**Dev Resources**

**Creators: Ryan Hardin, Johnny Hinton, Jerry Phan**

**Table of Contents**

1. **Introduction / Background** ------------------------------------ 3
2. **Class’ Technical Requirements** ----------------------------- 3
3. **External API** ------------------------------------------------------- 3
4. **Internal API** -------------------------------------------------------- 3
5. **Functionalities** ---------------------------------------------------- 4
6. **Design Documents** ---------------------------------------------- 5
7. **Implementation Documentation** ---------------------------- 6
8. **Deployment Documentation** -------------------------------- 6
9. **Test Cases and Test Reports** ------------------------------- 6
10. **Lessons Learned** ------------------------------------------------ 6
11. **Acknowledgements and References** --------------------- 7
12. **Gitlab Repository ------------------------------------------------- 7**

* Introduction / Background (the problem that you are solving, intended users)
  + Devresoures is a platform which allows users to share, read, and comment their resources, projects, or ideas to other developers (new developers to experienced developers).
  + Intended users are CS students, teachers, and/or enthusiasts who share the same interest in CS.
  + The problem we are solving is that there are many CS people in the world and many either do not have ideas to start on a project or a way to share their ideas to others, DevResouces allows users to share, read, and comment with other CS people with ease.
* Class’ technical requirements
  + Class

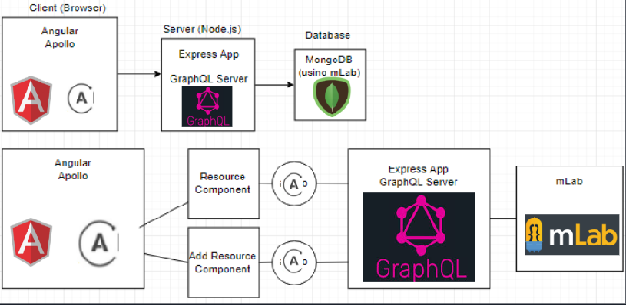
MySQL

Vagrant with Virtual Box

Ubuntu to run Virtual Machines  
LAMPython Stack

Our Project we used\*\*  
MEAN Stack  
MongoDB w/ mLAB for database  
Express - set up server  
Apollo for query handling  
Node.js for scripting in front and backend  
GraphQL- set up server

* External Api
  + **We initially were going to use Reddit’s API because it would allow us to access the user submitted and rated stories from reddit.com. It also provides advanced functionality, including user account information and sub-reddit moderation.**
* Internal Api
  + **GraphQL**
    - **Displays the data on the GraphQL server @ localhost:3000/graphiql**
    - **With the graphiql interface, the user is able to test out the queries and mutations that they have created to insure proper functionality when it comes to sending GET or POST requests.**
    - **With a proper query setup, the user will receive an easy to read layout of the data that is stored within their provided database, or if they want to use a ‘mutation’ they are able to create new data to be stored into the database without the need for a client side request.**
  + Use database to persist data
    - * **MongoDB w/ mLab**
  + Your application’s requirements
    - * **Node.js, Vagrant, Yarn,** **Angular CLI**
* Functionalities
  + Inputs, outputs, processing
    - **And example of the functionality in our application would be on the resources page where we have a new post sharing button and a list of all current resources that are stored into the database. On initialization of the page, an array is filled with all of our resources in the database, then recursively placed onto the page with the information being passed through the newPage function.**
    - **When the client side calls to have the array be filled with all resources, it uses a query that is set up in a similar way to the backend. The response from the graphql server populates the array with its response.**
* Design documents
* Requests and Routes to web
  + Client(browser) which will be our website (DevResources) we created (using Angular and apollo) will request a query (for a component) to our server (Express and GraphQL). And by using Apollo, it will create the query and bind it to the component. It will then send it to the server for it to handle and the server will grab data from the database (we used MongoDB with the use of mLAB), and return it back to the server and server will send the query back to component and client.
* Responses from web
  + I think explained with request and routes above ^  
    Also for the two points above, add the diagram I created in powerpoint slides to make report look better, ( if it doesnt show on this page, go to powerpoint slides screenshot diagram, copy and past on paint and crop diagram from screenshot)



* Database schema
  + **The schema of the database was based on a general idea of how a type forum based application is set up. Where for example a User will come with specific requirements such as a username, email, and password. Without these the database would not accept the information due to it not meeting the requirements given to the client side by the provided schema.**
* Implementation documentation
  + Project file tree

Image can be used, maybe gitlab? idk

* Deployment documentation
  + Configuration scripts
* Test cases and test reports
  + Tested website functionality with use of a local server
  + Tested the functionality of sharing resources - Functioned properly
  + Tested the functionality of commenting on resources
  + Tested Registration - Functioned, but would not redirect to main page or login due to time constraints
* Lessons learned
  + Jerry:
    - How to successfully create a web app, design it while it functioning properly.
    - Learned how to use GraphQL and how it performs with other tools such as Angular, Express, and Apollo.
    - With major time constraints due to group problems, time constraints with classes and basically full time jobs..learned to adapt and worked well efficiently and still made time to meet successfully for the project.
  + **Johnny:**
    - **GraphQL and its interaction with Angular was more time consuming to learn and get to perform correctly than initially expected. In the process of learning GraphQL I got the chance to learn more about how its querying system works and its interaction with the client side of applications. On top of that, learning that it was somewhat of a better version of the REST api after reading up on some articles on compare/contrast between the two.**
    - **Do not put too much faith in one single member when it comes to getting certain tasks done, it may place a burden on other members in the long run if something is to happen without notice.**
  + **Ryan**
    - **Looking back I think we should have taken the approach to work with backend technologies we are all far more familiar with than just one person. The frontend of the project was the easiest part but the backend logic is what kept us from completing the project.**
    - **We didn’t manage our time well with the workload needed and that caused us to panic and do as much as we could at the last minute.**
    - **Even though this project didn’t turn out the way we thought, I did enjoy working with both Jerry and Johnny. If we ever were having trouble or one of us had a question, we did an excellent job working together to find a solution.**

* Acknowledgements and references
  + [MongoDB Hosting Site](https://mlab.com/)
  + [apollo-graphql Documentation](https://www.apollographql.com/docs/) ( types, schema setups, graphql-tools, etc.)
  + [Mongoose Documentation](http://mongoosejs.com/docs/)
  + [AngularJS API](https://docs.angularjs.org/api)
* Repository
  + [**https://gitlab.cs.uab.edu/smso3223/CS421-FinalProject**](https://gitlab.cs.uab.edu/smso3223/CS421-FinalProject)