one of many capstone courses a student within the CS curriculum is able to take. The multimedia and hypertext course digs into the diverse range of multimedia content such as images, audio, video, and any information retrieval and access relating to it. With this comes the capstone project which is a semester long project given to us students to allow a display of mastery within our discipline. It has been a pleasure to have Dr. Farag and Vedant Shah guide and assist us with the project. An insight into real-world applications as well as a diverse approach to different problems has allowed us to grow as both people and developers.

The current discovery portal for CS4624 student projects serves as a platform for students working within the course to submit and hold their projects. Details included within the pages on the discovery portal consist of abstract, date, author, presentation, final report, source code, and collections. Additionally, the discovery portal contains filtering features to allow users to specifically search for any project dependent on; recent, issue date, author, title, subject, content type, and department. This allows teachers to easily access desired projects as well as a safe holding for semester long projects that students worked hard on.

With this comes the purpose of the project. After reviewing the functionality and appearance of the discovery portal, there were many things that needed to be improved on. The first very noticeable issue was that the search features did not either function properly or at all. For starters, the ‘By Issue Date’ filter would not allow filtering with either month or year. It was required that both were specified along with the open search requiring an exact formatting of date (xx-xx-xxxx). The ‘By Author’ filter expected that the full name was typed out, including the comma separating the first and last name. All open search features expected a prompt that would produce an exact match. It also came to our attention that some features produced over a thousand categories for only 269 projects. This should never be the case as the purpose of filtering is to narrow the search for projects.

After analyzing the discovery portal, our main focus became improving upon the search and filtering features. This was something that would require us to completely recreate the discovery portal due to existing source code being unavailable. With this, we first needed to create a front-end and back-end that would relay any information requests to have a web page display. To replicate the discovery portal further, we also implemented an authentication aspect. Accounts would be divided into ‘admin’, ‘professor’, and ‘user’, each having distinct permissions on what they are able to insert, delete, and modify. Following this was the development of our database to store all of the projects’ files as well as a schema that the search and filtering features would utilize. Finally, we implemented a search api for the back-end to access, a completed schema for each project, and created a function search and filter function. Upon completion, we hope to provide future CS4624 students and staff with a more convenient tool to guide them in their journey of completing their capstone projects.

Introduction

Project Description

This project was done in fulfilling a requirement for the CS 4624 Capstone Course. The Team is tasked with creating a web application that displays information and files of all previous projects that were originally uploaded to the VTechWorks website under the tag Multimedia, Hypertext & Information Access. This project is to be usable on all OS types and support user authentication and differing permission levels and accesses based on said authentication. Furthermore the web application should allow for users to search the database for projects based on specified search criteria. It will also allow users of higher permissions to create and upload projects to the database based on a form that requests information and files regarding the project.

Motivation

We are tasked with renovating the current VTechWorks site where projects from previous sessions of CS4624 are stored. Our motivation comes from wanting to work on a project that would allow us to leave our mark behind on Virginia Tech once graduated. We will be building a web app that shows the relevant information of each project in a user-friendly interface. Our interface will allow users to search for relevant projects and will enable browsing for projects using different categories. Furthermore, we will develop a feature where users can submit new project ideas for future CS4624 sessions. Furthermore, our interface will provide user authentication, and we will develop separate views and features for admins and students. Admins will have the ability to publish new projects, view all projects from previous semesters, and delete project pages. Our team members all have unique backgrounds and experiences and we are splitting roles between developing the front and back end of the project.

Project Requirements

User Authentication

After talking with the client we learned that our project needed to be able to cater to three different levels of permissions and these permissions would be based on the user authentication. The authentication had to be secure and not make use of any http requests that would show the confidential information (i.e. usernames and passwords) of the users.

Differing Permissions

The project needs to handle different permission levels that are associated with who is currently logged in to the web application. The project should be able to support three different user permission levels: Administrator, Client, and Student.

The Student level is the most basic and restricted of all the permission levels as those with student level access can only see the project that they have been a part of in the past or participating in at this current moment. They can also see any projects for the current term and have the ability to join a maximum of one project. If they're currently a part of a project, they also have the ability to leave the project and join another if they so choose.

The Client level shares all of the same permissions as the Student level in regard to viewing and interacting with projects that the user is a party to, but it also has the added feature of allowing the creation of a project in the database. A client is also able to be a part of multiple projects.

The Administrator level, like the Client level, has all prior permissions associated with the previous permission levels but they also possess the ability to delete current projects that exist in the database, not just those that they are a party to.

Storing Data in Database

At the core functionality of our project, we have to store all previous term project submissions from VTechworks into our database. We have decided to use MongoDB to store all of the project information, including title, authors, abstract, key words, date, publisher. In addition to this information, we must also store certain files for each project. These files will be the source code for each project, the final report, and the presentation powerpoint. Storing this information will allow for other functionalities within the web application to access the previous projects.

Searching Database

One of our project's priorities is to resolve the issues with VTechWorks current search functionality. The current site has all the appropriate categories for searching through the project directories, however the results are often not accurate. For example, the “Recent Submissions”, “By Issue Date”, and “By Department” categories do not provide accurate results. We will be keeping all of the existing search categories currently on VTechWorks and will be resolving all of the current bugs. Specifically, we will be using a JavaScript search API for querying the database for specifics on names, dates, and authors. When users run a search on the frontend with a given category, the query will be processed by the API depending on what’s needed. It’ll send back json data given the search requirement, whether that be just a name or even entire project details. This API also has the ability to search by multiple types of queries. For example, this is how projects from a certain semester are searched for since we need both month and year to determine which semester.

Project Proposal

An essential feature of our system is to establish future use of the tool by allowing clients to submit new project ideas. On both the client view and admin view, we will implement a button that opens up a fillable form with fields necessary for new project ideas. The system will then use these fields to populate a “Project Ideas” collection in our database. This database will then allow for students to view available projects at the beginning of their term and sign up for ideas that interest them. The purpose of this feature is to improve the current process for new projects which are currently hosted on Canvas, and instead allow for everything to be done on one cohesive web application.

Design

When beginning to think about the design of our system, we had to consider the differences in views for our three permission levels. First, we chose to start with our admin view because it is the most feature-rich view out of the two: Admins View and Student View. The admin must be able to view and search through all past projects, submit new project proposals, view client-submitted project proposals, and view current projects active for student sign up. Our thought process behind this choice was that many of the admin features will also be visible in the client and student view, so starting with the overarching view would allow us an easier implementation when moving onto the other views.

