A PROJECT REPORT

on

“My YouTube: Cloning Google’s YouTube Product”

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR’S DEGREE IN COMPUTER SCIENCE

BY

SHUBHANGI SUMAN (ROLL NO: 1605155)
PRATIL DUBEY (ROLL NO: 1605139)

UNDER THE GUIDANCE OF

PROF. ANIL KUMAR SWAIN

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024
April 2020

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

“My YouTube: Cloning Google’s YouTube Product“

submitted by
Pratil Dubey
Shubhnagi Suman

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: 04/04/2020

(Prof. Co-Guide Name)	(Prof. Guide Name)
Project Co-Guide	Project Guide

Acknowledgements

I am profoundly grateful to Prof. Anil Kumar Swain for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

Shubhangi Suman
Pratil Dubey

ABSTRACT

This is a learning and research based project using React JS to understand what goes behind making a real time working web application like Google's most popular product YouTube. We started from learning React JS which is a JavaScript library for building user interfaces. It is developed and maintained by Facebook and Community of developers. It is a single page tool used for building UI (User Interface) Components. It contains components and state management where the former is nothing but JavaScript functions that can be reused and the later refers to the management of the state of one or more user interface controls such as text fields, buttons, radio buttons, etc. in a Graphical User Interface (GUI).

We also learnt about Google API's from which we can fetch all the real time data of Google's YouTube database. All results on our application were the same as the ones we get by searching keywords directly on YouTube. With the YouTube Data API, a variety of YouTube features were added to the application.

The API receives a http request from GET www.googleapis.com/YouTube/v3/search and returns JSON data as a response. Thus, the API enables us to upload videos, manage playlists and subscriptions, update channel settings, and much more.

The API was used to search for videos matching specific search terms, topics, locations, publication dates, and much more. The APIs search.list method also supports searches for playlists and channels.

We used Axios with React to make requests to an API, return data from the API, and then do things with that data in our React app. Axios is a popular HTTP client that allows us to make GET and POST requests from the browser. Axios is a promise-based async/await library for the readable asynchronous code. And we can easily integrate it with React.js, and effortlessly use it in any frontend framework.

The idea is to build a Simple YouTube Clone project in ReactJS. It uses YouTube API in order to fetch videos and display it to the users. Here, you can search for any videos you want to watch.

This react application acts as a User Interface to display the data from YouTube database where you can search for your favourite topics ranging from Hollywood movie trailers and music videos, to amateur vlogs (video blogs). Not only it gives you the video which matches the keyword but it also gives some more suggestions related to your searched keyword. You can directly play any of the suggestions with a single click. Thus, it was a great learning opportunity to work on the most recent and highly used UI building tool and learning about Google's YouTube API.

Keywords: React JS, YouTube, JSX, Google API, Single Page Applications

Contents

	Title	**Page**
1	Introduction	7
2	Literature Survey	8
3	Requirement Analysis	9
4	Technologies Used	10
5	Basic concepts	12
6	JSX concepts	15
7	Background Study	20
6	System Design: Hierarchy of react components	34
7	Project analysis and Implementation	36
8	Results	38
9	Conclusion and Future work	39
10	References	40

List of Figures

1.1 Hierarchy of react components 2

1.2 Project design and implementation 9

Chapter 1

Introduction

The idea is to build a Simple YouTube Clone project in ReactJS. It uses YouTube API in order to fetch videos and display it to the users. Here, you can search for any videos you want to watch.

It uses Google YouTube v3 API from which it can fetch all the real time data of Google's YouTube database. All results on our application were the same as the ones we get by searching keywords directly on YouTube. With the YouTube Data API, a variety of YouTube features were added to the application. The API receives a http request from GET www.googleapis.com/YouTube/v3/search and returns JSON data as a response. Thus, the API enables us to upload videos, manage playlists and subscriptions, update channel settings, and much more.

The API was used to search for videos matching specific search terms, topics, locations, publication dates, and much more. The APIs search.list method also supports searches for playlists and channels.

This react application acts as a User Interface to display the data from YouTube database where you can search f or your favourite topics ranging from Hollywood movie trailers and music videos, to amateur vlogs (video blogs). Not only it gives you the video which matches the keyword but it also gives some more suggestions related to your searched keyword. You can directly play any of the suggestions with a single click.

Chapter 2

Literature Survey

YOUTUBE:

YouTube is a video sharing service that allows users to watch videos posted by other users and upload videos of their own. The service was started as an independent website in 2005 and was acquired by Google in 2006. Videos that have been uploaded to YouTube may appear on the YouTube website and can also be posted on other websites, though the files are hosted on the YouTube server.

The slogan of the YouTube website is "Broadcast Yourself." This implies the YouTube service is designed primarily for ordinary people who want to publish videos they have created. While several companies and organizations also use YouTube to promote their business, the vast majority of YouTube videos are created and uploaded by amateurs.

YouTube videos are posted by people from all over the world, from all types of backgrounds. Therefore, there is a wide range of videos available on YouTube. Some examples include amateur films, homemade music videos, sports bloopers, and other funny events caught on video. People also use YouTube to post instructional videos, such as step-by-step computer help, do-it-yourself guides, and other how-to videos. Since Google offers revenue sharing for advertisement clicks generated on video pages, some users have been able to turn YouTube into a profitable enterprise.

While YouTube can serve a business platform, most people simply visit YouTube for fun. Since so many people carry digital cameras or cell phones with video recording capability, more events are now captured on video than ever before. While this has created an abundant collection of entertaining videos, it also means that people should be aware that whatever they do in public might be caught on video. And if something is recorded on video, it just might end up on YouTube for the whole world to see.

Chapter 3

Requirements Analysis

Below are the following requirements that you'll need to follow in order to create a project like this or to run this project:

- **Node js and npm (node package manager):**

Node.js actually provides a runtime environment to execute JavaScript code from outside a browser. NPM, the default package manager for Node Js is used for managing and sharing the packages for any JavaScript projects. React uses Node.js and NPM for the management of dependencies and runtime.
We can install node js form the official website **https://nodejs.org**
Then we just have to follow the instructions on the installation dialog box and node js is installed. To check if the installation was successful or not you can run this command in your terminal/command prompt:
node –v and ***npm –v***

- **Create-react-app tool:**

After you're done with the installation of Node and npm, Now we need to install a tool named create-react-app using NPM as global. This tool is used to create react applications easily from our system.
npm install -g create-react-app
And then to create a react app we need to run this command:
create-react-app my-project
Then in react we have a file named: *package.json* where we have to write all names of the dependency and library we need to run the react app. To install the library mentioned we need to run the command ***npm install***.

- **To fetch api data from Youtube API**

1. Head over to the Google developers console
2. Create a new project by clicking on Select project drop down right next to the logo. Click the New Project button an give it a speaking name.
3. Select your project by choosing it in the Select Dropdown directly next to the logo in the header.
4. Click the Enable APIs and Services button
5. Search for youtube data
6. Click on the Youtube Data API v3
7. Click the blue enable button
8. In the dashboard, click Credentials on the left sidebar
9. Click the Create Credential button
10. Which API are you using: Youtube Data API v3
11. Where will you be calling the API from: Web browser
12. What data are you accessing: Public data
13. Click the What credentials do I need button
14. Copy the KEY, and paste it into src/api/youtube.js

- **Run the project**

To run the project we need to write the command ***npm start*** it automatically opens in a default browser at http:localhost:3000 by default, and whenever you change something in the code and save it it automatically reloads the page and renders the new code.

Chapter 4

Technologies Used

➢ REACT JS:

ReactJS is an open-source, component-based front end library responsible only for the view layer of the application. It is maintained by Facebook.

- React is a JavaScript library for building user interfaces.
- React is used to build single page applications.
- React allows us to create reusable UI components.

React uses a declarative paradigm that makes it easier to reason about the application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.
A react application is made of multiple components, each responsible for rendering a small, reusable piece of HTML. Components can be nested within other components to allow complex applications to be built out of simple building blocks. A component may also maintain internal state – for example, a TabList component may store a variable corresponding to the currently open tab.
NOTE: React is not a framework. It is just a library developed by Facebook to solve some problems that we were facing earlier.

➢ DOM:

DOM is (The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.).The real DOM is slow. So, to make it faster, React implements a virtual DOM that is basically a DOM tree representation in Javascript. So when it needs to read or write to the DOM, it will use the virtual representation of it. Then the virtual DOM will try to find the most efficient way to update the browser’s DOM.
Unlike browser DOM elements, React elements are plain objects and are cheap to create. React DOM takes care of updating the DOM to match the React elements. The reason for this is that JavaScript is very fast and it’s worth keeping a DOM tree in it to speed up its manipulation.

