CPEN321
Media Recommendation Engine

# Reflection

## Development: Front End

The front end team decided to construct the web application purely from scratch. The major advantage doing this was that we had full control over every component of the application. A downside to that was we spent a lot of time constructing all the HTML and CSS styling.  If we initially decided to use a template to build up from, we would have saved some time and used that time to flush out our implementations of our use cases. At the beginning of the project, we implemented naming conventions for all of the files and functions to make it easier to find things as the project grew in size. Throughout the project we were able to stick to the naming conventions and it made it very easy to find exactly where a file or a function was.  There were some controller decisions that we committed to which we decided may have not been the proper method to develop in. Since all of us were new to this development stack, we should have done more research or looked at other projects which follow a well planned outline.

## Testing: Back End

During our time working on the server, we found that manually testing our routes with Advanced REST Client and Postman as we implemented them gave us sufficient information to make decisions on whether a certain request behaved appropriately and whether we could push it to live. Additionally utilizing console logging with these clients was effective in isolating and fixing any bugs. Integration testing in this case was less important as a certain route would usually only affect a single part of a document on our database which makes it highly unlikely that a new route breaks something in an old route. However, as we started to create more complicated scenarios to test, we realized that we had to string together many requests to set it up. Manual testing made this difficult to manage and time consuming. We wish that we had started earlier on designing a collection of test requests in Postman so that we could automate these processes as it was difficult to pull anyone from what they were doing (routes, algorithms, third party APIs, etc) during the later development stages of the project. We also could have used the extra time to define a successful test and a failed test as we found it difficult to automatically distinguish between a correct response object from an incorrect object without actually looking at and reading the payload ourselves. At a minimum we would have liked to have a collection of tests to run through every possible response in every route, disregarding what the response was and just looking at the HTTP status codes.

## Design Pattern:

Our web application implemented the model-view-controller design pattern to communicate between the user and the server. As most web applications that provide the user information based off their inputs, we decided this design pattern would be ideal. Our controller framework of choice for this project was Angular. Since Angular and Node both are JavaScript frameworks, it was efficient for the back and front end to communicate their design decisions.

To accompany our client-server design pattern for this project, the server uses a RESTful API design pattern in order to connect the frontend to the backend and database. This design pattern is very modern and we found that it fits our application perfectly given the scope of our project, resources available and time constraints of our development life cycle. Having a design pattern for our API allowed all backend developers to be on the same page when it came to designing routes and gave some structure to the code in an otherwise hectic development process. This structure also allowed the backend to be much more efficient in communicating to the frontend about how to utilize the API. The consistency of RESTful design also allows the frontend to infer what operations a new route does or even where a new route for a requested feature would exist. We do not believe we would change the design pattern of the server if we were to do this project again however we do believe we could have organized and/or split up some of the routes more in order to make it clearer in which file certain database operations should exist.