

Group 6 - PortfolioAlec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Section 1: Program

<https://github.com/ryanhansen2222/esof423-s2021><http://esof423.cs.montana.edu:4006/home>

Section 2: Teamwork

1. Alan Tong

I completed the server side integration between the Adonis.js framework and the university's school server. Along with the server integration, I implemented front-end development of the comment API such as the home and log-in pages in HTML and CSS. Further, I documented our releases and the changes to our codebase as well as weekly debugging testing with the group. The majority of the hurdle for the project for me has been understanding each team member's contribution after every push onto the Github repository and fully understanding what each group member does before progressing further into the project without bugs and issues.

2. Samuel Forbes

As far as things I accomplished, I wrote database MySQL files using Amppps for the project. I completed the server-side integration between the Adonis.js framework and the university's school server independently of Alan as well. In addition, I helped update the management tools. Further, I started to integrate the MariaDB database into the school server. Finally, I've been majorly updating our CSS and EDGE files in the pursuit of making our final product more appealing. For the longest time, my hurdle was getting all of the Adonis files working locally. I could get them to work on the school server, but I didn't have the app key and everything to get it working locally. Now, my major hurdle would probably be working with GitHub. Simple Git operations are fine, but when it comes to working with different branches, it's been somewhat difficult. Generalizing, my main backlog items included researching Adonis, database integration, managing tools, figuring out the school server, researching APIs, and managing portfolios. This is projected to about 30.5 hours, but I would say I've spent closer to 35 hours on it.

3. Ryan Hansen

I focused primarily on full stack development (~25hr), but additionally spent some time on testing (~2hr) and documentation (~3hr).

4. Alec VanderKolk

For my part of the project, I have spent the majority of the time researching the functionality of software involved such as adonis, postman, and swagger. Additionally, I set up my local system and my school server account to run our project as every group member has. I created the UML diagrams for our project, namely, a class diagram, use-case diagram, and a package diagram. Additionally I designed the entity relationship diagram for the initial database design. All of these diagrams are included via hyperlink in section 5 below. On a few occasions, our group has gathered in or out of class and collectively troubleshooted problems, particularly with coordinating version control on github, fixing routing to video and comment CRUD views, and resolving server errors. Lastly, for the final weeks of the project, I have been working on creating the API specification. I have used swagger inspector to compile OpenAPI definitions for the site's

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

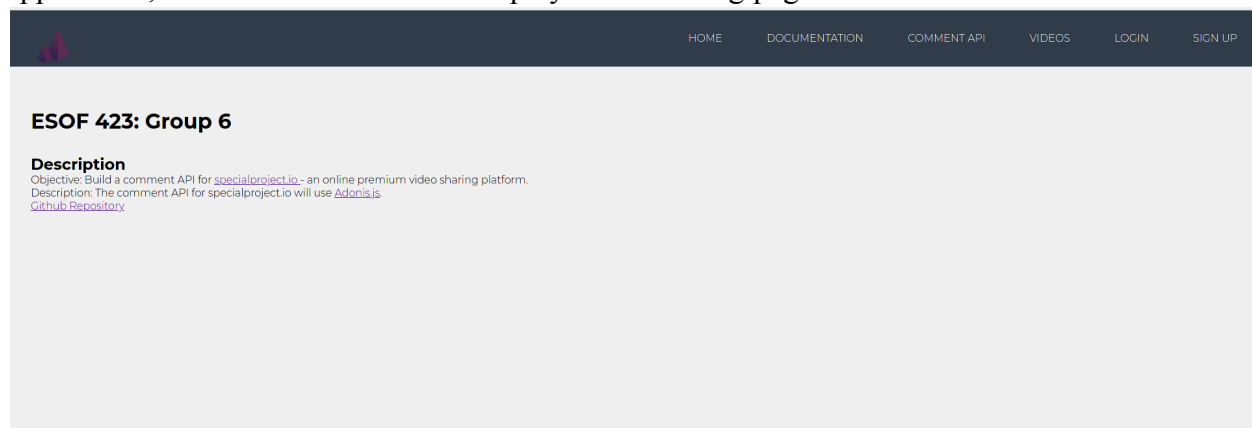
basic get calls. I created a test suite using Selenium that tests all links across the site, special characters in text fields, comment and video CRUD, user signup/login, and invalid text inputs. I did not keep track of the time spent on this project in total, but I probably put between 15 and 20 hours into the project. This number is hard to estimate as I don't really consider the researching aspect of the project to be time spent on the project. My biggest issues have been with resolving inconsistencies in our github repository and determining the most effective way to create the API using our existing codebase.