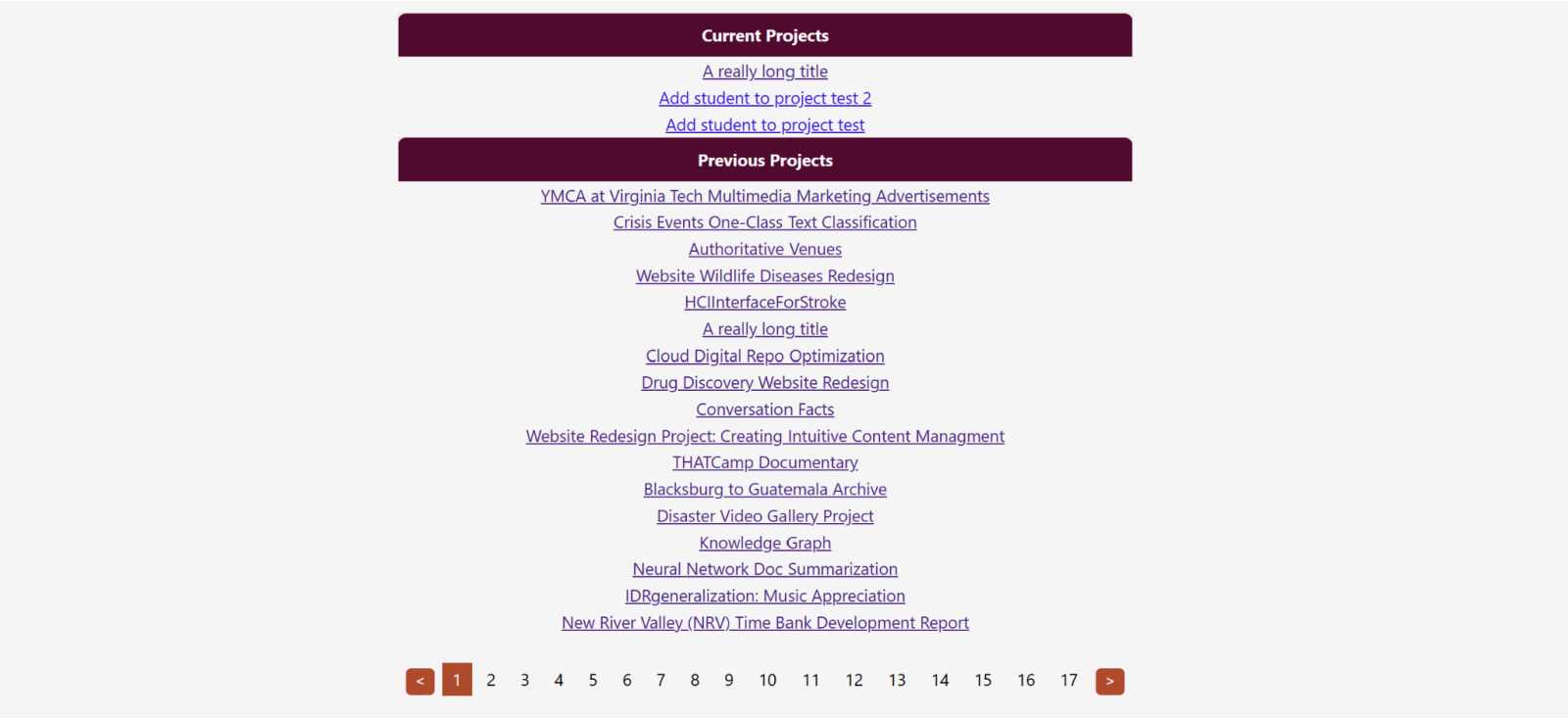


Figure 1

Figure 1 displays the initial view with the list of projects that the admin user would see when first accessing our system. The table shows each project title in the collection of previous terms CS 4624 projects which are stored in our MongoDB database. The table also shows the current projects in that term.

Figure 2

We have a “Filter Collections” button on top of the screen, which adds functionality to search through all of the projects more efficiently. The user may also search for projects manually using the search bar at the top of the screen. Currently, each project title in the collection has a clickable link that opens the following page when a user clicks it.

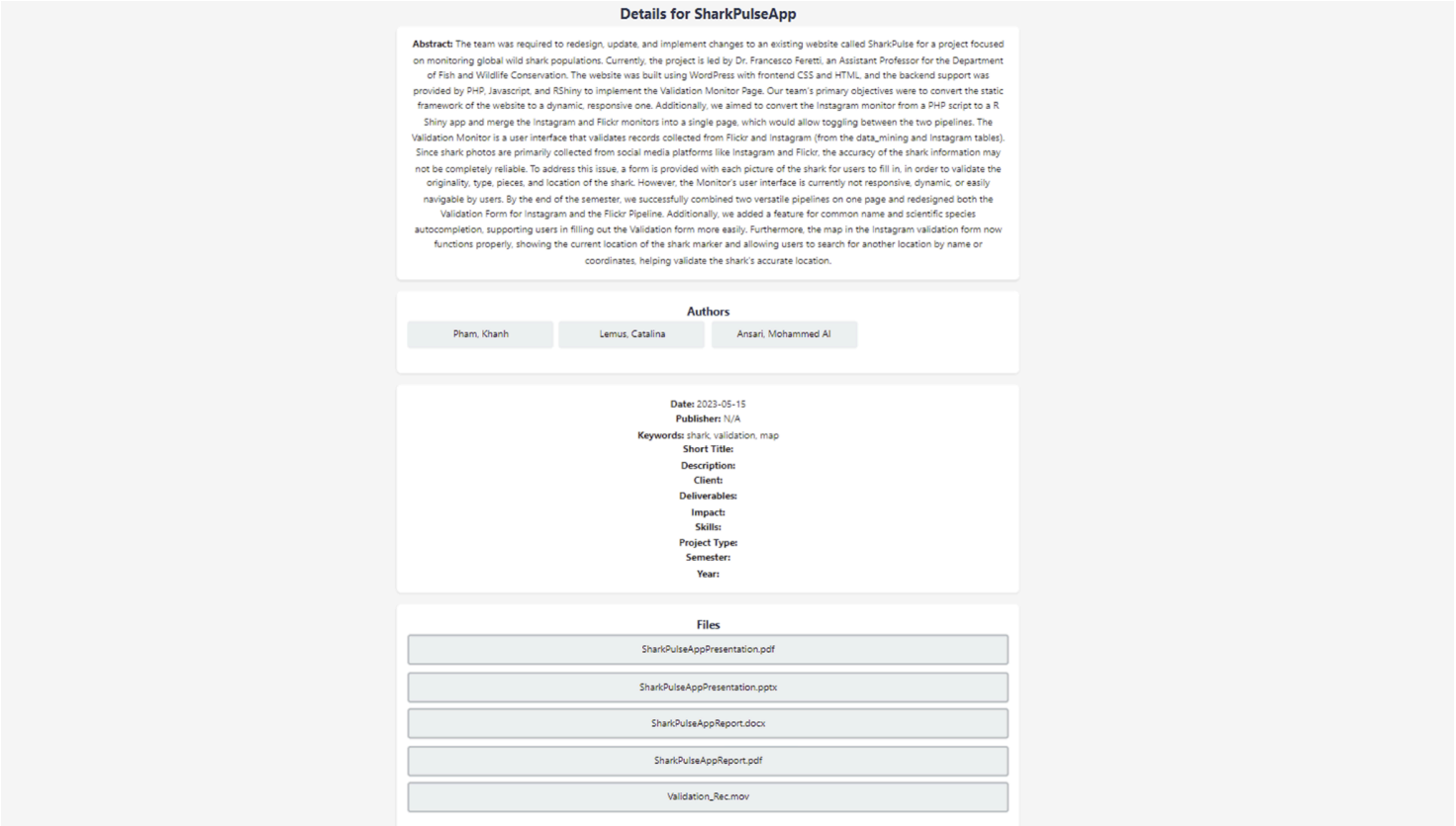


Figure 1.1

Each project link clicked on by the user opens up this details page including the authors, abstract, date, keywords, and publisher for each specific project. The view also includes a list of files for each project, which the admin can click to download locally.