➢ HTML:

HTML is the standard markup language for creating Web pages. It stands for HyperText Markup Language. It describes the structure of a Web page. It consists of a series of elements. HTML elements tell the browser how to display the content. HTML elements are represented by tags. HTML tags label pieces of content such as "heading", "paragraph", "table", and so on. Browsers do not display the HTML tags, but use them to render the content of the page.

➢ JAVASCRIPT:

JavaScript(often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.

➢ CSS:

- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- 3 Types of CSS Styles: Inline, External and Internal.
- We have used it for styling the react components of our app.

Chapter 5

Basic concepts

❖ React(react-dom):

The react-dom package provides DOM-specific methods that can be used at the top level of your app and as an escape to get outside of the React model if you need to.
This package serves as the entry point to the DOM and server renderers for React. It is intended to be paired with the generic React package, which is shipped as react to npm.
In React, for every DOM object, there is a corresponding "virtual DOM object." A virtual DOM object is a representation of a DOM object, like a lightweight copy.
A virtual DOM object has the same properties as a real DOM object, but it lacks the real thing's power to directly change what's on the screen.
Manipulating the DOM is slow. Manipulating the virtual DOM is much faster, because nothing gets drawn on screen.

reactDOM.render()
Components can be rendered to a particular element in the DOM using the React DOM library. When you render a JSX element, every single virtual DOM object gets updated.

ReactDOM.render(element, container[, callback])

Render a React element into the DOM in the supplied container and return a reference to the component (or returns null for stateless components).

If the React element was previously rendered into a container, this will perform an update on it and only mutate the DOM as necessary to reflect the latest React element.

❖ npm:

npm is the package manager for JavaScript and the world's largest software registry. It is used to Discover packages of reusable code.
Here, we have used npm as the command line client that allows developers to install and publish those packages.

Npm consists of three distinct components:

1. The Website
2. The Command Line Interface (CLI)
3. The registry

Few useful npm commands for our project are:

1. npm install -g create-react-app
   To install the react package globally we used -g with npm install command. Using create-react-app we will install the boilerplate of the react application.

2. npm create-react-app my-app
   This command will create your react application named my-app.

3. npm-start

   This would run our application in development mode. We can just navigate to http:localhost:3000 in any browser to preview our app live. The page will automatically reload whenever it detects any code change in the source files. Warnings and errors can also be seen in the console.

   Internally, npm start uses webpack dev server to start a dev server so that we can communicate with the same.

❖ Axios:

Axios is a popular HTTP client that allows us to make GET and POST requests from the browser.

Therefore, we can use Axios with React to make requests to an API, return data from the API, and then do things with that data in our React app.

Axios is a promise-based async/await library for the readable asynchronous code. We can easily integrate with React.js, and it is effortless to use in any frontend framework.

Eg: Let’s create a new file called API.js in which we’ll store our Axios configuration.

```
        import axios from "axios";
export default axios.create({
  baseURL: "https://randomuser.me/api/",
  responseType: "json"
});
```

The code inside API.js imports Axios and exports a new configured instance of it. It’s set up to use the RandomUser API as a base URL and also specify that we’d like JSON in return.

❖ Google YouTube v3 API:

With the YouTube Data API, you can add a variety of YouTube features to your application. Use the API to upload videos, manage playlists and subscriptions, update channel settings, and more.

Use the API to search for videos matching specific search terms, topics, locations, publication dates, and much more. The APIs search.list method also supports searches for playlists and channels.

HTTP request:

GET https://www.googleapis.com/youtube/v3/search

Response:

If successful, this method returns a response body with the following structure:

```
{
  "kind": "youtube#searchResult",
  "etag": etag,
  "id": {
    "kind": string,
    "videoId": string,
    "channelId": string,
    "playlistId": string
  },
  "snippet": {
    "publishedAt": datetime,
    "channelId": string,
    "title": string,
    "description": string,
    "thumbnails": {
      (key): {
        "url": string,
        "width": unsigned integer,
        "height": unsigned integer
      }
    },
    "channelTitle": string,
    "liveBroadcastContent": string
  }
}
```

Example:

This function searches for videos related to the keyword 'dogs'. The video IDs and titles of the search results are logged to Apps Script's log. This sample limits the results to 25.

```
function searchByKeyword() {
  var results = YouTube.Search.list('id,snippet', {q: 'dogs', maxResults: 25});

  for(var i in results.items) {
    var item = results.items[i];
    Logger.log('[%s] Title: %s', item.id.videoId, item.snippet.title);
  }
}
```

Chapter 6

JavaScript Concepts

Arrow Functions:

An arrow function expression has a shorter syntax compared to function expressions and does not bind its own this, arguments, super, or new.target. Arrow functions are always anonymous. These function expressions are best suited for non-method functions and they cannot be used as constructors.

Features:

- JavaScript arrow functions are anonymous functions.
- Arrow functions cannot be used as constructor functions, (ie: with the keyword new)
- Lexical binding of this inside arrow function: The value of this inside an arrow function always points to the same this object of the scope the function is defined in, and never changes.

Syntax:

For starters, arrow functions in JavaScript are always **anonymous**, so the first thing it sheds is any function name. It also does away with the "function" keyword and uses an arrow (=>) to separate the parameter(s) portion of the function from the function BODY. The result in its most basic form is the following:

Example:

```
Hello  = () => {
  return "Hello World!";
}
```

Arrow Function with Parameters:

Example:

```
hello = (val) => "Hello " + val;
```

Arrow Function without Parenthesis:

Example:

```
hello = val => "Hello " + val;
```

About this:

1. The handling of this is also different in arrow functions compared to regular functions.
2. In short, with arrow functions there are no binding of this.
3. In regular functions this keyword represented the object that called the function, which could be the window, the document, a button or whatever.
4. With arrow functions this keyword *always* represents the object that defined the arrow function.

No binding of this in arrow functions:

Unlike a regular function, an arrow function does not bind this. Instead, this is bound lexically (i.e. this keeps its meaning from its original context).

Characteristics of arrow function:

- For arrow functions with a single parameter, you can omit the parenthesis () around the parameter.
- For arrow functions with no or multiple parameters, wrap the parameters in parenthesis.
- For arrow functions with a single BODY statement, you can omit the braces {} around the statement. The value derived from the statement is automatically returned with no braces.
- For arrow functions with multiple BODY statements, wrap them in curly braces. No value is automatically returned with braces- use the return statement to specify the value.
- If a function BODY contains only a single object literal, wrap the function BODY in parenthesis () to differentiate it from the object literal wrapper.

React events:

An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.

Handling events with react have some syntactic differences from handling events on DOM. These are:

1. React events are named as **camelCase** instead of **lowercase**.
2. With JSX, a function is passed as the **event handler** instead of a **string**. For example:

Example:

```
<button onClick={showMessage}>
      Hello JavaTpoint
</button>
```

Constructor in React:

The constructor is a method used to initialize an object's state in a class. It automatically called during the creation of an object in a class.

The concept of a constructor is the same in React. The constructor in a React component is called before the component is mounted. When you implement the constructor for a React component, you need to call **super(props)** method before any other statement. If you do not call super(props) method, **this.props** will be undefined in the constructor and can lead to bugs.

Syntax:

```
Constructor(props){
     super(props);
}
```

In React, constructors are mainly used for two purposes:

1. It used for initializing the local state of the component by assigning an object to this.state.
2. It used for binding event handler methods that occur in your component.

React map:

A map is a data collection type where data is stored in the form of pairs. It contains a unique key. The value stored in the map must be mapped to the key. We cannot store a duplicate pair in the map(). It is because of the uniqueness of each stored key. It is mainly used for fast searching and looking up data.

A map is not the feature of React. Instead, it is the standard JavaScript function that could be called on any array. The map() method creates a new array by calling a provided function on every element in the calling array.

In the given example,

The map() function takes an array of numbers and double their values. We assign the new array returned by map() to the variable doubleValue and log it.

```
var numbers = [1, 2, 3, 4, 5];
const doubleValue = numbers.map((number)=>{
   return (number * 2);
});
console.log(doubleValue);
```

Spread Operator:

Spread operator allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in variable array where there is more than 1 values are expected.It allows us the privilege to obtain a list of parameters from an array. Syntax of Spread operator is same as Rest parameter but it works completely opposite of it.