Section 3: Design Pattern

A design pattern used consistently throughout our project is the state pattern. The primary states that alter the functionality of the project are whether or not a user is logged in. If a user is not logged in, they cannot see the My Videos page, post any content, or logout. Once they are logged in, however, the new state now displays My Videos, displays Logout tab, and allows for content creation while removing the Login and Sign Up tabs.

Section 4: Technical Writing

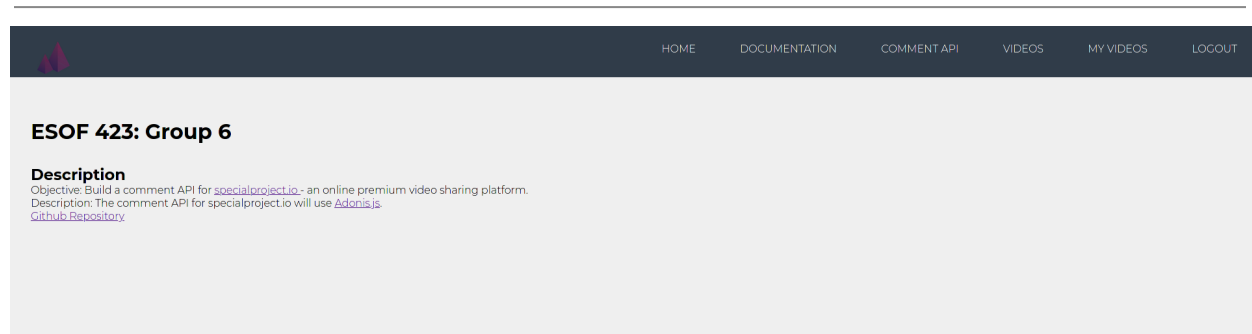
The special project API runs on an adonis framework with a sqlite as the database. It requires a number of dependencies including adonisjs, sqlite3, adonis-swagger, @adonisjs-ignitor, nodemon, Node.js > 8.0.0, and NPM > 3.0.0. The project was built on a Linux platform, but can be adapted to work on other operating systems. To use the API, first clone the repository at <https://github.com/ryanhansen2222/esof423-s2021>. Within this repository, navigate to /src. The adonis project to use within this folder will be named /yardstick-archive. Assuming all dependencies are installed, running nodemon server.js from the command line while in the directory yardstick-archive should start the project. This will initially be set up to run on the local host, but the .env file can be modified to match the needs of the user. Upon serving the application, the home screen should display the following page:



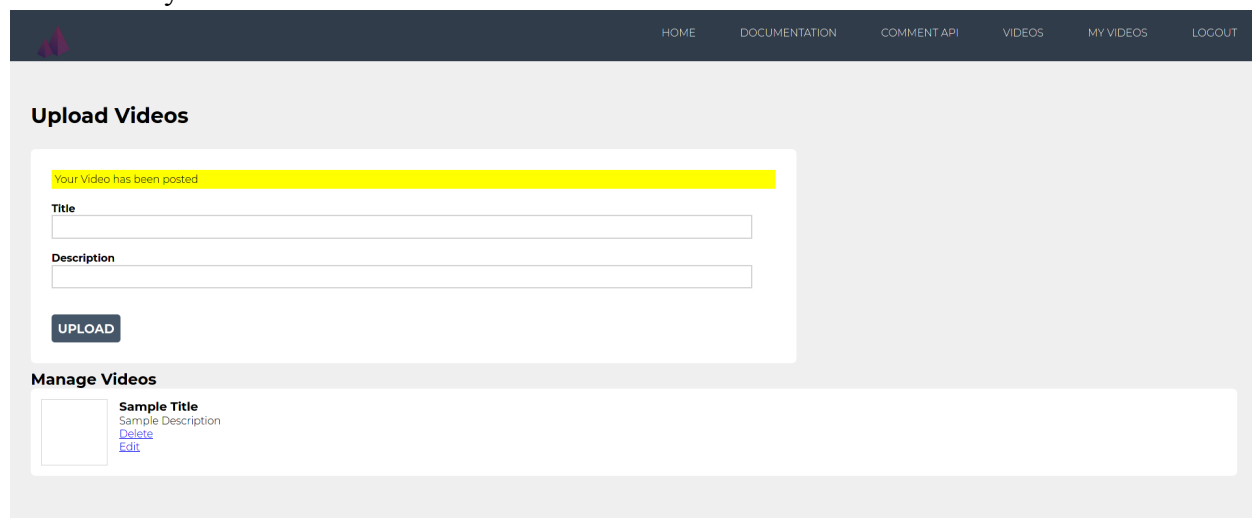
From this point, a user can view the documentation for the API, login, or sign up for the site. Upon creating an account or logging in, the home page changes to the following:

