# YouTube Clone Frontend Framework

## Introduction

In the first phase of the YouTube Clone project, we build out the backend for our application, creating an API that exposes access to Comments that are stored in a database using the unique videoId for each video those comments are being left on.

Now, we are moving into the second phase of the project. We will be using React to build a frontend client application that will consume both your backend API for comments as well as the YouTube Data API for video information and related videos.   
  
This frontend will increase the complexity of asynchronous programming, as multiple components will be dependent on data coming from external resources. We will also be utilizing starter code that incorporates React Router and the concept of users who are registered with our application to control who can utilize certain features!

## Technologies

React.js, HTML/CSS, React Router, JSON Web Token, Axios, YouTube Data API

## Learning Objective

The objective of this project is to leverage modern React features and multiple external data sources to create a wrapper around the YouTube Data API – creating a minimal alternative client that exposes only the critical features needed to enjoy videos!

💡 Be sure to watch all supporting content, including videos on using React Router and utilizing the starter code to add features, before diving into user stories!

## Resources

**Lectures**

* All previous React lectures
* Promises, Async/Await, Requests
* React & JWT
* React Router
* Custom Hooks & Dynamically Generating Components

**Documents**

* User Stories document – contains information and links relating to making requests to the YouTube Data API
* React Django Starter Code Setup Guide

**Relevant Projects**

* Music Library (both front and back end)

**Other Resources**

* **Video – Hiding Sensitive Data on Git**
* **Video –YouTube Clone Flex Project Walkthrough**
* **Video – React Router Demo**
* **Video – Conserving API Requests with 3rd Party APIs**

## Tasks

Build a React frontend that will have multiple pages to create a YouTube experience for your users.

1. **Utilize the links in the user stories to obtain an API key for the YouTube Data API and test the two requests in Postman.** Pay close attention to the documentation and be sure to note the possible params you could utilize in your requests (including snippet)
2. **Build and test the React Frontend** using React best practices (components organized into their own folders with external CSS stylesheets, lifting state up as high as possible in component hierarchy, passing data from parents to children as props, etc)

## Setup Steps:

*For Step 2 from Tasks*

1. Start by setting up the pages you’ll need. Your project will need at least one main page called YouTubePage that contains descendant routes for two other pages – SearchResultsPage and VideoPage (Be sure to look at the React Router demo for ideas on how you can connect them!)
2. Create a SearchBar component that should be visible no matter which page your application is currently loaded to!
3. Work on getting your search results from the YouTube Data API to appear on your SearchResults page. Make sure to examine the data coming back closely. How can you utilize a thumbnail image to represent the video in question?
4. Once SearchResults are working, create an onClick event function handler to allow clicking a video to navigate to the VideoPage, where the embedded player will show the YouTube video selected.
5. Once the YouTube video is working correctly in the iframe component, work on getting Related Videos to display. Note that the same videoId you are using to grab the selected video will also be used to find related videos. How can a React Router route be set up to pull this information from a URL param?
6. After related videos are properly displaying, begin working on comments . This is trickier than it seems at first so be patient and problem solve! Remember that we want to lift state as high as possible in our component hierarchy. The videoId for your video will be used to attach the comment to the correct video, and the user who left the comment will be sent back via the JSON Web Token. **Only logged in users should be allowed to add a comment!**

## End Result

Please see the YouTube Clone Flex Project Walkthrough video to get an idea of your end result!