Syntax:

var variablename1 = [...value];

In the above syntax, **…** is spread operator which will target all values in particular variable. When … occurs in function call or alike,its called a spread operator. Spread operator can be used in many cases,like when we want to *expand,copy,concat,with math object*.

Conditional rendering in react:

There’s more than one way to use conditional expressions in React. And, as with most things in programming, some are better suited than others depending on the problem you’re trying to solve.

This tutorial covers the most popular conditional rendering methods:

- If/else
- Prevent rendering with null
- Element variables
- Ternary operator
- Short-circuit AND operator (&&)
- Immediately invoked function expressions (IIFEs)
- Subcomponents
- Higher-order components (HOCs)

JSX

- **Consider this variable declaration:**

const element = <h1>Hello, world!</h1>;

It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React "elements".

- **Why JSX?**

React embraces the fact that rendering logic is inherently coupled with other UI logic: how events are handled, how the state changes over time, and how the data is prepared for display.

Instead of artificially separating technologies by putting markup and logic in separate files, React separates concerns with loosely coupled units called "components" that contain both. We will come back to components in a further section, but if you're not yet comfortable putting markup in JS, this talk might convince you otherwise.

React doesn't require using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.

- **Embedding Expressions in JSX**

You can put any valid JavaScript expression inside the curly braces in JSX. For example, 2 + 2, user.firstName, or formatName(user) are all valid JavaScript expressions.

In the example below, we embed the result of calling a JavaScript function, formatName(user), into an <h1> element.

```
function formatName(user) {
        return user.firstName + ' ' + user.lastName;
}
const user = {
        firstName: 'Harper',
        lastName: 'Perez'
};
const element = (
        <h1>
          Hello, {formatName(user)}!
        </h1>
);
ReactDOM.render(
        element,
        document.getElementById('root')
);
```

We split JSX over multiple lines for readability. While it isn't required, when doing this, we also recommend wrapping it in parentheses to avoid the pitfalls of automatic semicolon insertion.

- ## JSX is an Expression Too

After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions.

- ## Specifying Attributes with JSX

You may use quotes to specify string literals as attributes:

```
const element = <div tabIndex="0"></div>;
```

You may also use curly braces to embed a JavaScript expression in an attribute:

```
const element = <img src={user.avatarUrl}></img>;
```

Don't put quotes around curly braces when embedding a JavaScript expression in an attribute. You should either use quotes (for string values) or curly braces (for expressions), but not both in the same attribute.

Since JSX is closer to JavaScript than to HTML, React DOM uses camelCase property naming convention instead of HTML attribute names.

For example, class becomes className in JSX, and tabindex becomes tabIndex.

- ## JSX Represents Objects

Babel compile JSX down to React.createElement() calls.

These two examples are identical:

```
const element = (
        <h1 className="greeting">
          Hello, world!
        </h1>
);
const element = React.createElement(
        'h1',
        {className: 'greeting'},
        'Hello, world!'
);
React.createElement() performs a few checks to help you write bug-free code
but essentially it creates an object like this:
// Note: this structure is simplified
const element = {
        type: 'h1',
        props: {
                className: 'greeting',
                children: 'Hello, world!'
        }
};
```

These objects are called “React elements”. You can think of them as descriptions of what you want to see on the screen. React reads these objects and uses them to construct the DOM and keep it up to date.

Chapter 6

Background Study

Virtual Dom

- **Real DOM**

First things first, DOM stands for "Document Object Model". The DOM in simple words represents the UI of your application. Everytime there is a change in the state of your application UI, the DOM gets updated to represent that change. Now the catch is frequently manipulating the DOM affects performance, making it slow.

- **What makes DOM manipulation slow?**

The DOM is represented as a tree data structure. Because of that, the changes and updates to the DOM are fast. But after the change, the updated element and its children have to be re-rendered to update the application UI. The re-rendering or re-painting of the UI is what makes it slow. Therefore, the more UI components you have, the more expensive the DOM updates could be, since they would need to be re-rendered for every DOM update.

- **Virtual DOM**

That's where the concept of virtual DOM comes in and performs significantly better than the real DOM. The virtual DOM is only a virtual representation of the DOM. Every time the state of our application changes, the virtual DOM gets updated instead of the real DOM.

Well, you may ask" Isn't the virtual DOM doing the same thing as the real DOM, this sounds like double work? How can this be faster than just updating the real DOM?"

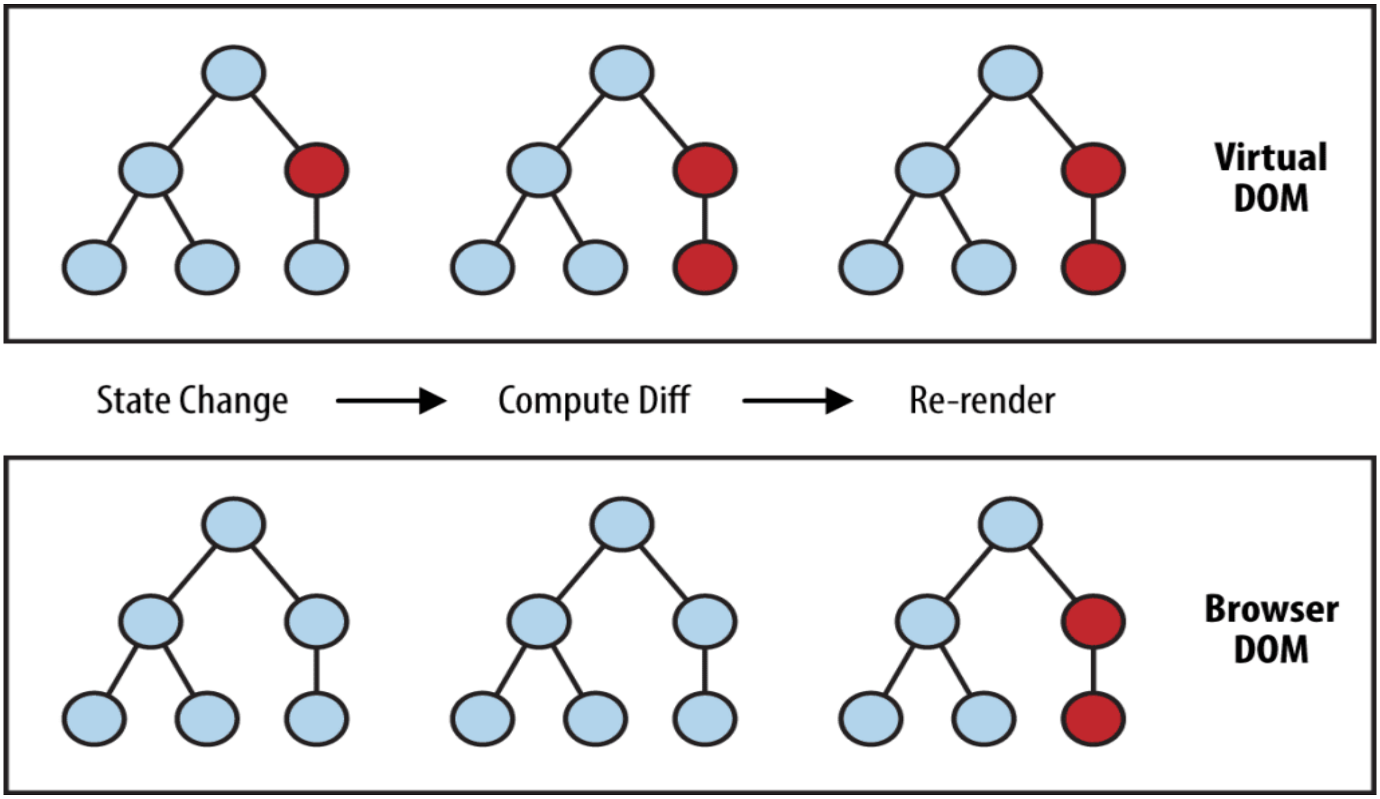
The answer is virtual DOM is much faster and efficient, here is why.

- **How is Virtual DOM faster?**

When new elements are added to the UI, a virtual DOM, which is represented as a tree is created. Each element is a node on this tree. If the state of any of these elements changes, a new virtual DOM tree is created. This tree is then compared or "diffed" with the previous virtual DOM tree.

Once this is done, the virtual DOM calculates the best possible method to make these changes to the real DOM. This ensures that there are minimal operations on the real DOM. Hence, reducing the performance cost of updating the real DOM.

