Project Report

HONEYCOMB: A VIDEOSTREAMING APPLICATION

A Synopsis Submitted in Partial Fulfillment of the

Requirements for the degree of

BACHELOR IN TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGENEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Aditya Pratap Singh - 21BCE1719

Aryaman Chauhan - 21BCE1486

Kshitij Sharma - 21BCE5933

SCOPE, VIT Chennai

BCSE404L: Internet and Web Programming

Acknowledgement

We would like to extend our heartfelt gratitude to the esteemed faculty of VIT Chennai, especially Dr. Lekshmi RajeshKannan, for her invaluable guidance and unwavering support throughout the BCSE404L: Internet and Web Programming course.

Dr. Lekshmi's expertise and encouragement have been instrumental in shaping this project report. Her insightful feedback has truly enriched our learning experience, and we are immensely grateful for her dedication to our academic growth.

Additionally, we wish to express our appreciation to all those who have contributed directly or indirectly to the successful completion of this project. Your support and assistance have been invaluable, and we thank you from the bottom of our hearts.

Warm regards,

Aditya Pratap Singh - 21BCE1719

Aryaman Chauhan - 21BCE1486

Kshitij Sharma - 21BCE5933

Abstract

The digital age has witnessed an unprecedented surge in the popularity of video streaming platforms, revolutionizing the way we consume entertainment and information. In response to this trend, HoneyComb emerges as a meticulously crafted web application, meticulously designed to replicate the essence of leading video streaming websites. Through the adept utilization of cutting-edge web technologies such as HTML, React, JavaScript, and Tailwind CSS, HoneyComb delivers a seamless and immersive user experience. Its feature-rich platform encompasses a robust array of functionalities, including intuitive video search, seamless playback capabilities, and effortless video sharing. Moreover, the integration of optional user accounts elevates the user experience by offering personalized features such as playlist creation, video liking, and watch history tracking. HoneyComb aims to redefine the paradigm of online video consumption by prioritizing user engagement, accessibility, and innovation.

Keywords: Video Streaming, Web Application, User Experience, Personalisation

Introduction

The landscape of online media consumption has undergone a profound transformation with the widespread adoption of video streaming services. HoneyComb emerges as a dynamic and innovative web application, poised to capture the essence of leading video streaming platforms while introducing unique and compelling features. Built upon a foundation of modern web technologies such as HTML, React, JavaScript, and Tailwind CSS, HoneyComb exemplifies a commitment to excellence in user experience and functionality. By seamlessly blending intuitive design with cutting-edge features, HoneyComb seeks to carve a niche in the competitive realm of online video streaming. With an unwavering focus on user satisfaction and accessibility, HoneyComb aspires to become the platform of choice for discerning users seeking quality video content.

Methodology

Code files - https://github.com/LMN8R/HoneyComb

HTML (Hypertext Markup Language)

HTML serves as the backbone of web pages, providing the structure and layout for content displayed on the browser. In HoneyComb, HTML is used to create the fundamental structure of the web application, defining elements such as headers, footers, and navigation menus.

JavaScript

JavaScript is a versatile programming language commonly used for web development. It enables developers to add interactivity and functionality to web pages, enhancing the overall user experience. In HoneyComb, JavaScript is leveraged to implement features such as video playback, search functionality, and user interactions, making the application dynamic and responsive.