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes



Note that the difference between the previous two pages is the addition of the “My Videos” tab and the “Login” and “Sign Up” tabs have been replaced by “Logout.” The core functionality of the API works through the “My Videos” and “Videos” views. From within “My Videos,” new media can be uploaded to the site. Note that the media upload still needs to be implemented for this section and currently video creation simply creates a video title and description. Once a video has been created it can be edited or deleted from the “My Videos” page. This functionality is outlined by the screenshot below:



Once a video has been posted to the site, users can comment on the video under the “Videos” tab. “Videos” contains a paginated view of videos posted to the site and users can comment by selecting a video and filling out the associated GUI on the /watchvideo view. Once a comment is posted, it appears in a paginated list of comments under the video content. Comments can be liked by any user, but comments can only be edited or deleted by the user that created them. This functionality should appear as in the following image:

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes


1

Video plays here


-- paginationTestUser --

COMMENT

All comments



paginationTestUser:
it
[Like 0](#)



h:
test comment
[Like 1](#)
[Delete](#)
[Edit](#)

[Previous](#)
1
2
[Next](#)

The yardstick-archive repository can be copied and modified directly to meet the needs of the user. Alternatively database migrations, routes, edge files, controllers, and other required files can be used individually to meet the user's needs from within their own adonis project.

A Selenium test suite is included within the repository under /src/yardstick-archive/public. To run this, the Selenium ide is a required extension from your browser. Select the Selenium extension in your browser and choose open existing project. Select the 423 Test Suite.side file. Then, the url within the .side files will need to be changed to match your working url or local host. A video showing this test suite running successfully can be found at: <https://youtu.be/ZJil9BfRrE>.

Section 5: UML

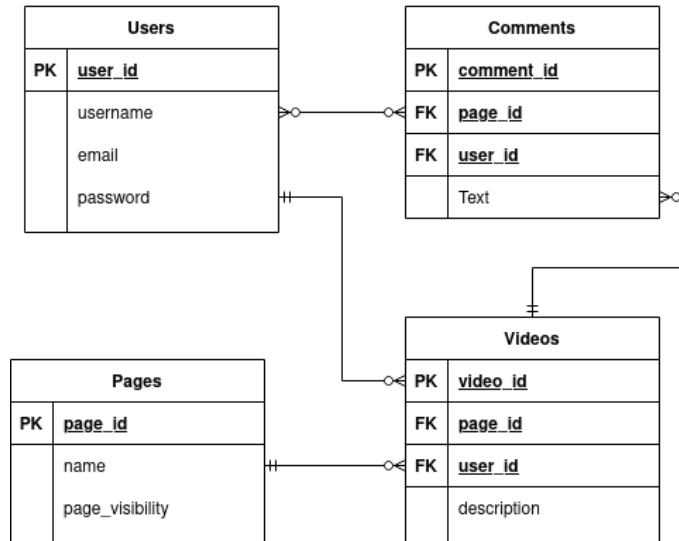
ERD:

https://drive.google.com/file/d/1F4riy0bCoFnYve2mrVIV0ScUry9waL_O/view?usp=sharing