The image below shows the virtual DOM tree and the diffing process.



Source: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch02.html

The red circles represent the nodes that have changed. These nodes represent the UI elements that have had their state changed. The difference between the previous version of the virtual DOM tree and the current virtual DOM tree is then calculated. The whole parent subtree then gets re-rendered to give the updated UI. This updated tree is then batch updated to the real DOM.

- **How does React use Virtual DOM**

In React every UI piece is a component, and each component has a state. React follows the observable pattern and listens for state changes. When the state of a component changes, React updates the virtual DOM tree. Once the virtual DOM has been updated, React then compares the current version of the virtual DOM with the previous version of the virtual DOM. This process is called “diffing”.

Once React knows which virtual DOM objects have changed, then React updates only those objects, in the real DOM. This makes the performance far better when compared to manipulating the real DOM directly. This makes React standout as a high performance JavaScript library.

In simple words, you tell React what state you want the UI to be in, and it makes sure that the DOM matches that state. The great benefit here is that as a developer, you would not need to know how the attribute manipulation, event handling or the manual DOM updates happen behind the scenes. All of these details are abstracted away from React developers. All you need to do is update the states of your component as and when needed and React takes care of the rest. This ensures a superior developer experience when using React.

- **React render() function**

render() is where the UI gets updated and rendered. render() is the required lifecycle method in React. You can learn more about React lifecycle methods in detail from my blog post.

render() function is the point of entry where the tree of React elements are created. When a state or prop within the component is updated, the render() will return a different tree of React elements. If you use setState() within the component, React immediately detects the state change and re-renders the component.

React then figures out how to efficiently update the UI to match the most recent tree changes.

This is when React updates its virtual DOM first and updates only the object that have changed in the real DOM.

- **Batch Update**

React follows a batch update mechanism to update the real DOM. Hence, leading to increased performance. This means that updates to the real DOM are sent in batches, instead of sending updates for every single change in state.

The repainting of the UI is the most expensive part, and React efficiently ensures that the real DOM receives only batched updates to repaint the UI.

Components

You can compare components with lego blocks. We use them as building blocks to build a bigger and meaningful application.

```
import React from 'react';
class MyComponent extends React.Component {
        render () {
                return <div> This is a component </div>
        }
}
```

Now we can use this component anywhere inside our react application by writing something like this

```
<MyComponent />
```

Yeah! just like a simple HTML tag. This is the beauty of React it uses your existing knowledge of HTML & Javascript with some of new and modern concepts like virtual dom & component based design.

One of the most interesting things React allows you to do is to use your custom React components inside another concept. Something like this

```
import React from 'react';
class MyComponent extends React.Component {
        render () {
                return <div> This is a component </div>
        }
}
class MyOtherComponent extends React.Component {
        render () {
                return (
                  <div>
                    <MyComponent />
                  </div>
                )
        }
}
```

This way you are able to compose more complex and useful user interface for your users. This feature also allow us to reuse component inside each other which encourages code reuse & easy to maintain code base.

- **Props & State**

React uses a render method which you have seen above to hold component views. Component’s render method return JSX which then use to create real HTML output which will be rendered in the browser.

The interesting Thing about render method is that it runs every time when your component State or Props updates. This process ensures whenever you change your data using application logic component’s view update to show new data to users. By using this simple mechanism React allows us to build powerful and almost easy to manage UI components.

- **State**

State is unique to your component. Every component has it’s own State where it store & retrieve data from. Let me show you a example

```
import React from 'react';
class MyComponent extends React.Component {
        constructor(props) {
                super(props);
                this.state = {
                  name: "Manoj"
                };
        }
        render () {
                return <div> My name is {this.state.name} </div>
        }
}
// if we render this component the output will be
My name is Manoj
```

Ignore the super(props) (out of the scope of this article) focus on this.state this is where our component state lives. We are basically creating a state in our constructor and then retrieving the data from state and using it inside our component.

You noticed *{this.state.name}*? If you are wondering what it is, this is how you embed a javascript expression inside your JSX. Whatever inside {} will be executed and its output is rendered in the browser. In case above JSX will retrieve data from this.state.name and embed it into our resulting html. Output will be something like this

```
My name is Manoj
```

React gives us a method which allow us to update the state and as you know whenever you update your state your component render method will be invoked resulting the browser to render updated data. Example.

```
class MyComponent extends React.Component {
 constructor(props) {
          super(props);
          this.state = {
            name: "Manoj"
          };
          this.changeName = this.changeName.bind(this);
 }
 changeName () {
          this.setState({
            name: "Your Name"
          });
 }
 render () {
                return <div onClick={this.changeName}> My name is
        {this.state.name} </div>
         }
}
```

In the above code we are telling our React component to call this.changeName whenever user clicks on the div. The this.changeName used this.setState which is provided by React to update your component state. When we click the div this.changeName change our name from “Manoj” to “Your Name”. As soon as component’s state updates the render process runs again but this time with new value of this.state.name which is “Your name” and our text changes from “My name is Manoj” to “My name is You Name”.
You can run this code on jsFiddle here.

- **Props**

Visualize props as options that can be passed to a component to customize its functionality.
For example, I have a heading component which renders a heading with subtitle.\

```
class MyHeading extends React.Component {
         render () {
                  return <div>
                    <h1>This is a heading</h1>
                    <p>Subtitle</p>
                  </div>
         }
}
```

If I will use this component it will always render same HTML, something like this.

This is a heading
Subtitle

If we use our component in this way it is not of much use right? Because it will always render the same HTML, for solving this problem React came up with props.

A way to pass data from one component to its child component. For example.

```
class MyHeading extends React.Component {
        render () {
                return <div>
                  <h1>{this.props.heading}</h1>
                  <p>{this.props.subtitle}</p>
                </div>
        }
}
// Now I can use this component like this
<MyHeading heading="Whoo! this is awesome" subtitle="And this is a subtitle" />
<MyHeading heading="Whoo! this is More awesome" subtitle="And this is second subtitle" />
```

The output will be

```
Whoo! this is awesome
And this is a subtitle
Whoo! this is More awesome
And this is second subtitle
```

This is great isn't it ? Now your component is reusable right. You can pass anything in your heading and subtitle and it will be rendered inside you MyHeading component.

You can define Props on your component just like you are setting a HTML attribute. Like we defined heading and subtitle above and then you can access them in your component by using this.props provided by React.

Whatever prop you define on your component you can access them as this.props.whateverNameYouGivenToYourProp. You can also pass Numbers, Functions & javascript expressions to your prop by using {}.

```
<MyHeading heading={1+1} subtitle={"this is a "+"subtitle"} />
```

Output will be

```
2
this is a subtitle
```

I hope this article helped you to understand React Components at more deeper level and you will use this knowledge to get started with React.

In the next article I will be talking about component life cycle & life cycle methods. If you want the article early please recommend and share this article among your fellow developers.

- **Element vs. component in React**

React is based on JavaScript and uses JSX (JavaScript XML) instead of HTML. JSX allows you to write HTML elements in .js files, and translates those HTML tags into React elements.

For example, a plain <h1> Title </h1> in HTML will look like this in JSX:

```
// This is a React element, not a component
```

```
const title = <h1> Title </h1>;
```

A React element is the smallest building block, and more elements put together form a component. So it’s element > component.

An element is basically what you want to see on the screen. In HTML, you would see the element displayed as soon as you write it in your .html file and hit save. This happens because you’re adding the node to the DOM.

In React you need to do the same thing, but you’re not affecting the actual DOM. Instead, you’re using the ReactDOM, which is a virtual DOM. A virtual DOM is a “copy” of the actual DOM, which holds updates made by the user, as they happen.

So if you want to add something to the ReactDOM, you need to use a specific method, called render().

Let’s say you want to render the title element from above. You call the method as follows:

```
ReactDOM.render(element, document.getElementById('id'));
```

The first parameter is the element, so what needs to be rendered. The second parameter is the container, so where it should be rendered in the DOM

Moving to components. A React component is a JS function which also returns the markup that we want to see (so the elements), but through classical function syntax.

```
function Component() {
        return <h1>This is the Heading</h1>;
}
```

When the component is created as a JS function, it is referred to as functional component.

A React component is rendered in the same way as an element, by using the render() method of the ReactDOM. However, instead of calling the function name, like we do in JS, we call the React component.

