CS 410 Project Source Code Documentation

How to use our application:

Our application was built using NodeJS and React for our backend and frontend implementations, respectively. The user must first clone our project in Visual Studio Code to access our code. To run our code, two separate terminals will have to be opened by the user. In the first terminal, the user will have to type in the command ```cd front-end```. From there, the user can type ```npm install``` to install the React packages, and subsequently ```npm start``` to start the React application. This will result in a localhost:3000 webpage to open containing our application. In the second terminal, the user must type in the command ```npm install -g nodemon``` and then ```nodemon --exec node back-end/server.js``` to run the backend of our application. Upon successful access, the user should see "Server running on port 3002" in the terminal. Note: for Windows users who may still not be able to run nodemon, please refer to https://stackoverflow.com/questions/63423584/how-to-fix-error-nodemon-ps1-cannot-be-loaded-because-running-scripts-is-disabl for a solution to successfully run nodemon. The front page of our web application contains four input fields. The first is the name of the YouTube channel to search. We limited the YouTube channel a user could access in this application to one: SpaceX. The user should input "SpaceX" into the first input field. The second and third fields of our application are the start date and end date for the query, representing the years upon which the search will be conducted. The start date must be closer to 2022 than the end date. For example, a valid start date would be "2022" and a valid end date would be 2020". Finally, the final input field can be any word that the user wants to input, such as "music". The user must enter the inputs into the fields without the quotation marks for successful output. Once the user has entered these values into the input fields, the user should click on the "Search" button. Once this button is clicked, the user will be redirected to the Search Engine page. Once the application is done running, which may take up to 15 minutes, a table will be displayed on this page showing the links to the videos with transcripts that contain the query searched over the specified time period and the specific time that query occurred in that video's transcript.

How the software is implemented/How our application works: Our application is built upon a multi-stage system to find instances of a given query in the transcripts of the videos of a YouTube channel over a given period of time. After the user has input values into each of the fields and clicks the "Search" button, the values from the input fields are passed to a web scraper, which operates over YouTube, to find all videos that contain transcripts from that specific YouTube channel over the specified time period. The web scraper passes these results to a SQLite database, which stores the link to the video, the start and end dates, and the transcripts from the videos, which include timestamps. The database is then accessed by the search engine, which ranks the videos from newest to oldest. The search engine then interacts with the front end of the application and displays a table with all of the data regarding the videos that contain the query, as well as the timestamps of the occurrence of that query within the transcripts of those videos.

Team Member Contributions:

Note: In many instances, we worked as a group on multiple functionalities via Visual Studio Code's LiveShare functionality. At the end of the sessions, the LiveShare host committed all our changes to Github. This helped us avoid any merge conflicts on Github

- Adnan: Frontend applications for query page and search engine page, making sure links work for each page, made sure that input fields passed data to web scraper, helped with search engine backend implementation and display, and assisted Matt with layout of web scraper.
- Heet: BackEnd applications, setting up SQL databases to perform POST requests to queries, and then parsing data from SQL to rank queries based on frequency.
- Anshul: BackEnd applications, setting up SQL databases to perform POST requests to queries, and then parsing data from SQL to rank queries based on frequency.
- Matt: Setup the web scraper and linking the frontend and backend. Integrated the web scraper with the search engine to display the correct transcript results to the front end.
- Vrush: Helped with frontend implementation, helped connect web scraper to database and database to search engine, helped add display functionality to search engine frontend.