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

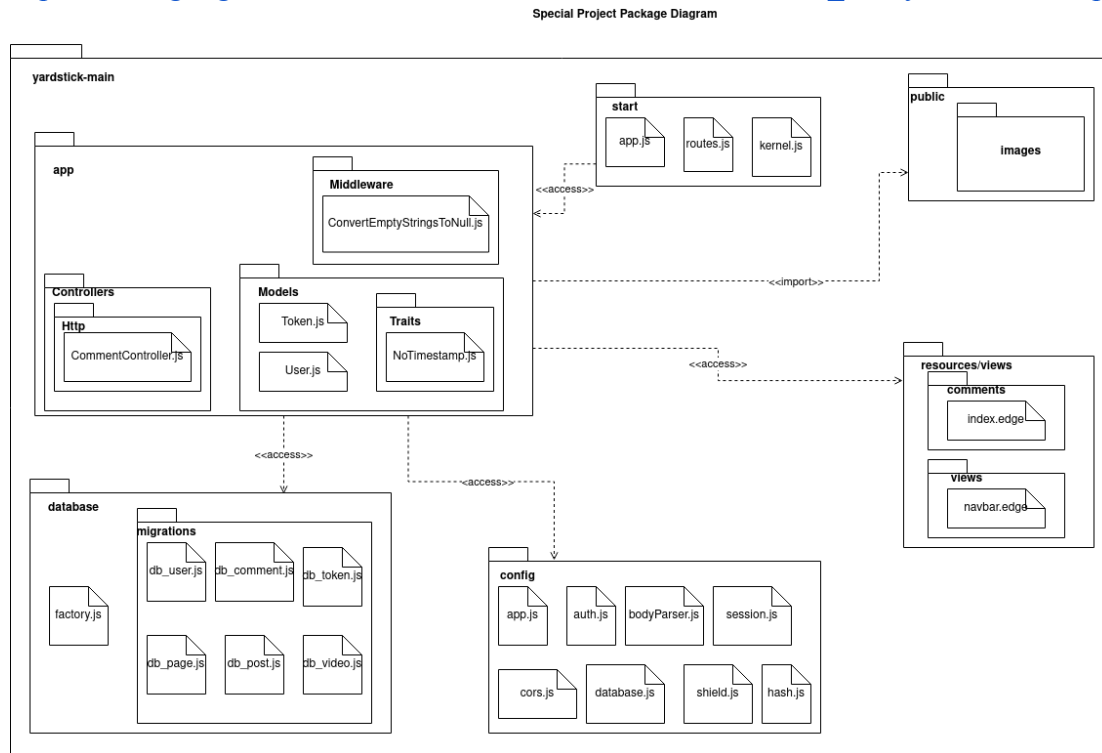
Special Project ERD



Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Package Diagram:

https://drive.google.com/file/d/1vDhDSk-nTQEWzakHBS1CYr_L23iyFfRi/view?usp=sharing

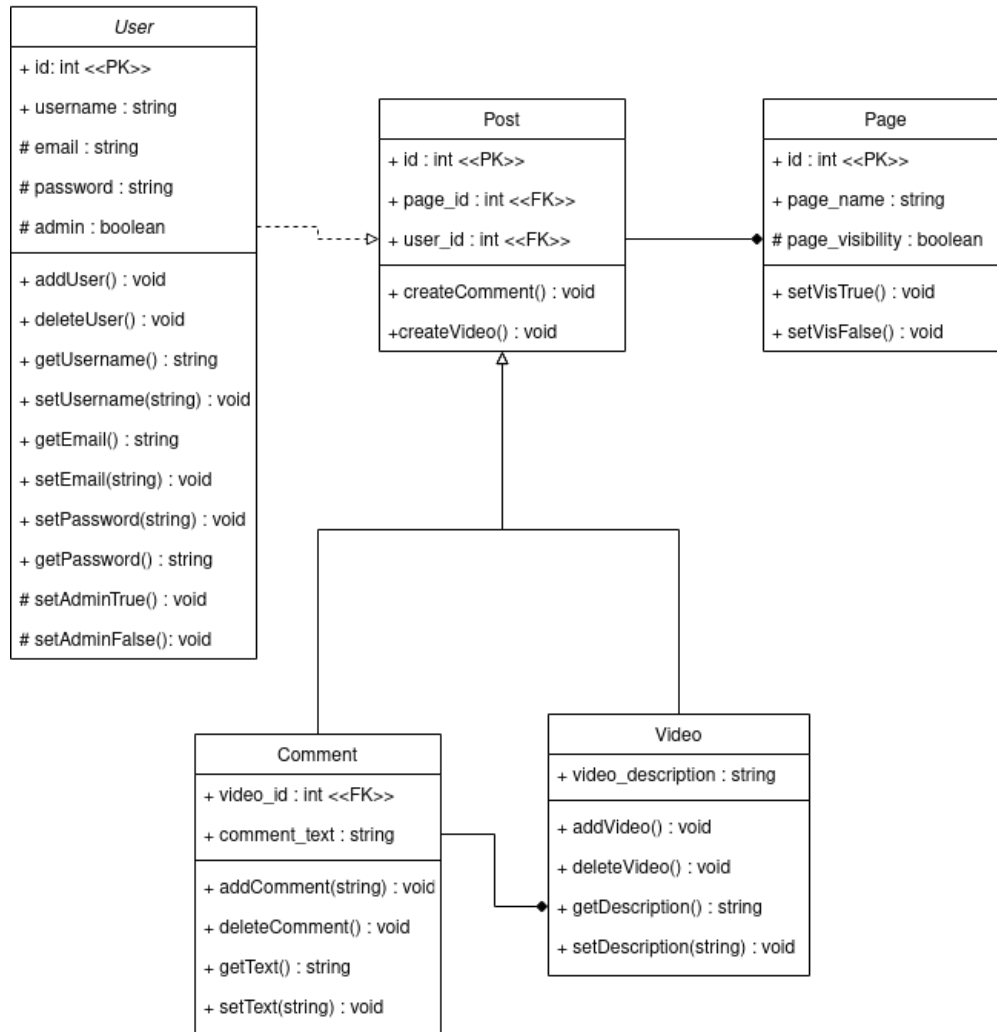
Class Diagram:

https://drive.google.com/file/d/1z0-5B-vE_xTFnt8PyRSp5lZ2qTe5i8S-/view?usp=sharing

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Special Project Class Diagram

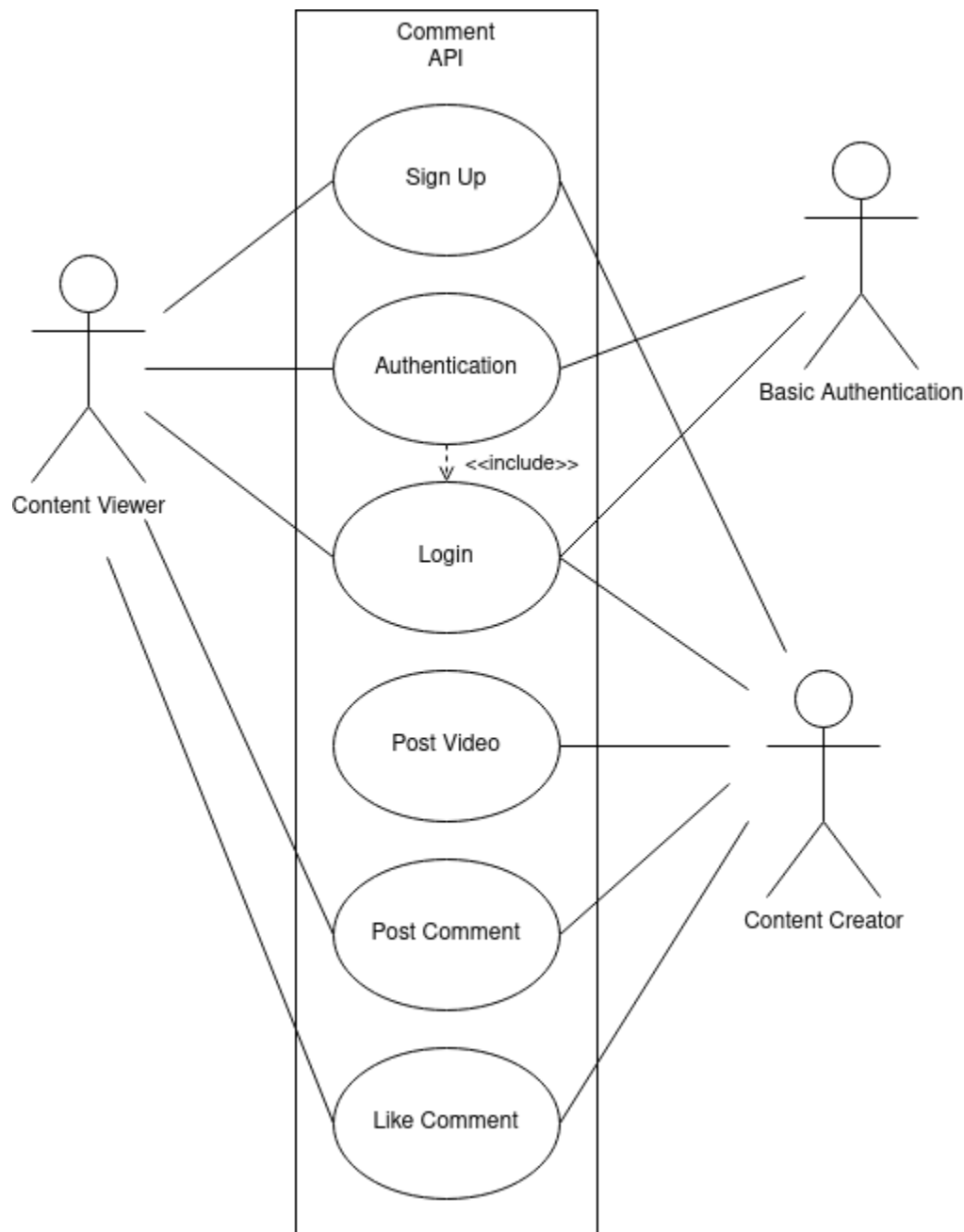


Use-Case Diagram:

<https://drive.google.com/file/d/1spdor4hpAhrVW9fCxtlX2wNfcBWHpE3y/view?usp=sharing>

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

**Section 6: Design Trade-offs**

As far as the videos went, we could have made a more visually appealing hub of videos. We also could have eliminated the possibility of people liking a post indefinitely. Another thing we could have done was to actually have the feature of being able to upload a video and then play it. We could have spent even more time fine-tuning the visuals of this website.

Group 6 - PortfolioAlec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Section 7: Software Development Life Cycle

The primary advantages of agile were consistent updates across group members. Multiple times each week we checked in to determine progress and consult each other on issues, which helped the project move forward. On a larger project like this, the scrum meeting and sprint goals helped us break down the project into smaller chunks so that it was more approachable. At the beginning of the semester, this was more difficult as many of our sprint goals were to familiarize ourselves with the