So we don’t write:

```
ReactDOM.render(Component(), document.getElementById('id'));
```

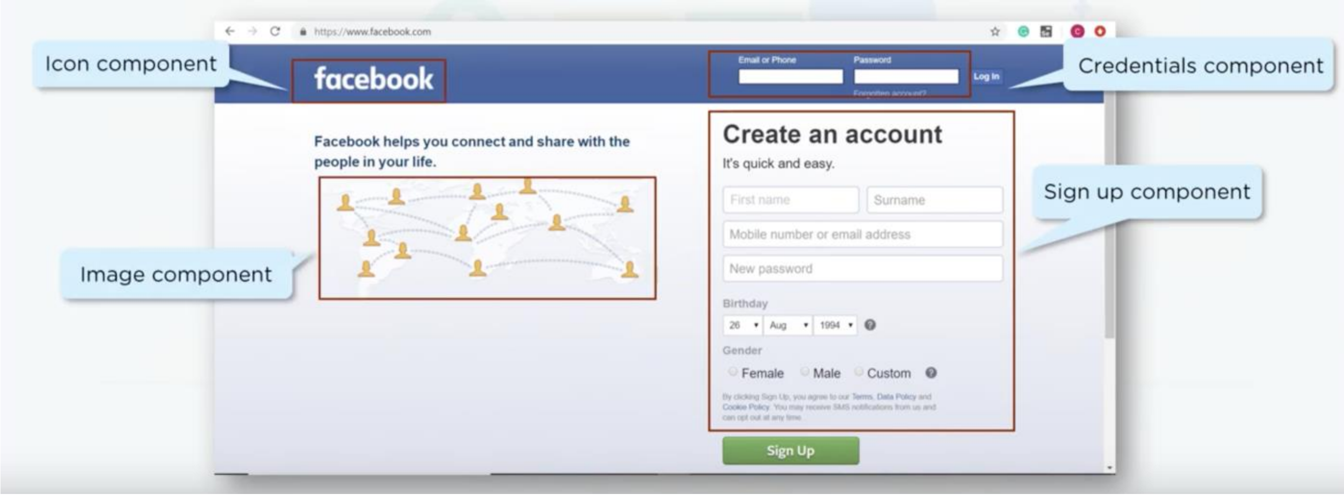
Instead, we write:

```
ReactDOM.render(<Component />, document.getElementById('id'));
```

So a React element uses regular HTML tags, while in a React component, the function name gives the name of the tag.

All React components must start with upper case letters.

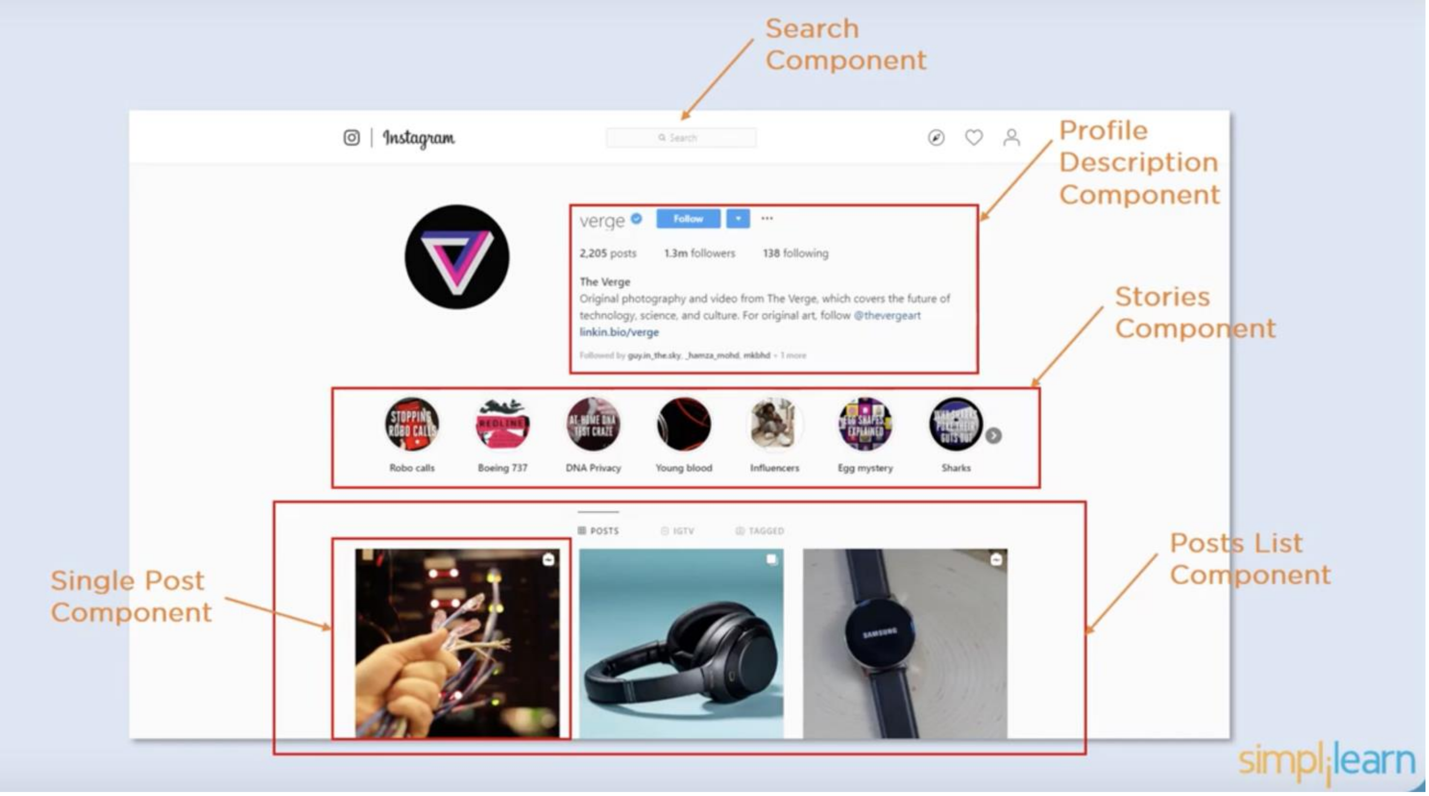
- **Examples**
  The websites decomposition into components:



Facebook helps you connect and share with the people in your life.

Create an account

It's quick and easy.



The Verge
Original photography and video from The Verge, which covers the future of technology, science, and culture. For original art, follow @thevergeart
linkin.bio/verge

Lifecycle Methods:

- **componentWillMount** is executed before rendering, on both the server and the client side.
- **componentDidMount** is executed after the first render only on the client side. This is where AJAX requests and DOM or state updates should occur. This method is also used for integration with other JavaScript frameworks and any functions with delayed execution such as **setTimeout** or **setInterval**. We are using it to update the state so we can trigger the other lifecycle methods.
- **componentWillReceiveProps** is invoked as soon as the props are updated before another render is called. We triggered it from **setNewNumber** when we updated the state.
- **shouldComponentUpdate** should return **true** or **false** value. This will determine if the component will be updated or not. This is set to **true** by default. If you are sure that the component doesn't need to render after **state** or **props** are updated, you can return **false** value.
- **componentWillUpdate** is called just before rendering.
- **componentDidUpdate** is called just after rendering.
- **componentWillUnmount** is called after the component is unmounted from the dom. We are unmounting our component in **main.js**.

Keys:

React **keys** are useful when working with dynamically created components or when your lists are altered by the users. Setting the **key** value will keep your components uniquely identified after the change.