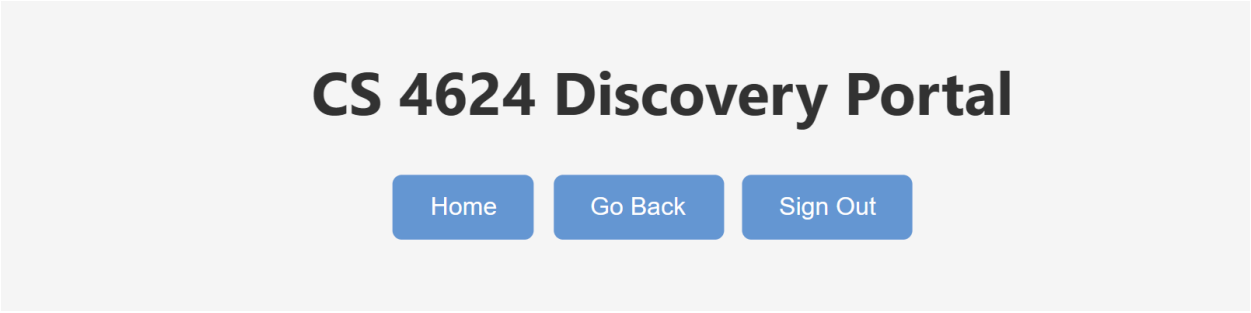


Figure 1.2

The “Home” button currently brings the user back to the initial page with the project collections table. The “Go Back” button brings users to the previous page that they were on before navigating away. The Sign Out button allows users to sign into their account and return to the homepage.

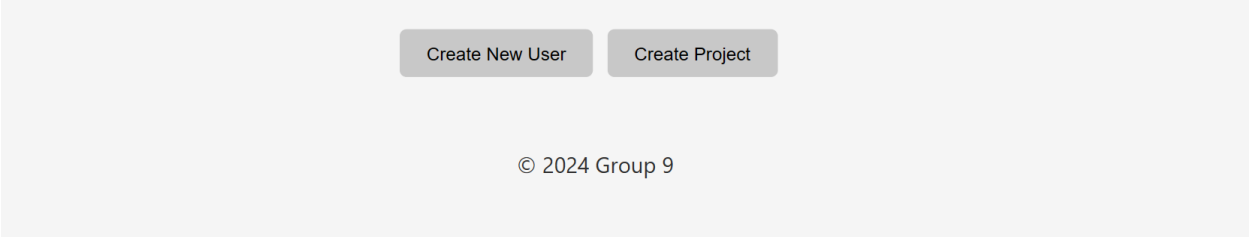


Figure 1.3

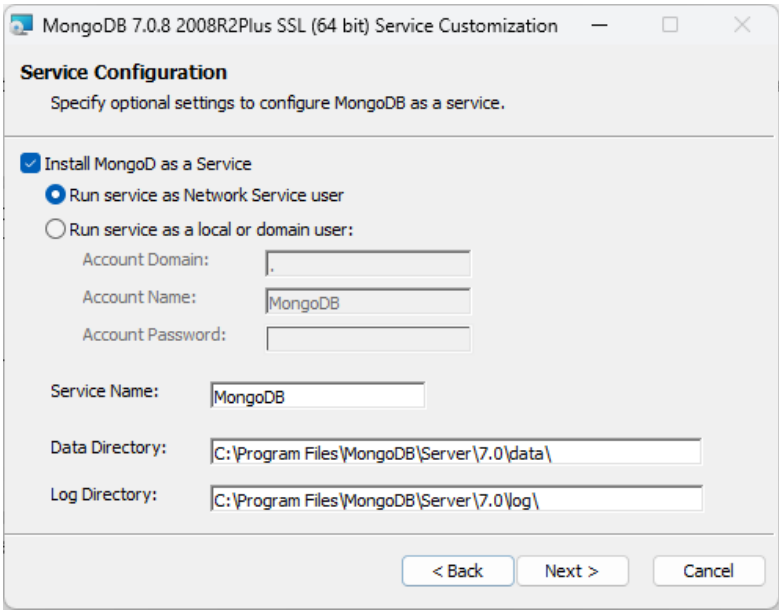
The footer of the page has options for the admin to create a new user and to create a new project.

Developers Manual

Prerequisite Installs/Softwares

(Note: Please install these prerequisites in the given order. Also these instructions were done on the Windows OS so any console commands may differ if you are using a Mac/Linux OS.)

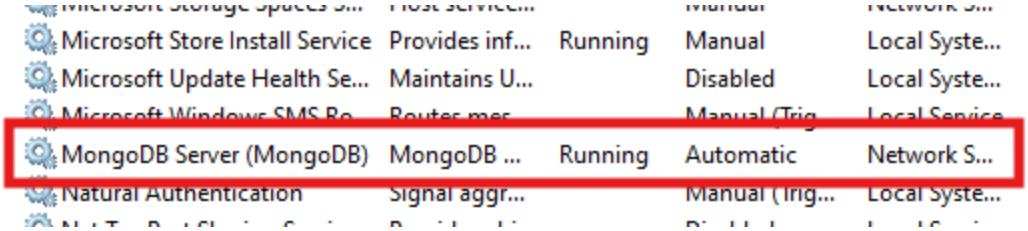
1. Install/Setup MongoDB Community Server
   1.1. Go to the following site: https://www.mongodb.com/try/download/community
   1.2. On the onscreen wizard make sure to select
      - The most current version (at the time of this documentation its 7.0.8)
      - Your systems OS
      - .msi
   1.3. Once downloaded follow the installation wizard
      - Accept the terms of service
      - From set up type choose “Complete”
      - In service configuration make sure the following are checked:



      - (Optional but highly encouraged) Keep the box in the corner checked to install Mongo Compass.
        Its Mongos local UI interface and will be used in later images
      - Finish installation, this may take a few minutes.

1.4. Verify its installation

- Click on the windows icon in the bottom left corner
- Type “Services” and an icon with two gears should show
- Scroll down till you reach the “M” and you should see MongoDB Server as one of the services



1.5.