Tailwind CSS

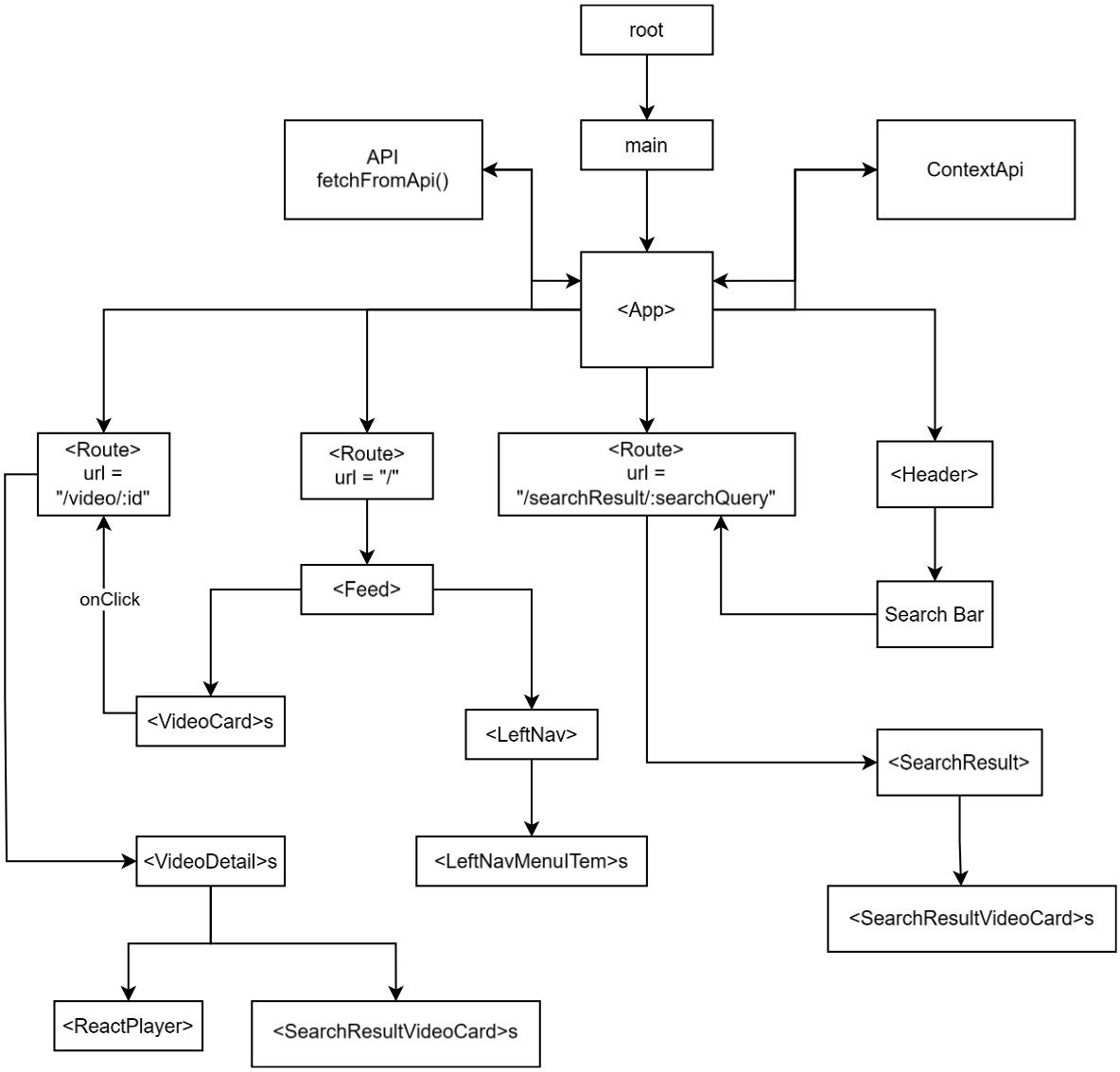
Tailwind CSS is a utility-first CSS framework that streamlines the process of styling web applications. Unlike traditional CSS frameworks, Tailwind CSS provides a set of utility classes that can be easily applied to HTML elements, allowing for rapid prototyping and customization. In HoneyComb, Tailwind CSS is utilized to design and style the user interface, ensuring consistency and aesthetic appeal across the application.

React

React is a JavaScript library for building user interfaces, developed by Facebook. It facilitates the

creation of dynamic and interactive components, enabling developers to efficiently manage complex user interfaces. In HoneyComb, React is utilized to create responsive and engaging user interfaces, ensuring seamless navigation and interaction throughout the application.

React DOM allows you to render React components into the DOM (Document Object Model) of a web page. When you create a React application, you use ReactDOM to display your components on the screen.