A “key” is a special string attribute you need to include when creating lists of elements in React. Keys are used in React to identify which items in the list are changed, updated or deleted. In other words we can say that keys are used to give an indentity to the elements in the lists. The next thing that comes in mind is that what should be good to be choose as key for the items in lists. It is recommended to use a string as a key that uniquely identifies the items in the list. Below example shows a list with string keys:

```
const numbers = [ 1, 2, 3, 4, 5 ];
const updatedNums = numbers.map((number)=>{
return <li>{ number } </li>;
});
```

Styling react using CSS:

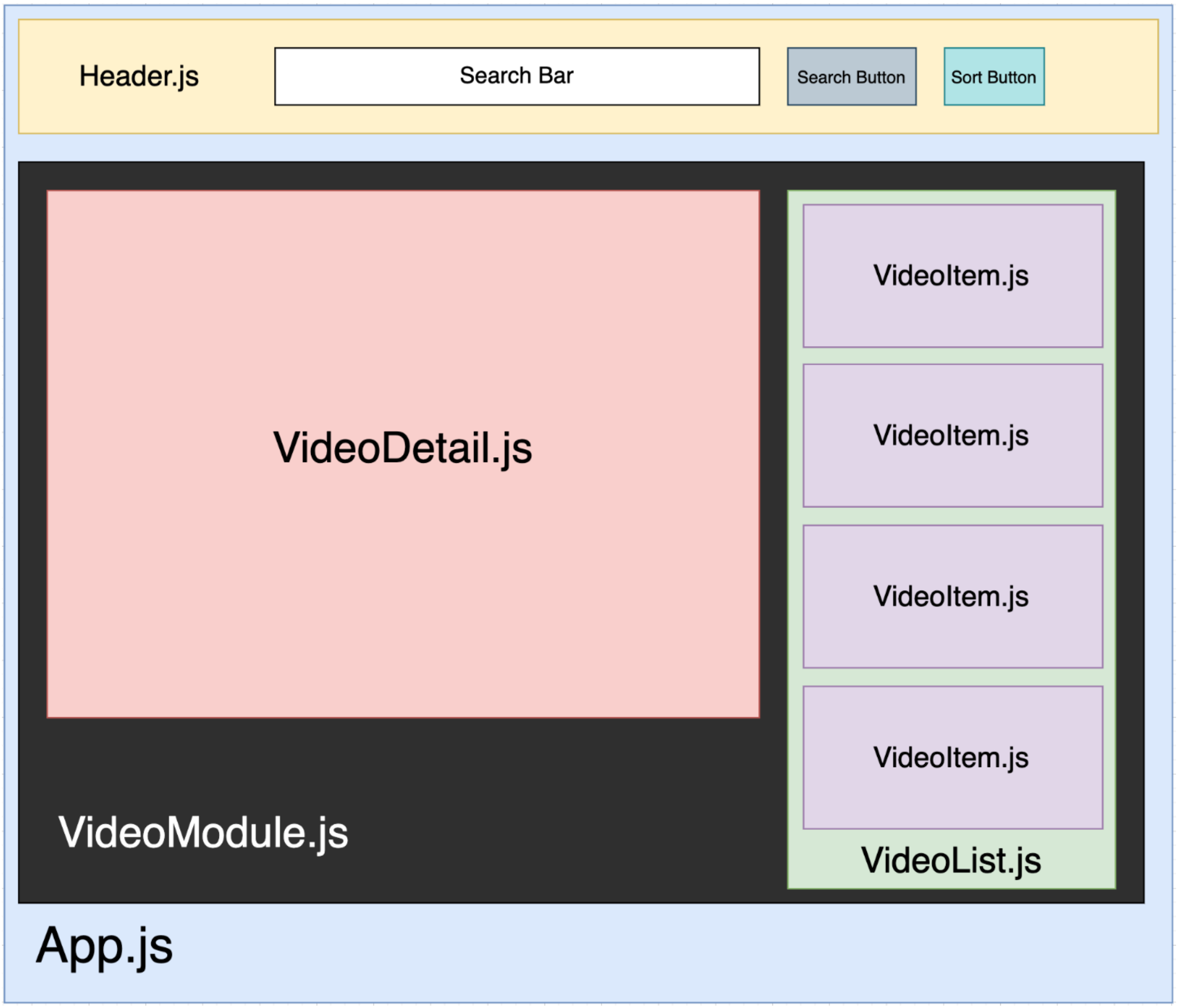
To style an element with the inline style attribute, the value must be a JavaScript object:

```
class MyHeader extends React.Component {

  render() {
    const mystyle = {
      color: "white",
      backgroundColor: "DodgerBlue",
      padding: "10px",
      fontFamily: "Arial"
    };
    return (
      <div>
      <h1 style={mystyle}>Hello Style!</h1>
      <p>Add a little style!</p>
      </div>
    );
  }
}
```

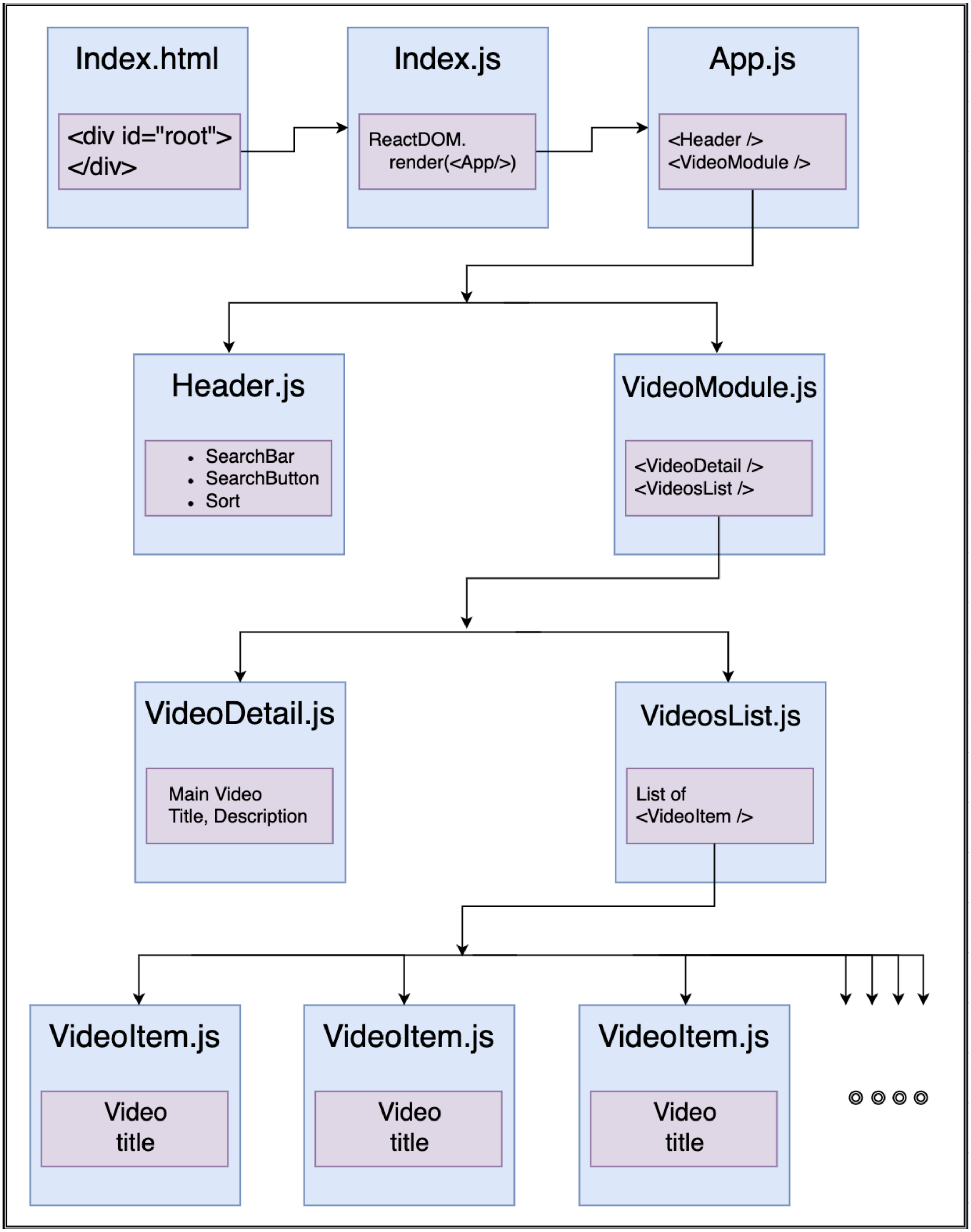
Chapter 6

System Design: Hierarchy of react components



This mini project was divided into a set of react components. The App.js component renders two components namely Header and VideoModule. The header component has a fixed header styling with a input search type textbox and having two buttons i.e. search and sort on the right of it. Search button allows you to search for videos based on your query typed in the search field. And a sort button helps you to sort the results based on a few factors like: relevance, date, rating, title, video count and view count.

The VideoModule component renders two more components namely VideoDetail and VideoList. The VideoDetail module is responsible for displaying the selected video in the left half of the screen while the VideoList module renders another component VideoItem. The VideoItem is responsible for playing the selected video from the suggested videos based on the keyword inserted by the user in the search bar.