2. Install Mongo Tools

2.1. First install the MongoDB Shell (https://www.mongodb.com/try/download/shell)

- The most current version (at the time of this documentation its 2.2.4)
- Your systems OS
- .msi

2.2. Follow the installation instructions for your specific OS (https://www.mongodb.com/docs/mongodb-shell/install/)

2.3. Install MongoDB Command Line Database Tools (https://www.mongodb.com/try/download/database-tools) following the onscreen wizard

- The most current version (at the time of this documentation its 100.9.4)
- Your systems OS
- .msi

(NOTE: Be sure to pay attention to where the tool was installed this will be important in the next step)

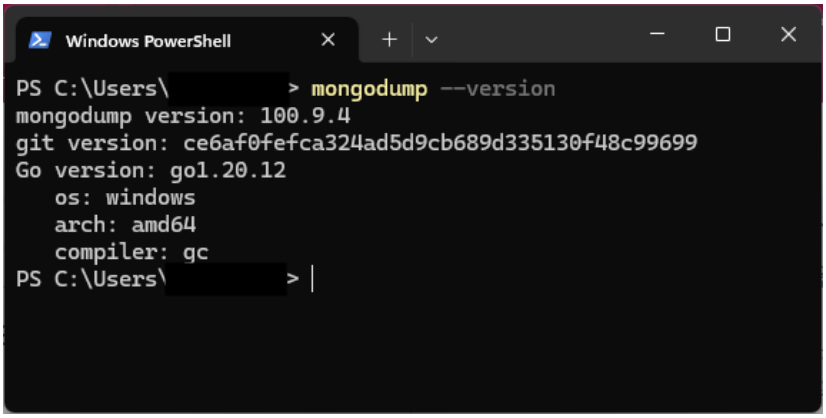
2.4. Follow the installation instructions for your specific OS (https://www.mongodb.com/docs/database-tools/installation/installation/)

2.5. Verify it was installed correctly:

- Run the following command on the command terminal

```
mongodump --version
```

This command is the same across all OS's and should display a similar result to the following image:



3. Downloading and Installing Node.js

3.1. Download Node.js (https://nodejs.org/en/download)
following the onscreen wizard

- Choose the current version (at the time of this documentation its 21.7.1)
- Choose your systems OS
- Choose your system architecture
- Click "Download" for the version based on above specs

3.2. Install Node.js

- Run the installation wizard
  (NOTE: the default installation settings that the wizard recommends is fine for implementing this system)

3.3. Verify Installation

- Open the command prompt and enter the following commands
  node -v
  npm -v

This should give you the following versions
v21.7.1 for node
10.5.0 for npm

4. Install Microsoft Visual Studio Code

(NOTE: This is the coding IDE that was used to develop the project, while you are free to make use of any IDE of your choice please note that the following instructions pertain to the use of Microsoft VS Code.)

4.1. Download Visual Studio (https://code.visualstudio.com/Download)
- Click the download based on your operating system
- Once the download is complete, open the installation wizard

4.2. Install Visual Studio
- Provided is the documentation to install VS Code on your system based on your OS (https://code.visualstudio.com/docs/setup/setup-overview)

4.3. Launch VS Code
- You may be prompted to install additional components or extensions. You can choose to install them now or skip this step and install them later.

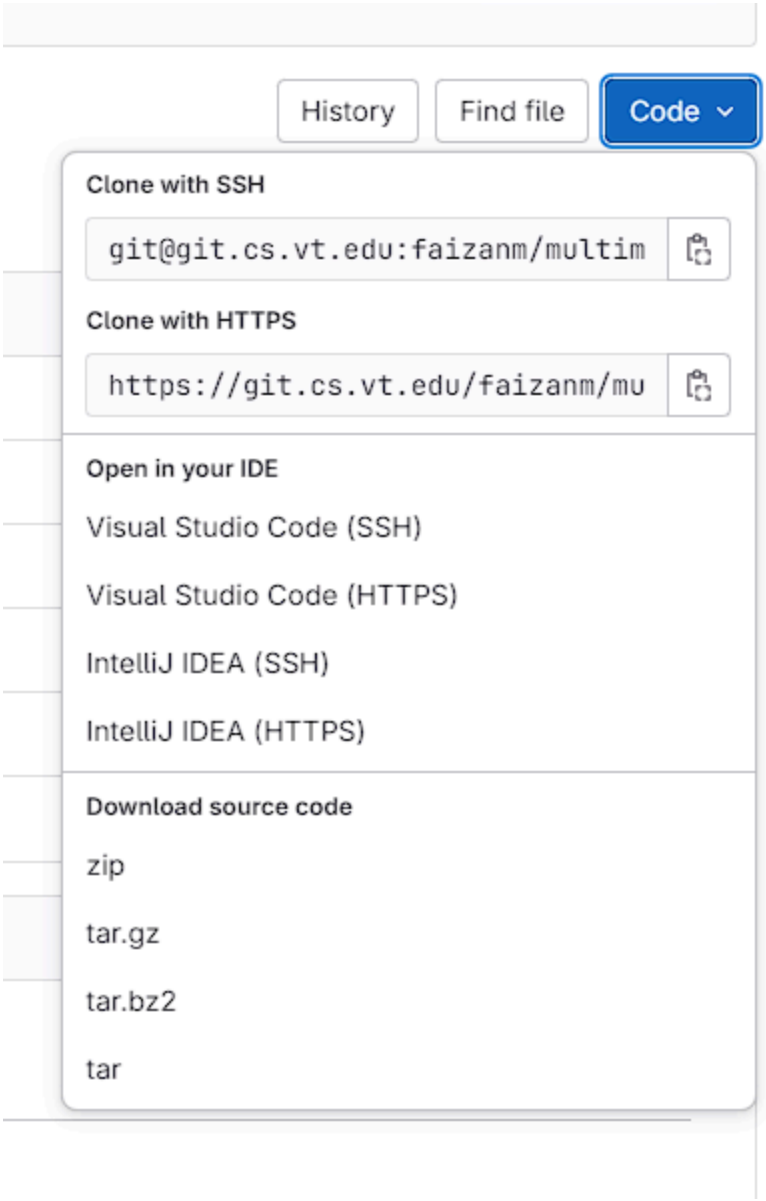
5. Get All Package files

5.1. Go to the following sharepoint to get the needed project folders: https://virginiatech-my.sharepoint.com/personal/srperryman_vt_edu/_layouts/15/onedrive.aspx?login_hint=srperryman%40vt%2Eedu&id=%2Fpersonal%2Fsrperryman%5Fvt%5Fedu%2FDocuments%2FCS%204624%20File%20Share&view=0

5.2. Unzip both the dump and the Projects

Deployment from Local Install

1. Clone the project:
    1.1. Open a two terminals on VS code and type the following command in the first:
    git clone https://git.cs.vt.edu/faizanm/multimedia-capstone-project.git
    or
    You can navigate to the link here, where you can download the project directly after clicking on the blue code button:



2. Install needed packages

2.1. In the same terminal that you cloned the project in type the following commands:

cd multimedia-capstone-project/server/

2.2. next input the following command:

npm install bcryptjs cors express jsonwebtoken mongodb mongoose nodemon axios cookie-parser dotenv multer

2.3. Now navigate to the second terminal you opened and type the following commands:

cd multimedia-capstone-project/frontend/

2.4. next input the following command:

2.4.1. npm install react react-router-dom axios react-toastify, bcryptjs, jsonwebtoken

2.5. Restart VS Code by closing and reopening the application

3. Setting Up the Local DataBase

(This will be done in the Mongos UI interface Compass so to better visualize the process)

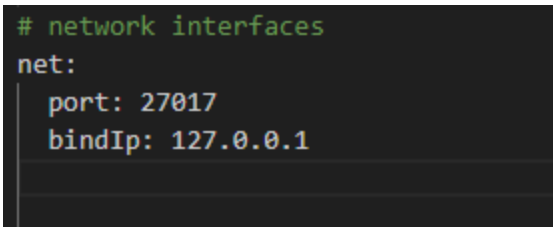
3.1. Compass

3.1.1. First Establish a connection to your local database, by default this is mongodb://localhost:27017, however it is highly recommended that you check the mongod config file which can be found in the bin file of where you have your local MongoDB downloaded.