Index.html

```
<!DOCTYPEhtml>
<htmllang="en">
  <head>
    <metacharset="UTF-8" />
    <linkrel="icon"type="image/svg+xml"href="HONEYCOMB_arya.svg" />
    <metaname="viewport"content="width=device-width, initial-scale=1.0" />
    <title>HoneyComb</title>
  </head>
  <body>
    <divid="root"></div>
    <scripttype="module"src="/src/main.jsx"></script>
  </body>
</html>
```

Root node for ReactDOM

Main.jsx

```
import React from"react";
importReactDOMfrom"react-dom/client";
import App from"./App.jsx";
import"./index.css";
import{ AppContext } from"./context/ContextApi.jsx";
import{ BrowserRouter } from"react-router-dom";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <AppContext>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </AppContext>
  </React.StrictMode>
);
```

Root is created as the root for ReactDOM.

A. <AppContext>:

- This is a custom context provider component.
- It wraps the entire application and provides context data to its child components.

B. <BrowserRouter>:

- This is a React Router component.
- It listens to changes in the URL and renders the appropriate component based on the route.

C. <App>:

It is a custom component which is the includes the whole application.

App.jsx

```
import Header from"./components/Header";
import Feed from"./components/Feed";
importSearchResultfrom"./components/SearchResult";
importVideoDetailfrom"./components/VideoDetail";

import{ Routes, Route } from"react-router-dom";
import{ useEffect } from"react";

functionApp() {
  useEffect(() => {
    window.scrollTo(0, 0);
  }, []);

  return (
    <divclassName="flex flex-col h-full">
      <Header />
      <Routes>
        <Routepath="/"element={<Feed />} />
        <Routepath="/searchResult/:searchQuery"element={<SearchResult />} />
```

```
        <Routepath="/video/:id"element={<VideoDetail />} />
      </Routes>
    </div>
  );
}

exportdefaultApp;
```

A. <Header>:

a. Custom component that loads the nav bar and includes the search bar, logo and user profile picture.

B. <Routes>:

a. It is a component provided by React, it defines the routes for the application. The child <Route> components define what to render for specific URL paths.

b. Following are the Routes:

i. <Route path="/" element={<Feed />} />

1. A route for the root URL ("/").
2. When the URL matches "/", it renders the Feed component.

ii. <Route path="/searchResult/:searchQuery" element={<SearchResult />} />

1. A route for URLs like "/searchResult/:searchQuery ", where ":searchQuery" is a dynamic parameter.
2. It renders the SearchResult component and passes the search query as a parameter.

iii. <Route path="/video/:id" element={<VideoDetail />} />

1. A route for URLs like "/video/:id", where ":id" is a dynamic video ID.

2. It renders the VideoDetail component and passes the video ID as a parameter.

Header.jsx

```
import{ useContext, useState } from"react";
import{ Link, useLocation, useNavigate } from"react-router-dom";

import logo from"/HONEYCOMB_arya.svg";
import{ SlMenu } from"react-icons/sl";
import{ IoIosSearch } from"react-icons/io";
import{ RiVideoAddLine } from"react-icons/ri";
import{ FiBell } from"react-icons/fi";
import{ CgClose } from"react-icons/cg";

import{ Context } from"../context/ContextApi";
import Loader from"../shared/loader";

constHeader= () => {
  const [searchQuery, setSearchQuery] =useState("");
```

A. searchQuery is initialized to “”

```
const{ loading, mobileMenu, setMobileMenu } =useContext(Context);

  constnavigate=useNavigate();
```

B. useNavigate() object is created, which when called routes to that specified route.

```
constsearchQueryHandler= (event) => {
    if (
      (event?.key=="Enter"||event=="searchButton") &&
      searchQuery?.length>0
    ) {
      navigate(`/searchResult/${searchQuery}`);
    }
  };
```

C. This function handles the search requests, and if “Enter” is clicked on the keyboard or “searchButton” is clicked and the search query has length greater length than 0, it will call the navigate function to “/searchResult/${searchQuery}”, which is matched by <Routes> and <searchResult> element is called with searchQuery as the input.

```
const{ pathname } =useLocation();
constpageName=pathname?.split("/")?.filter(Boolean)?.[0];
```