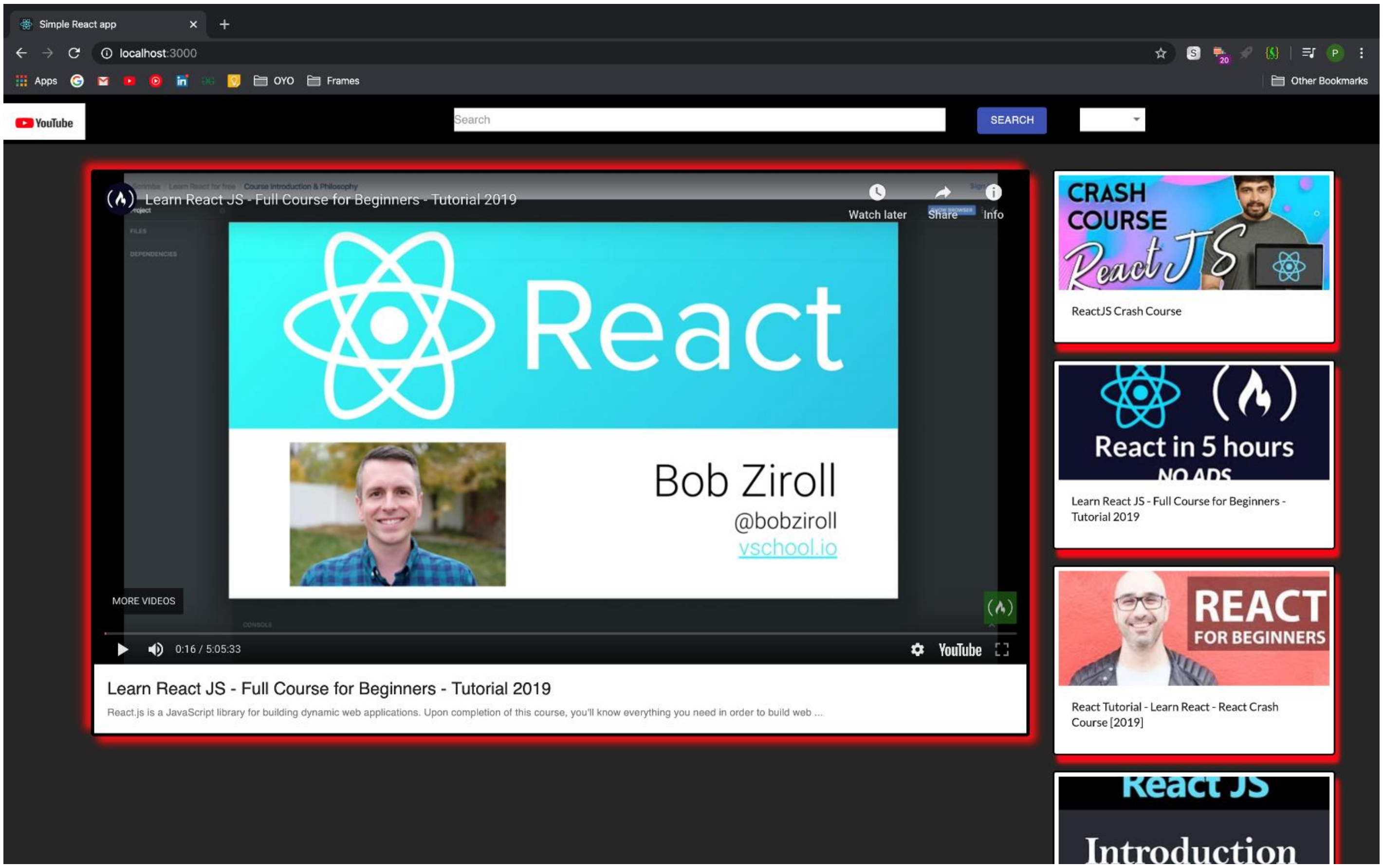

Index.html
<div id="root"></div>
Index.js
ReactDOM. render(<App/>)
App.js
<Header />
<VideoModule />
Header.js
SearchBar
SearchButton
Sort
VideoModule.js
<VideoDetail />
<VideosList />
VideoDetail.js
Main Video Title, Description
VideosList.js
List of <VideoItem />
VideoItem.js
Video title
VideoItem.js
Video title
VideoItem.js
Video title

Chapter 7

Project analysis and Implementation

This mini project was divided into a set of react components.The App.js component renders two components namely Header and VideoModule. The header component has a fixed header styling with a input search type textbox and having two buttons i.e search and sort on the right of it. Search button allows you to search for videos based on your query typed in the search field. And a sort button helps you to sort the results based on a few factors like: relevance, date, rating, title,videoCount and viewCount.

The VideoModule component renders two more components namely VideoDetail and VideoList. The VideoDetail module is responsible for displaying the selected video in the left half of the screen While the VideoList module renders another component VideoItem. The VideoItem is responsible for playing the selected video from the suggested videos based on the keyword inserted by the user in the search bar.



Learn React JS - Full Course for Beginners - Tutorial 2019

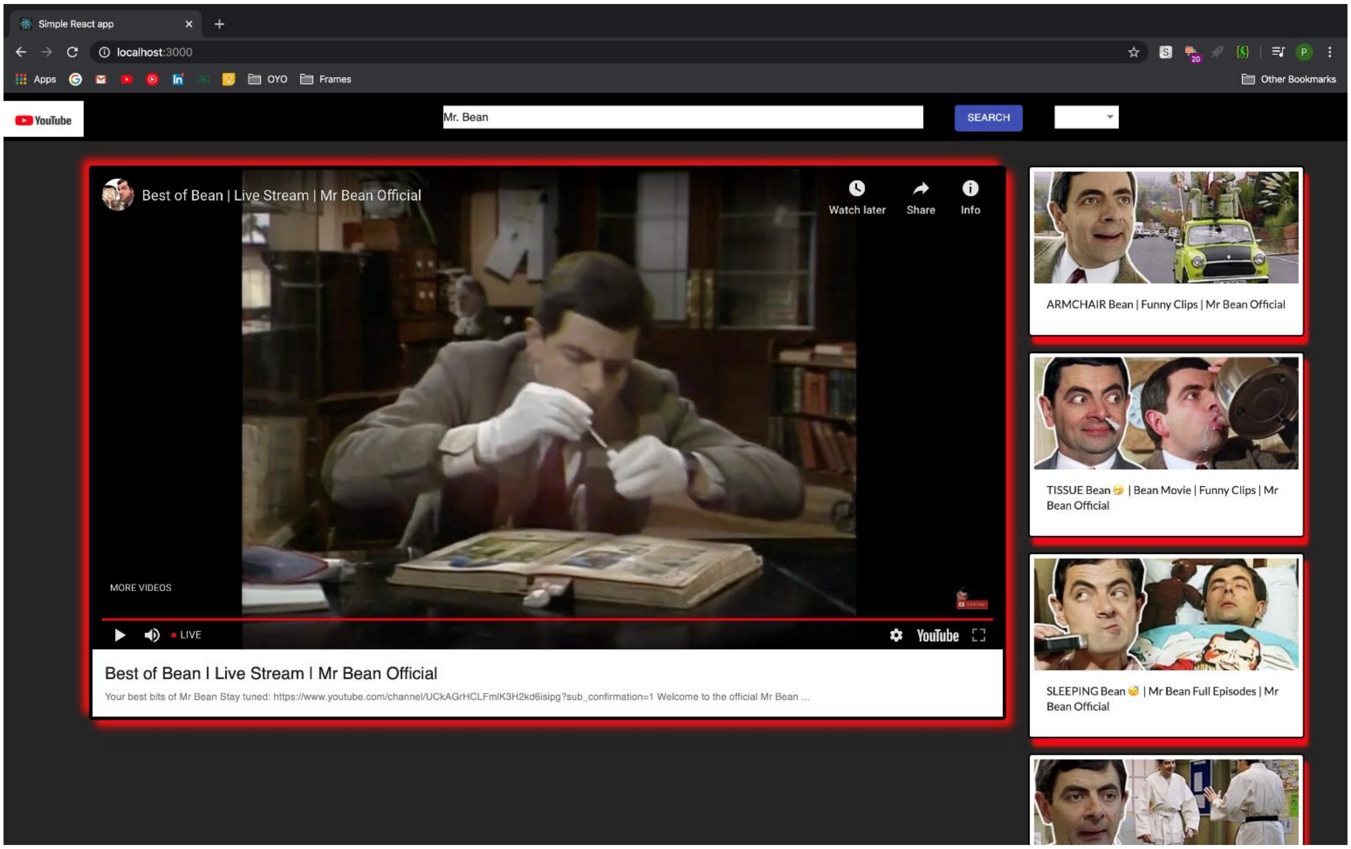
React.js is a JavaScript library for building dynamic web applications. Upon completion of this course, you'll know everything you need in order to build web ...