Examples in default download location:

Windows: C:\Program Files\MongoDB\Server\7.0\bin\mongod.cfg
Mac: /usr/local/etc/mongod.conf
Linux: /etc/mongod.conf

3.1.2. Make sure the number after mongodb://localhost:27017 is the same as the network port in the config file:



3.1.3. Once that is confirmed click the connect button on MongoDB Compass.

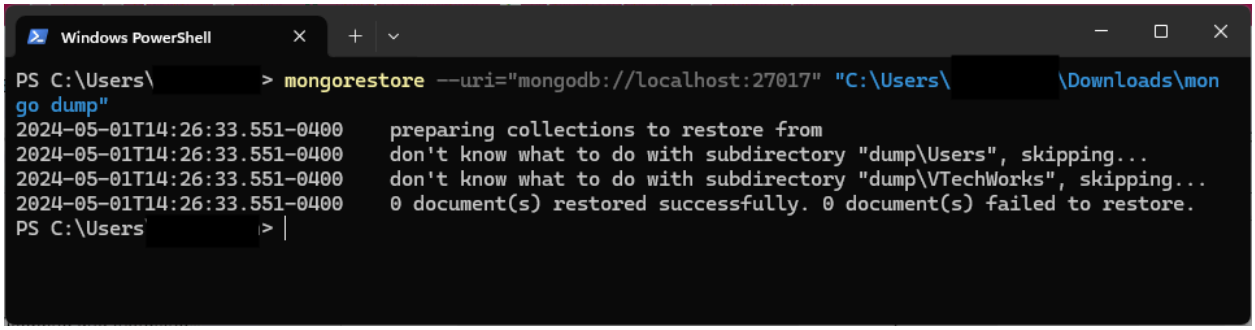
3.1.4. Once the connection has been established you will see a screen that is empty with a side panel that has the word “Databases” and below that should be three databases called “admin”, “config”, and “local”.

3.1.5. Now, pull up a command line terminal

3.1.6. Once you have brought up the terminal, you will take the dump file supplied with the code and run the following command:

mongorestore --uri="mongodb://localhost:27017" path/to/extracted/zip/folder/dump

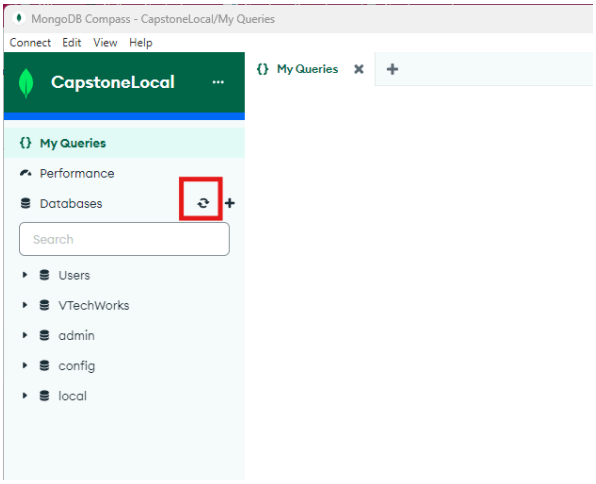
If you get an error regarding subdirectories that looks like this:



That means you need to alter your path to the extracted file by adding \dump to the end of it.
I.g.

mongorestore --uri="mongodb://localhost:27017" "C:\Users\JDoe\Downloads\mongo dump\dump"

3.1.7. Now on the MongoDB Compass click on the icon of two arrows moving in a circle to refresh it and you should see all the files loaded.

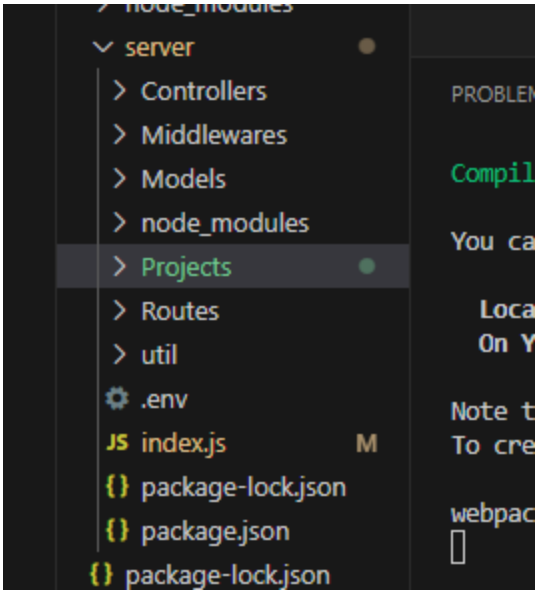


4. Getting started

4.1. Now that all the installs have been completed and the database set up we can now start the system

4.2. First things first take the Projects folder that you downloaded from the sharepoint earlier and unzip the folder.

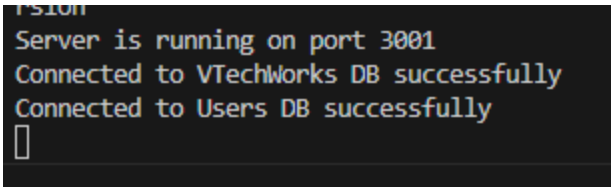
4.3. Next open the unzipped folder and you should see another folder of the same name. Drag and drop that nested folder so it will be copied into the server folder of the project, replacing the current place holder projects folder that exists



4.4. In the first terminal where you are in the /server/ directory type the following command:

npm start

This should give you a block of text in the terminal as it boots up and should end with the following three connection lines:



4.5. In the second terminal where you are in the /frontend/ directory type the same command:

npm start

This should then navigate you to your browser where the website has opened. If not just open a browser of your choice (for this project we used Google) and you should be greeted with the login page.

You have now successfully installed the project and can begin working with the portal. We thank you for downloading our project and wish you the best of luck.

Implementation

Frontend Development

We are developing the front end with ReactJS and are styling the user interface with HTML and CSS. Furthermore, we are developing the front end via different components for the Homepage, User View, Admin View, etc. This strategy allows for us to develop the interface in increments and allows for our team to better document our progress.

Backend Development

Our backend is built on Node.js, utilizing Express.js to create a server that efficiently handles HTTP requests for various functionalities including user authentication, data retrieval, and administrative operations. For security, we incorporate bcryptjs for hashing passwords, ensuring that user credentials are stored securely. Additionally, we use json web tokens for authentication, enabling secure communication between the client and the server. This token-based approach allows us to implement protected routes, where certain actions, like creating an admin user or changing a password, require authentication and specific role-based authorization. The backend architecture supports CORS, enabling our ReactJS frontend to interact with the backend without encountering common web security issues related to cross-domain requests. Our routing is structured to offer clear pathways for different types of requests. This ranges from user registration and login to fetching collection data and managing user roles.

Database Development

Database development has been quite the obstacle for our project. Finding a free source to store the files of 269 projects was a hard thing to find. Opening and analyzing the size of the files from a small selected sample projected that the size of all files would be around 200gb. MongoDB is definitely a valid option and one that has been favored for us for the majority of the design. However, MongoDB is only able to do this for a fee once it has passed the limit of 16mb. Not only this, but given the size and count of files that would need to be transferred would also have to call for sharding/fragmenting to MongoDB. This process would be crucial to maintaining file content and avoiding any files corrupting and to say the least, a very difficult process. Thus, we decided to leave MongoDB to handle the schema while containing the files within the projects locally on our machines. Within VTechWorks, we went through each project individually and downloaded each file onto a flash drive that followed a path of USB (D:) > Projects > (‘Project Title’). Project content will be accessed and displayed by the backend through file pathing.