D. Determine the current route

```
return (
    <divclassName="sticky top-0 z-20 flex flex-row items-center justify-
between h-14 px-4 md:px-5 bg-white dark:bg-gray-900">
      {loading&&<Loader />}

      <divclassName="flex h-5 items-center">
        {pageName!=="video"&& (
          <div
            className="flex md:hidden md:mr-6 mr-4 cursor-pointer items-center
justify-center h-10 w-10 rounded-full hover:bg-[#8888]/[0.6]"
            onClick={mobileMenuToggle}
          >
            {mobileMenu? (
              <CgCloseclassName="dark:text-white text-black text-xl" />
            ) : (
              <SlMenuclassName="dark:text-white text-black text-xl" />
            )}
          </div>
        )}
        <Linkto="/"className="flex h-5 items-center mt-4">
          <img
            className="hidden dark:hiddenmd:block h-20"
            src={logo}
            alt="Honeycomb"
          />
          <divclassName="hidden dark:hiddenmd:block text-yellow-500 dark:text-
yellow-300 text-lg cursor-pointer flex items-center px-3 mb-[4px] rounded-lg
hover:dark:bg-white/[0.15] hover:dark:bg-white[0.15]">HoneyComb</div>
        </Link>
```

```
      </div>
```

E. Left Side

a. If loading is true, it renders a Loader component (presumably a loading spinner).

b. If the pageName is not equal to “video,” render the following content. This conditionally displays the menu toggle button.

c. Upon clicking the “Honeycomb” logo, it navigates to home page “/”.

d. This div provided the Navigation menu toggle, “Honeycomb” logo and name.

```
<divclassName="group flex items-center my-1 mr-7">
        <divclassName="flex h-8 md:h-10 md:ml-10 md:pl-5 border-2 border-
yellow-300 group-focus-within:border-2 md:group-focus-within:ml-5 md:group-
focus-within:pl-0">
          <divclassName="w-10 items-center justify-center hidden group-focus-
within:md:flex">
            <IoIosSearchclassName="text-black dark:text-white text-xl" />
          </div>
          <input
            type="text"
            className="bg-transparent outline-none text-black dark:text-white
pr-5 pl-5 md:pl-0 w-44 md:group-focus-within:pl-0 md:w-64 lg:w-[500px]"
            onChange={(e) =>setSearchQuery(e.target.value)}
            onKeyUp={searchQueryHandler}
            placeholder="Search"
            value={searchQuery}
          />
        </div>
        <button
          className="w-[40px] md:w-[60px] h-8 md:h-10 flex items-center
justify-center border-2 border-l-0 border-yellow-300 bg-white/[0.1] dark:bg-
black/[0.1]"
          onClick={() =>searchQueryHandler("searchButton")}
        >
          <IoIosSearchclassName="text-black dark:text-white text-xl" />
        </button>
      </div>
```

F. Search bar

a. <input>: The search query is written here, the “onChange” event handler updates the “searchQuery” state variable. The onKeyUp triggers the searchQueryHandler.

b. The search button, when clicked also triggers the searchQueryHandler.

```
<divclassName="flex items-center">
        <divclassName="hidden md:flex">
          <divclassName="flex items-center justify-center h-10 w-10 rounded-
full hover:bg-[#303030]/[0.6]">
            <RiVideoAddLineclassName="text-black dark:text-white text-xl
cursor-pointer" />
          </div>
          <divclassName="flex items-center justify-center ml-2 h-10 w-10
rounded-full hover:bg-[#303030]/[0.6]">
            <FiBellclassName="text-black dark:text-white text-xl cursor-
pointer" />
          </div>
        </div>
        <divclassName="flex h-8 w-8 overflow-hidden md:ml-4 mx-1">
          <img
            src="pfp.jpg"
            className="w-full h-full object-cover"
          />
        </div>
      </div>
    </div>
  );
};

exportdefaultHeader;
```

Header Output

G. Right side: Adds a few icons and user profile picture.

LeftNav.jsx

```
import React, { useContext } from"react";
import{ useNavigate } from"react-router-dom";

importLeftNavMenuItemfrom"./LeftNavMenuItem";
import{ Context } from"../context/ContextApi";
import{ categories } from"../utils/Constants";

functionLeftNav() {
  const{ selectcategories, setSelectCategories, mobileMenu } =
    useContext(Context);
  constnavigate=useNavigate();

  constclickHandle= (name, type) => {
    switch (type) {
      case"category":
        returnsetSelectCategories(name);
      case"home":
        returnsetSelectCategories(name);
      case"menu":
        returnfalse;

      default:
        break;
    }
  };
```