software involved. As the semester progressed, we were able to divide the project more easily by assigning pieces of the portfolio, specific app functionality, and testing across members. The burndown tracking was helpful to know how close we were to a final product, but it did not tend to reflect the time we would have to spend on each portion accurately. Since we were meeting for short amounts of time each week, often there was miscommunication in the time between classes. This led to some inconsistencies across the git repository and for members developing locally. These issues were usually resolved outside of the weekly meetings within class time in longer independent meetings. These meetings seemed far more efficient than developing individually and could have been incorporated into our weekly sprint plans.

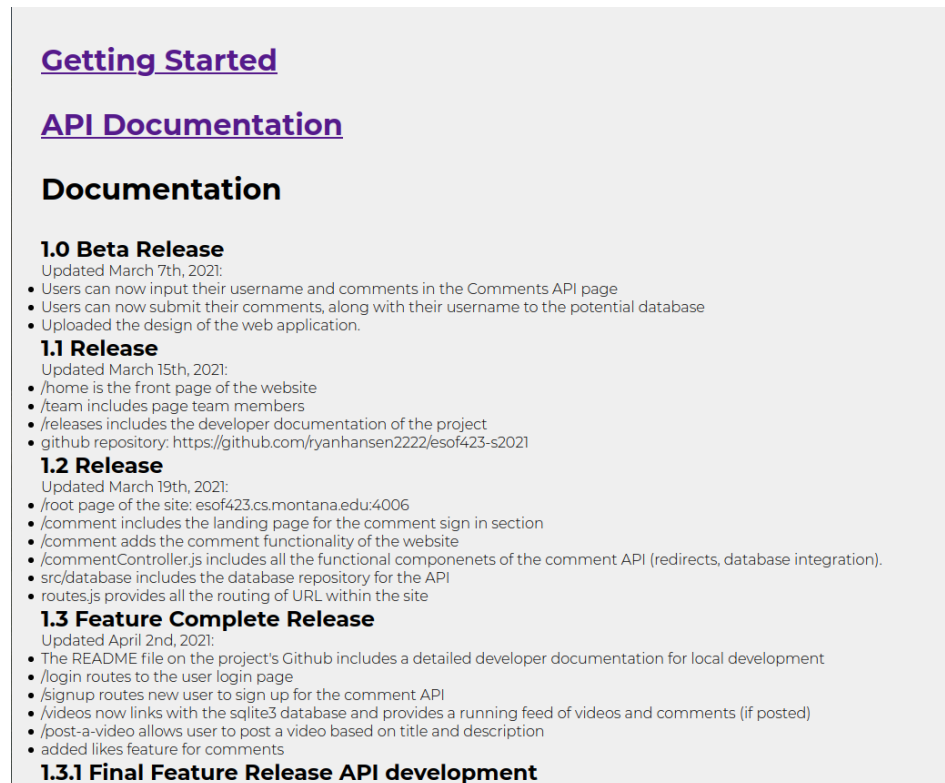
If we were to start this project over, test driven development in coordination with some amount of agile would seem more appropriate. Since the API is constructing a simple comment system, each test could have reflected a small piece of functionality, e.g. object CRUD functionality, comment likes, or page views. This would have allowed for development with less meetings between group members as the things that would be discussed could be turned into tests that the other developers could work to resolve.