User Manual

Starting the System

Step 1: Open our source code in VSCode (do not connect to rlogin)

Step 2: On one terminal, first cd into the project folder and then the folder named “server”. Run command “npm start”. Wait for the “MongoDB Successfully Connected” Prompt.

Step 3: Open a new terminal, cd into the project folder, and then cd into the folder “front end”. Run command “npm start”. You will be redirected to the webpage for the project.

New users should click “Sign Up” at the top right corner of the screen to access features of the site. Returning users/admins should click “Sign In” to complete the user authentication process.

There are different views for each of our three user types: students, clients, and admin. Each of these views requires slightly different user interaction, so they will be broken down into sections.

Student Usage

Upon signing in to the system, the student user will be greeted with a home page view.

Note: It may take a few seconds for the server to search through the database to display the projects available for students to view. The user will see a “Loading” screen while projects are being fetched.

Once projects are done loading, the user will see a list of projects underneath the “Current Projects” heading. The projects listed in this section are current projects created by clients for students in the current term.

The user can click on any of the projects by selecting the hyperlink connected to each project title. Once a project is selected, the user can read through all of the project information to determine if this is a project they are interested in completing for their capstone project.

If the user decides they wish to sign up for this project, they can click on the “Join Project” button. If there is still availability for them to join, they will successfully be listed as an author under the project. If the project is already at capacity, they will receive an error message. Using the “Home” or “Go Back” navigation buttons, users can navigate back to the home page.

Note: Projects may take a few seconds to load when navigating back to the home page

Assuming a student user has joined a project, there will now be a project listed under the “Your Project” heading.

Note: Students may only sign up for one project. If they want to change projects, they must select the “Leave Project” button at the bottom of the page. After doing this and confirming their action, they will be able to browse projects again and sign up for a different project.

If the user wants to view other students who have signed up for the same project or review information about their project, they simply have to click on the title of their project under the “Your Project” category.

The final functionality allowed by the user view is a “Sign Out” button which signs the user out of their accounts and brings them back to the sign in page.

Client Usage

Upon signing into the system, the client user will be greeted with the home page view.

Note: The projects under the “Current Projects” heading may take a few seconds to appear due to the server searching through the database to find projects with a date in the current term.

The main functionalities on the client view is to create new projects for students to sign up for, view recently added projects to see which students signed up for them, and view any completed projects from previous terms.

Note: The client user can only view projects that they are the client of.

To create a new project, the client must select the “Create Project” button on the bottom right of the page. This button will open a form with fields that describe all needed information for a new project. After filling out each field, the client can navigate to the next page of the form by

selecting the “Next” button. If the client has students in mind to work on a project while creating it, they can add their names to the “Author” fields. The form of the names should be “Last name, First name”. If there are no authors at the time of creation, the client can leave these fields blank for students to sign up later. If the client needs to add more than one author or keyword, they can click the “Add Author/Keyword” button below the respective field. The client can also choose to leave certain fields blank and they will not appear on the detail view when viewing the project after creation, but long title, abstract, and the date fields must be filled in or the server will not allow the user to proceed. Once all of the desired fields are completed on all of the pages of the form, the client must click “Create Project” on the last page. The system will prompt the client with a message indicating success, then the client can return to the home page by clicking the blue “Home” or “Go Back” button in the top right of the page.

Once back on the home page, the client will see their new project under “Current Project”. It is also under this heading that clients can view any other projects they have created. When a project is clicked on, it opens a detailed view with all of the relevant fields and displays any student names that may have signed up to work on the selected project.

If the client wants to delete a current project, they simply have to click on the project’s title and then when they reach the detailed view they must click “Delete Project”. The system will ask the user to confirm this action before permanently deleting the project. Once the user returns to the home page, the deleted project will be removed.

If the client user has created any projects in previous terms, their projects will appear under the “Previous Projects” heading. When the client clicks on a past project, they will be able to view all relevant fields and any submitted files associated with the selected project.

Another functionality of the client user is that they can search through all of the projects. If the user types a name in the “Search by Author” field and clicks “Search Author” they can find projects completed by a certain author. If the user types a word in the “Search Projects”, the table will automatically display projects that contain the searched word. In terms of filtering, the user can filter all results by A-Z, Z-A, and by year. If the user selects to sort by year, they can narrow the results down further by selecting a certain semester.

The final functionality allowed by the client view is a “Sign Out” button which signs the user out of their accounts and brings them back to the sign in page.

Admin Usage

Upon signing into the system, the admin user will be greeted with the home page view.

Note: The projects under the “Current Projects” heading may take a few seconds to appear due to the server searching through the database to find projects with a date in the current term.

The main functionality within the admin view is the ability to browse through all projects, both current and past. Under the “Current Projects” heading, the user can view all projects that exist within the current term. Under “Previous Projects”, the user can see all past project submissions that were submitted to the database.

The admin user has the ability to create new projects or import old projects. They can create a new project by following the exact directions in the client view section. To “import” old projects, the admin should follow the following demonstration:

1. Click “Create Project”
2. Fill out the following fields exactly as stated (omit quotation marks when filling in fields):
    a. Long Title: “SharkPulseApp”
    b. Abstract: "In recent years, collaborative work between previous CS4624 capstone groups and Dr. Francesco Ferretti's team contributed to the development of the SharkPulse project. With the goal of enhancing shark conservation and elevating public awareness through the collection and analysis of global, crowd-sourced shark sightings data, SharkPulse developed a data / machine-learning pipeline to detect and classify sharks from a given image. This report presents the improvement of the machine-learning pipeline previously established in "Shark detection and classification with machine learning" (Jenrette et al.). The improvements to the pipeline increased classification accuracy as well as species breadth. Mainly, the existing classification architectures are replaced with Transformers (ViTs). The updated shark identifier achieves an accuracy of 96%, the updated genus classifier, an accuracy of 72%, and the updated genus-specific species classifiers, an average accuracy of 74%. This updated classification system is able to classify 27 genera and 51 species. A framework for automating data-collection, model training, and maintenance is also introduced. Potential future work is discussed, including integrating the model into the SharkPulse platform."

    c. Author (click “Add Author” for each additional author): "Pham, Khanh", "Lemus, Catalina", "Ansari, Mohammed Al"
    d. Year: “2023”
    e. Month: “05”
    f. Day: “15”
    g. Keyword (click “Add Keyword” for each additional keyword): “shark”, “validation”, “map”
    h. Publisher: “N/A”
3. Leave all of the extra fields blank and click to the last page of the form to select “Create Project”
4. Click the “Home” button

Once following these steps, the admin can click on the “SharkPulseApp” project to see that the project has been auto-populated with its associated files from local storage.

The admin also has the ability to delete current projects. Past projects cannot be deleted. If the admin wants to delete a current project they simply have to click on the title which will open the detailed view and then click on the “Delete Project” button. The server will give the user one last chance to cancel deletion before proceeding. Once a project is successfully deleted, the server will notify the user and upon returning to the home page the project will be removed.

