Aaron Leenknecht

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	Yes. Repo was public and link was correct. The README file was precise in explaining how to start the program locally	No action necessary
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	Variable names were short/precise in necessary scenarios and long/descriptive in necessary scenarios. Comments through the project were hit and miss. Some files were extremely well commented and others had zero comments. The api test file had zero comments in it and had a giant block of code that was commented out. Commenting needs to be done in the client/src directory. The code seems to correctly follow guidelines and style	We do not think test files need comments; comments are to understand code so you can edit it but tests should not be changed. They are there to make sure new additions do not affect old code. As for the other files, we read through them and added more comments.
Implementation	Is it shorter/easier/faster/cleaner/safer to write functionally equivalent code? Do you see useful abstractions?	In the app.js file in the directory: client/src, some of the if statements can be written to where not every if needs to be called independently. I think their code is well written to achieve safety as well as efficiency. The above mentioned comment about re-writing if statements is such a marginally effective piece of code in terms of computation time.	This refers to a point where there is one “if else” construct next to another “if else” construct. These should be separate as they are; they are separate checks. No action is necessary.
Maintainability	Are there unit tests? Should there be? Are the test covering	They seem to have tests written for the entire project. I’ve looked	As mentioned, we do not believe tests need comments. No other

	interesting cases? Are they readable?	through the test files and it achieves a very high code coverage. Tests are readable but most aren't commented for maintainability. Their client asked for unit tests so the tests they have written are needed and I believe they achieve what is being asked.	action is necessary.
Requirements	Does the code fulfill the requirements?	The code seems to fulfill all requirements except for the possibility of one. Section 3.4 of their requirements document says that 100 users will be online and asked if the application slows down at all. I'm not sure if that was tested successfully or not and don't have the means to personally test that requirement.	We found it difficult to test this requirement as we did not have access to 100 simultaneous users. We contacted our client in regards to adjusting this.
Other	Are there other things that stand out that can be improved?	I think there should be more emotion labels to choose from. There were several videos that I watched where the appropriate emotion I was feeling was not available. A neutral label was given for that scenario but that's not getting all the data, making it slightly biased. I know those emotions were given to the group by their client which makes my previous comments outside the scope of their project at this particular time.	The client has provided the list of emotions; this was not our decision.

Khoa Tran

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	Yes, but not without a lot of troubleshooting. Node 14 will not work for connecting to the Postgres DB. The required Node version is specified in 'package.json' to be 10.15.3, but it may be good to mention this in the README. Node 12 and 13 work, but give a warning which I have communicated to the team on Slack. Node 10 is supported until April 30, 2021, so the team and client may consider targeting Node 12, the latest LTS version.	This is due to an error with versioning of Node and NPM. Our project uses 10.15.3 and 6.4.1 respectively; more recent versions conflict with version 7.17.1 of pg, the node-postgres package we use. Those running MacOS were more likely to have Node 14, which did not work; not everyone on MacOS had Node 14 and those with earlier versions succeeded. We decided not to update all of these packages because support for them extends past the life of this project and there are no vulnerabilities associated with them. We added the versions to the README to avoid confusion.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	The code is generally easily to follow. Consider using async/await instead of .then() when possible for more legible and modern code. I found a couple of variables that could be better defined, such as line 56. "letters" can be changed to "validChars" or "validPattern" for better *self-commenting code*. Another example is line 99-100. I'm not sure what the purpose of having two variables "lowerUsername" and "currentUsername" is, because the first one is assigned to the second and then never used again.	Our group chose to use promises in the .then().catch() format over using async/await because the two are functionally equivalent and we think that promises are more readable.
Implementation	Is it shorter/easier/faster/cleaner/safer to write functionally equivalent code? Do you see useful abstractions?	Code can use a more consistent and standardized style. Consider using a JS formatter like prettier to properly indent and insert spaces in your code. VS Code has a built-in formatter. My favorite settings	We went through the client side code and formatted the spaces better.

		are “format on save”, “format on type”, and “format on paste”. These tell the editor to format your code automatically, and often.	
Maintainability	Are there unit tests? Should there be? Are the test covering interesting cases? Are they readable?	Unit tests cover reading and inserting users, votes, videos, and labels. They could use comments detailing what they do. However, they make excellent use of async/await and are very well formatted.	As mentioned, we do not believe tests need comments. No other action is necessary.
Requirements	Does the code fulfill the requirements?	The user-side requirements are met. Data such as users, videos, and votes were loaded correctly. Obviously, we were not able to test client-side (Xandr) requirements such as ensuring data integrity and generating reports.	No action required.
Other	Are there other things that stand out that can be improved?	As mentioned in the “Build” section, as a long-term improvement I would ensure the code’s compatibility with later versions of Node. In the meantime, mention the target Node version in README.	Later versions of node are beta, and not LTS yet. No action required.

Aidan Grimshaw

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	I was able to successfully build the website using the instructions provided in the README. The install writeup was both concise and enough to get the app going.	No action necessary.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	I noticed that the code on Heroku was printing possibly important information to the console, including a list of all the usernames when the user tries to log in. I would remove those print statements along with all others in your production branch. Additionally, I would remove the comic sans font in your header and replace it with the font that is used in the rest of your project. I liked the extensive and well-structured comments throughout the project.	We agreed that it should not print those things, and fixed it. We also agreed about the font, and changed it.
Implementation	Is it shorter/easier/faster/cleaner/safer to write functionally equivalent code? Do you see useful abstractions?	In the login process, it's probably not a good idea to be requesting all the users from the database and then searching through them to check if there is a match with the current user's username. This isn't very performant and the request will fail with any large number of users. Instead, I would suggest a statement similar to "SELECT 1 FROM someTable WHERE(username=someUsername) THEN TRUE	We agree. This has been changed.

		ELSE FALSE”	
Maintainability	Are there unit tests? Should there be? Are the test covering interesting cases? Are they readable?	There are unit tests for the frontend. They seem to be covering most of the important cases as far as rendering correctly. There are also tests for the backend that seem to test endpoints comprehensibly. Unit tests are well commented and readable	No action necessary.
Requirements	Does the code fulfill the requirements?	The code fulfills the requirements. It seemed like a lot of the requests timed out when I was testing it, but my internet is being really bad so I’m not sure if It’s the site.	No action necessary.
Other	Are there other things that stand out that can be improved?	I think it would be cleaner, less error prone, and less confusing if the client and the backend were separated out into separate repos or folders instead of the client being in a subdirectory of the backend	The client-side must be a subdirectory of the backend; our implementation requires it. The client-side is actually not a separate entity, but its files are served by the back end.