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Documentation:**1. Internal (adonis site)**

The central hub for our project documentation is within our front end. It contains links to the other documentation resources we provide (Github, Swagger) as well as some unique front end guides.



Documentation tab of our front end

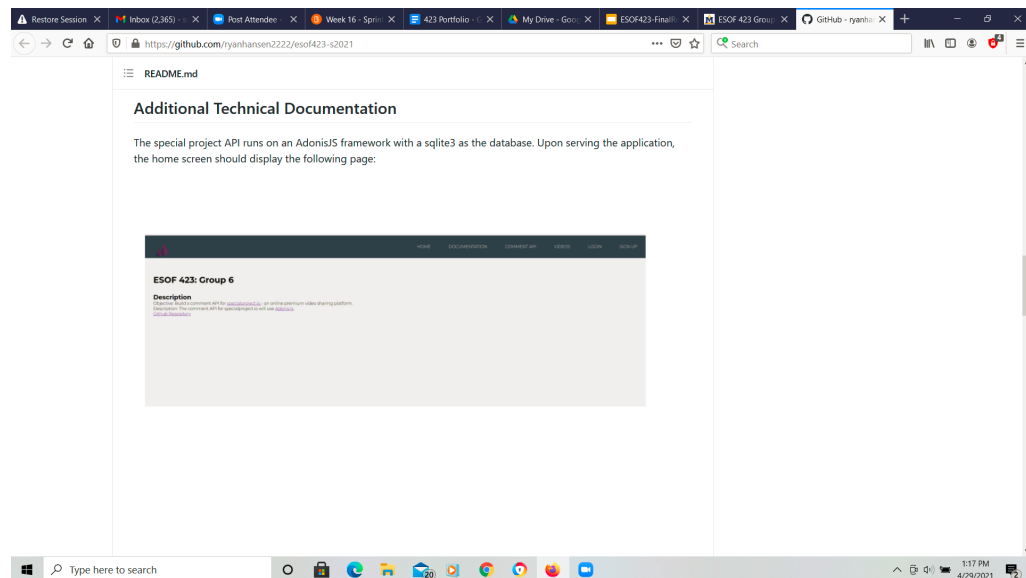
2. Github

Some technical documentation exists on the GitHub repository. A link to it is here:

<https://github.com/ryanhansen2222/esof423-s2021>.

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes



3. Swagger

Swagger provided a nice documentation tool for the API specs of our project. It can be reached from the following link -

https://app.swaggerhub.com/apis/ryanhansen222/special_comments/1.0.0-oas3

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Special Comments API

1.0.0-oas3

OAS3

API for comment data (On video data)

Servers

Authorize



default



GET

/videos



GET

/post-a-comment/edit/{id}



POST

/watchvideo/{id}



GET

/watchvideo/{id}

SSL certificates are validated | [Do not validate](#) ⓘRouting requests via SwaggerHub proxy | [Use browser instead](#) ⓘ**Features:****1. CRUD Comments**

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Pranking Snail



-- test account --

COMMENT

2. CRUD Videos

Upload Videos

Title

Description

URL

UPLOAD

Manage Videos

Pranking Snail

GONE WRONG

DELETE

EDIT

Best video ever

u already know

DELETE

EDIT

3. React to Comments

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes



Likes: -33 Funny: 17



4. API Endpoints for custom comment requests

GET /post-a-comment/edit/{id}

Update a comment (id)

Parameters

Try it out

| Name | Description |
|----------------------|--|
| id * required | The comment you want to edit |
| integer | |
| (path) | <input type="text" value="id - The comment you want to edit"/> |

Responses

| Code | Description | Links |
|------|--|----------|
| 200 | Successfully returned a list of comments | No links |

Media type

application/json ▾

Controls Accept header.