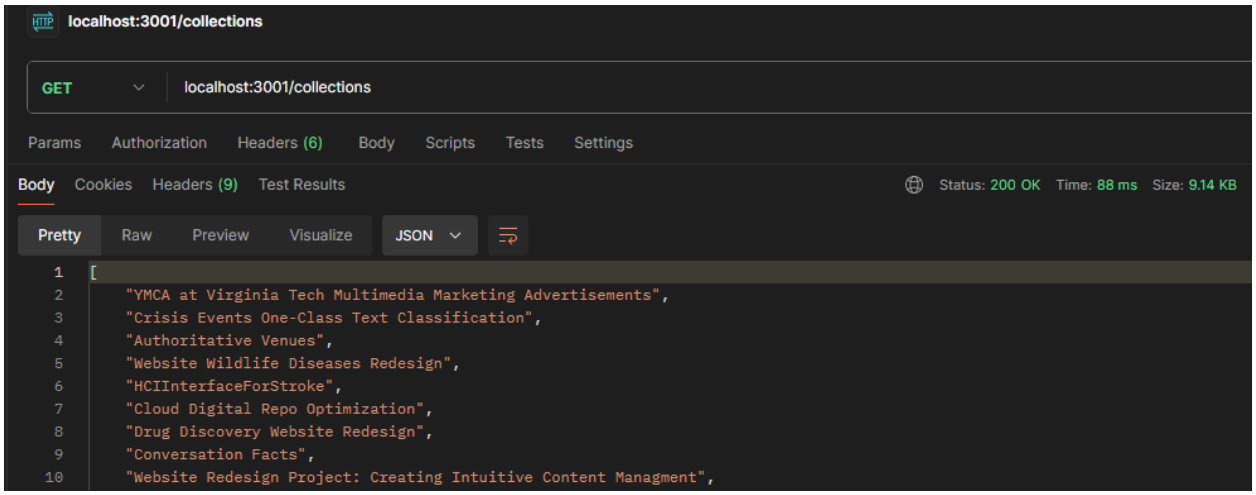
Another functionality of the admin user is that they can search through all of the projects. If the user types a name in the “Search by Author” field and clicks “Search Author” they can find projects completed by a certain author. If the user types a word in the “Search Projects”, the table will automatically display projects that contain the searched word. In terms of filtering, the user can filter all results by A-Z, Z-A, and by year. If the user selects to sort by year, they can narrow the results down further by selecting a certain semester.

The admin user also has the privilege to create new users by selecting the “Create New User” button. When they click on this option, they must fill out the first name and last name of the new user, set their username, set their password, and choose whether they should be a student or a client user. Once submitted, this user will be added to the system.

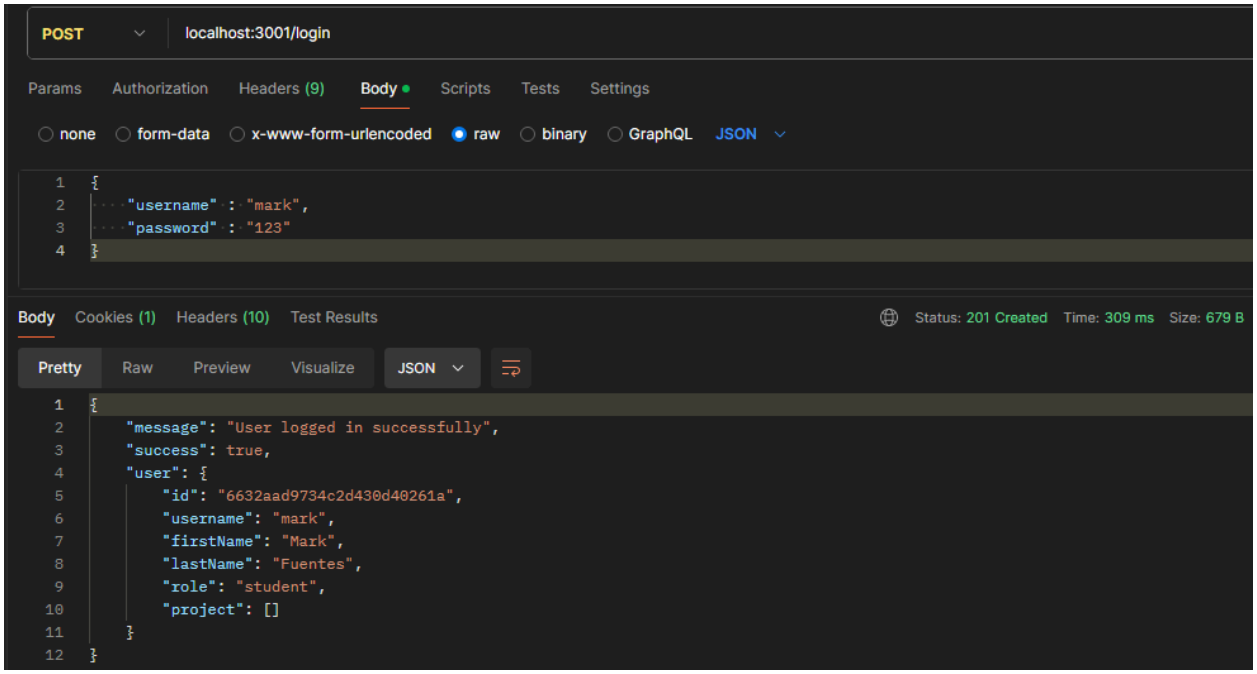
The final functionality allowed by the admin view is a “Sign Out” button which signs the user out of their accounts and brings them back to the sign in page.

Testing

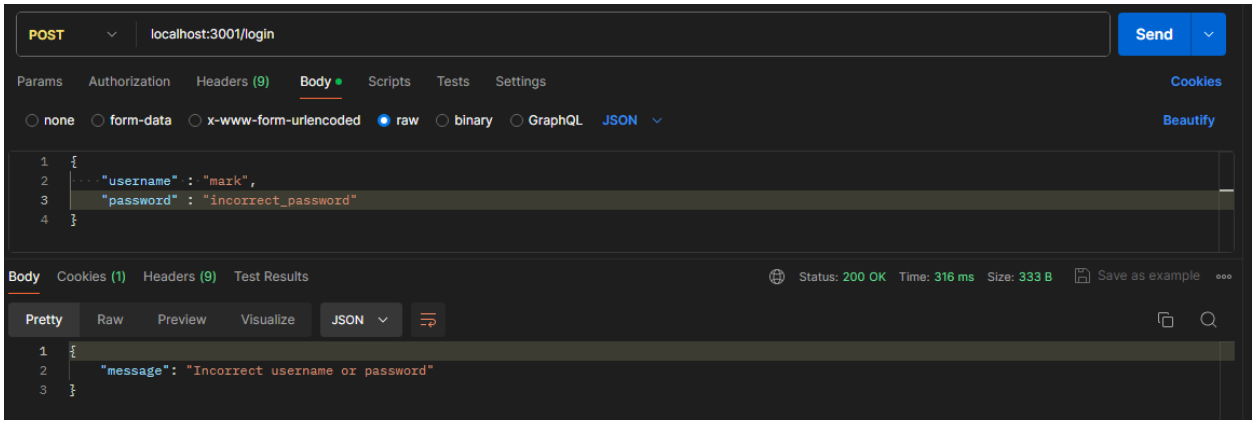
For the back end, tests were done using Postman to test the functionality and accuracy of our created APIs. Using this, we were able to test successful database connection and search. This was important because it allowed us to determine if errors were occurring in the backend or from errors in rendering the frontend.