ReactJS Crash Course

Learn React JS - Full Course for Beginners - Tutorial 2019

React Tutorial - Learn React - React Crash Course [2019]

YOUTUBE CLONE



Best of Bean | Live Stream | Mr Bean Official

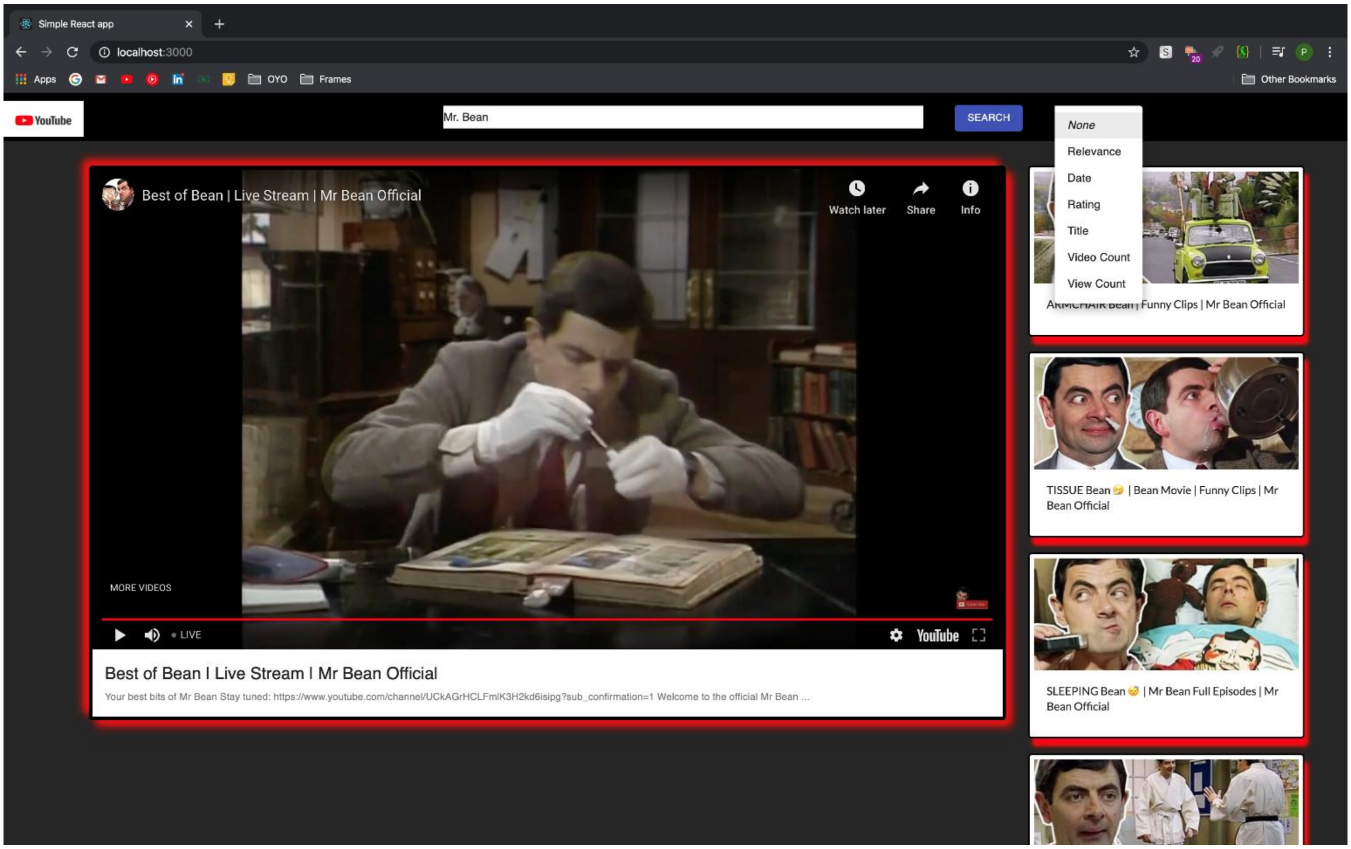
Best of Bean I Live Stream I Mr Bean Official

Your best bits of Mr Bean Stay tuned: https://www.youtube.com/channel/UCkAGrHCLFmlK3H2kd6isipg?sub_confirmation=1 Welcome to the official Mr Bean ...

ARMCHAIR Bean | Funny Clips | Mr Bean Official

TISSUE Bean 😉 | Bean Movie | Funny Clips | Mr Bean Official

SLEEPING Bean 😉 | Mr Bean Full Episodes | Mr Bean Official



Best of Bean | Live Stream | Mr Bean Official

Best of Bean I Live Stream I Mr Bean Official

Your best bits of Mr Bean Stay tuned: https://www.youtube.com/channel/UCkAGrHCLFmlK3H2kd6isipg?sub_confirmation=1 Welcome to the official Mr Bean ...

ARMCHAIR Bean | Funny Clips | Mr Bean Official

TISSUE Bean 😉 | Bean Movie | Funny Clips | Mr Bean Official

SLEEPING Bean 😉 | Mr Bean Full Episodes | Mr Bean Official

Chapter 8

Results

Learning outcomes:

Started with a 126 lessons course on scrimba on React JS. And then having an inspiration to try to build a product with massive user demand i.e. YouTube. Having read blogs about the installation process and environment setup to learning about npm commands to create a react app from scratch. Then, moving on we read about YouTube API v3 on Google API page. And read about axios library to fetch and use a real YouTube database.

Chapter 9
Conclusion and Future work

Conclusion:

Building this project helped me solidify my knowledge in the main concepts of React. This project will include React components, lifecycle methods, state management, passing props from parent to child components, API request and much more. Thus, it was a great overall learning experience.

The future work for this react app will be including major features of Youtube such as:

1. Home feed with infinite scroll.
2. Trending videos.
3. YouTube landing page.
4. Displaying comments and video details.
5. Having create your own video feature

Chapter 10

References

- https://developers.google.com/youtube/v3
- https://scrimba.com/g/glearnreact
- https://www.techomoro.com/how-to-install-and-setup-a-react-app-on-windows-10/
- https://scrimba.com/g/gintrotojavascript
- https://reactjs.org/docs/faq-internals.html
- https://developers.google.com/apis-explorer
- https://scrimba.com/g/glearnreact
- https://www.techomoro.com/how-to-install-and-setup-a-react-app-on-windows-10/
- https://scrimba.com/g/gintrotojavascript
- https://reactjs.org/docs/faq-internals.html
- https://www.tutorialspoint.com/reactjs/reactjs_overview.htm
- https://reactjs.org/docs/getting-started.html
- https://reactjs.net/tutorials/aspnetcore

CONTRIBUTION REPORT:

YOUTUBE CLONE

Shubhangi Suman
1605155

Abstract: Started with a 126 lessons course on scrimba on React JS. And then having an inspiration to try to build a product with massive user demand i.e. YouTube. Having read blogs about the installation process and environment setup to learning about npm commands to create a react app from scratch. Then, moving on we read about YouTube API v3 on Google API page. And read about axios library to fetch and use a real YouTube database.

CONTRIBUTION:

- Frontend development using React JS.
- Designing and styling the UI using CSS, Html.
- Learning and using Youtube API to fetch videos data.
- Project Documentation and reporting.
- Header Component
- Google's Youtube API

Full Signature of Supervisor: ……………………………. Full signature of the student: ……………………………..

CONTRIBUTION REPORT:

YOUTUBE CLONE

Pratil Dubey
1605139

Abstract: Started with a 126 lessons course on scrimba on React JS. And then having an inspiration to try to build a product with massive user demand i.e. YouTube. Having read blogs about the installation process and environment setup to learning about npm commands to create a react app from scratch. Then, moving on we read about YouTube API v3 on Google API page. And read about axios library to fetch and use a real YouTube database.

CONTRIBUTION:

- Frontend development using React JS.
- Designing and styling the UI using CSS, Html.
- Learning and using Youtube API to fetch videos data.
- Body Component Structure and styling.
- Project Documentation and reporting.

Full Signature of Supervisor: ……………………………

Full signature of the student: ……………………………..