Example Value | Schema

```
[
  {
    "title": "string",
    "description": "string",
    "id": 0,
    "url": "string"
  }
]
```

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

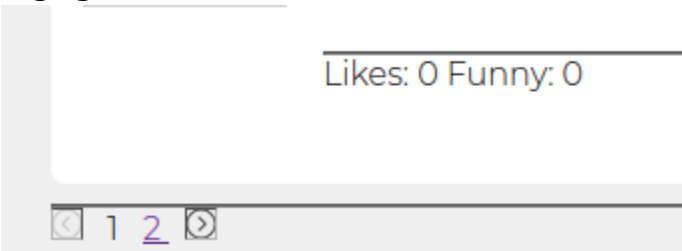
5. Embedded Youtube Videos

Pranking Snail



-- test account --

6. Paging for comments



7. User specific database management options (only like videos when you are logged in, can only edit own comments etc)

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

All comments**test account:**

wow can you prank king kong?



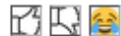
Likes: -33 Funny: 17

DELETE

EDIT

**1percentblack:**

SWEEEEET



Likes: 0 Funny: 2

Testing:**1. User:****User test format**

Procedure: The goal of user testing is to make sure users will successfully use our software. It is unclear how literate or clever these users will be. Our procedure is therefore very simple. Given a computer with a browser open to the homepage of our website, we task our user to find a video they want to watch, comment on it, edit their comment, and finally delete it.

Reasoning: Many of the important features of the API must be used to complete our procedure. While we agree using the API is very different than using our front end, we spent a good chunk of time making sure our API functions interface well with the front end. Most of the API calls boil down to specifically formatted database queries interfaced with an abstract front end. If the user has trouble using or

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

understanding a certain task, at worst we found a failure in effective documentation. At best, we get insight into a new bug in the API and immediately resolve the issue.

User Test Demographic: While the main product we deliver is a comment API, we target the secondary users - people accessing a front end website using the API. We target secondary users for two main reasons. First, within the scope of this class, we do not expect to have a developer build a front end and implement our comment API within our short sprint time-frame. Second, we believe valuable information about the comment API remains even from secondary user feedback.

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

User test 1:

Tanner Smith (roommate, No CS background, but computer literate)

Immediately left documentation page

Resolution: Make it shorter/more appealing??

Clicked around for a while (Super mislead by Comment API tab)

Resolution: Delete Comment API tab

After he found the videos tab, he tried to click on a video he wanted to watch. He had to try a couple times before he figured out you have to click on the video title.

Resolution: Make entire div the a=href

There is no comment button unless you are logged in, which I had to tell him about.

Resolution: Make alt display saying you have to be logged in to comment

Liking comments, disliking, and funny were semi intuitive. He asked if the word 'likes' corresponded with the thumbs up and down icons.

Resolution: Replace those words 'likes + funny' with the icons

FEEDBACK: commenter icons (instead of blank boxes), actual video playback (instead of text)

Resolution: Add those in (like we have been talking about)

User Test 2:

Carson Hansen (No CS Background, brother, computer literate)

Looked at getting started page

Scrolled through pictures super fast (did not read anything)

Resolution: Less text to read, better pictures

Saw picture that showed videos tab on navigation, so left

Found a video, dead clicked on the box a few times

Resolution: Make entire div the href

Could not comment (because not logged in)

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Watched the video a bit

Realized he should make an account to be able to comment/react

-- Make this part more clear

Commented on the video

Started messing around with likes and dislikes

Reactions + Commenting reloads the page

Resolution: Make a static function such that no routes need to be called to load in the page (with components)

Noticed Infinite reactions, and abused it.

-- We already decided this was out of the scope of our project

Feedback: Fix infinite reactions

2. Black Box:

Selenium testing:

Automated testing suite to black box interface with nearly every feature of our front end. Helped catch many bugs.

A few example demos can be reached via the following links -

https://www.youtube.com/watch?v=RmGWo0WXcs8&ab_channel=AlecVanderkolk

https://www.youtube.com/watch?v=ZJiIl9BfRrE&ab_channel=AlecVanderkolk

https://www.youtube.com/watch?v=AlQXuyGknVg&ab_channel=AlecVanderkolk

3. White Box:

Group 6 - Portfolio

Alec VanderKolk, Ryan Hansen, Alan Tong, Samuel Forbes

Maintenance

To maintain this project, we used GitHub to open issues and track them using that built-in feature. We kept the design modular, so we could easily identify where the issues were coming from. To further sustain our product, we had Agile and weekly meetings to keep tabs on the files. Finally, we had user testing and Selenium to find bugs and maintain our product in that way.