Above is an example of the /collections API successfully returning the names of all projects. Postman was especially helpful in also testing the HTTP status codes. Below, we can see an already created user successfully signing in (right below)



and unsuccessfully signing in with an incorrect password response.



After thoroughly making sure that each API worked the way it intended with its required response body, returning the correct json data, and giving the correct status code, we then moved to implementing the frontend which was written and tested incrementally. To start, we wanted to make sure that user sign-in and sign-up worked and routed the given user to the correct home page. After writing the backend portion of it, its frontend was created and tested through the use of visualizing it after running “npm start” and also checking the database to make sure it was storing the data in the users database.

As we added more functionality to users and projects, all backend code was again first tested through Postman before any frontend code was written. Once everything was inputting and outputting correctly, again we made small changes to the frontend to then be tested visually and correctly updating the database. This approach allowed us to fully determine which part of our code could potentially give errors. If something wasn’t showing up correctly on the frontend, a simple check in Postman could let us know if this was a backend or frontend issue.

Future Work

After working hard on the CS 4624 Discovery Portal all semester, we are very proud of all we have accomplished in terms of meeting our client's functionality requests. However, we have a few additional features we wish we had been able to implement during the semester if time constraints allowed.

First, we would like to compartmentalize our project better by using Solr for our search functionality. We decided on using Javascript for our searching and filtering to keep the feature simple under our time constraints, but switching to Solr would be a great idea for improvement in the future. Second, we would like to dockerize our project. Dockerization could significantly improve the efficiency of downloading and setting up our source code for new users. Third, we would like to improve the formatting and aesthetic of the front end. Currently, a lot of our views are dependent on the amount of text in the table. This leads to inconsistent sizing of our containers which is not satisfying to view. In the future, an upgrade to the entire front end would significantly improve the user satisfaction while using our project. Fourth, the security of our authentication could be improved. Currently, our authentication encrypts passwords for security, but a user can easily bypass the login screen by changing the pathing in the web url. Increasing security measures here would improve the overall reliability and usability of our app. Finally, the last change we would want to make in the future would be to support final project submissions by students. Currently, all submissions for CS4624 are uploaded to VTechWorks. To make our web application fully ready to replace VTechWorks, we need to support final submissions by students. This would entail allowing files to be uploaded, edits to the project fields by students, and a final submit button that moves the current project to the past project category. We hope that the work we have accomplished this semester will be improved and used in the future for new CS4624 students years after we graduate from Virginia Tech.

Timeline

Every 2 Weeks Every Week	Meeting with client (Fridays 3:00pm-4:00pm) Meeting with team (Wednesdays 7:00pm-8:00pm)
Week 4 (2/5)	Derive requirements, assign requirements to each team member, develop specific plans for implementing requirements, create presentation 1, split up presentation amongst team, practice presentation
Week 5 (2/12)	Present presentation 1, contact client to discuss requirements
Week 6 (2/19)	Make edits to presentation 1 after feedback, begin design phase
Week 7 (2/26)	Begin implementation - keep it very basic for now focusing on linking and functionality
Spring Break (3/4)	⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗⊗
Week 8 (3/11)	Continue implementation, begin preparing presentation 2, start pulling data from VTechworks into database, Start making front-end features interact with back-end
Week 9 (3/18)	Present presentation 2, ensure functionality between front-end features and back-end
Week 10 (3/25)	Work on interim report, continue implementation, determine plan for rest of semester, prepare VTechWorks, Interim peer reviews
Week 11 (4/1)	Make edits to presentation 2 after feedback, have working database linked to front end & back-end
Week 12 (4/8)	Begin testing project and prototype, fix any bugs and implement tweaks
Week 13 (4/15)	Begin working on final presentation, final report, and ensure project performs as expected
Week 14 (4/22)	Final presentation & peer reviews
Week 15 (4/29)	Final project established

Lessons Learned

Overall, development and design has been smooth at a steady pace. Regardless, there are always some issues that arise from software development.

An issue that we faced during the development of the front end was confusion with the many css files used for styling. Instead of having an individual CSS file for each component of our project, in retrospect, we should have tried to consolidate these files. This would have saved our team a considerable amount of time in debugging issues with styling. One such example would be that we spent a lot of time trying to figure out why the header component of the component was not a fixed width on all of the pages.

Furthermore, we believe that the group would have benefited from more piecewise development of our platform, particularly concerning the front end. We developed many of the components for the front end simultaneously, which caused issues later on. When we tried to merge all of these components together, there was lots of unexpected behavior that needed to be resolved. If we had ensured that there were no issues with the existing components before starting on the new ones, our team would have saved time trying to resolve these issues.

One of the first issues encountered was an attempt to run our program within rlogin.Assuming unfamiliarity with rlogin, rlogin is a unix program that acts as host using Virginia Tech's network. During the linking phase between our front-end, back-end, and database, running the program resulted in the terminal to hang with no output of any kind. This had an easy fix which was just running the program on our local machines. While this may seem to be something that does not require in-depth explanation, it is important to note that rlogin is typically used for all CS projects. This will lead to many people making the assumption that programs will need to run within rlogin. Another issue was the development of our data-base. As discussed earlier, there are many reliable methods that one can use to create a database consisting of all projects and files. However, with the avoidance of paying any fee or subscription, the options become much more limited. It’s a good idea to allow MongoDB to act as a schema for the filtering function, having each collection named the title of the project. This way, the filter function will access MongoDB for key aspects to allow for filtering whilst being able to access the file contents within the local machine using pathing. Initially, MongoDB was going to be responsible for containing both the schema and files paying a subscription fee. After reading and attempting to understand the process of sharding and fragmenting, this idea was shot down. Sharding and fragmenting is a very difficult concept to understand, let alone to correctly implement, and the transfer of the database is completely reliant upon it.

Lastly, which is the most obvious and important, is to communicate with the client as frequently as possible. Early on in the semester, the project details were extremely broad and even unknown. The more we interacted with our client and each other, the clearer our design for the project became.

Acknowledgements

We would like to acknowledge our professor and client Dr. Farag (mmagdy@vt.edu) as well as the class GTA Vedant Shah (vedant02@vt.edu). They have proven to be invaluable resources that have helped tremendously with any issues we have encountered.

References

- “CS4624: Multimedia, Hypertext, and Information Access.” https://vtechworks.lib.vt.edu/collections/d467626b-ddf1-4a8f-ab15-5d931bfae4c7

- “CSS: Cascading Style Sheets | MDN,” *MDN Web Docs*, Mar. 05, 2024. https://developer.mozilla.org/en-US/docs/Web/CSS

- “Index | Node.js v21.7.1 Documentation.” https://nodejs.org/docs/latest/api/

- “JSON.” https://www.json.org/json-en.html

- “MongoDB documentation,” *MongoDB Documentation*. https://www.mongodb.com/docs/