A. clickHandle

a. This function take two inputs, name and type. Based on the type, the state variable setSelectCategories is updated.

```
return (
    <div
      className={`md:block w-[240px] overflow-y-auto h-full py-4 bg-white
dark:bg-gray-900 absolute md:relative z-10 hide translate-x-[-240px]
md:translate-x-0 transition-all ${
        mobileMenu? "translate-x-[0px]" : ""
      }`}
    >
      <divclassName="flex px-5 flex-col">
```

```
        {categories?.map((item) => {
          return (
            <React.Fragmentkey={item.name}>
              <LeftNavMenuItem
                key={item.type}
                text={item.type===="home"?"Home":item.name}
                icon={item.icon}
                action={() => {
                  clickHandle(item.name, item.type);
                  navigate("/");
                }}
                className={`${
                  selectcategories===item.name
                    ? "bg-black/[0.15] dark:bg-white/[0.15]"
                    : ""
                }`}
              />
              {item.divider&& (
                <hrclassName="my-5 border-white/[0.2] dark:border-black/[0.2]"
/>
              )}
            </React.Fragment>
          );
        })}
        <hrclassName="my-5 border-white[0.2]" />
        <divclassName="text-black/[0.5] dark:text-white/[0.5] text-[12px]
font-semibold">
          IWP Project
        </div>
      </div>
    </div>
  );
}

exportdefaultLeftNav;
```

B. Reactive window:

a. md:block w-[240px]: The width is set to 240 pixels on medium and larger screens.

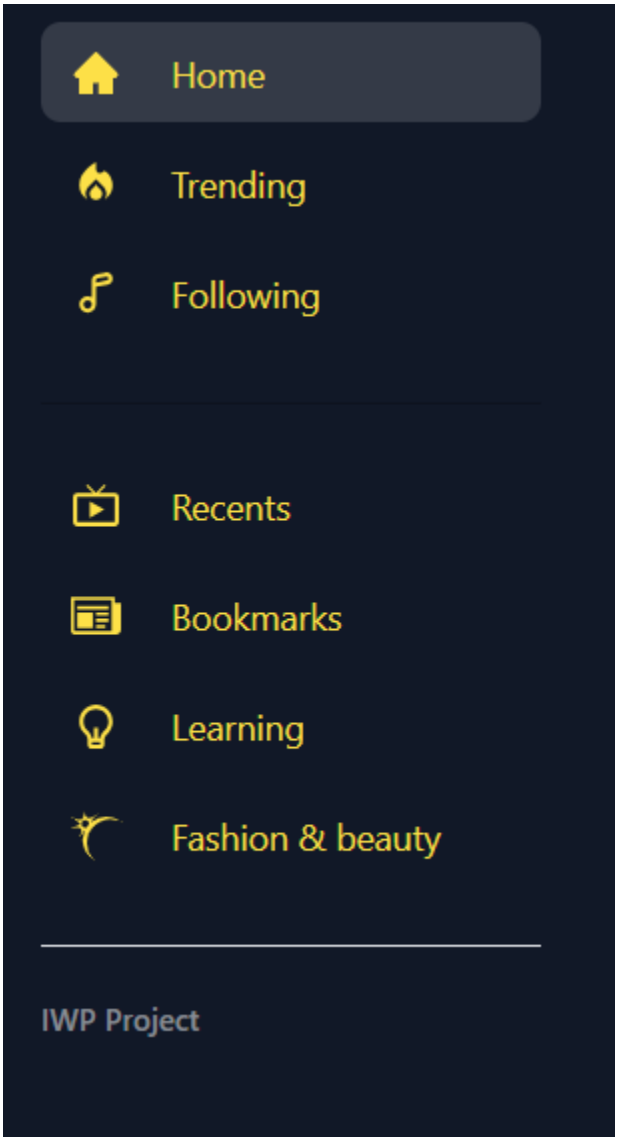
b. overflow-y-auto: vertical scrolling if contents overflow.

c. md:relative: The position is absolute, to cover the entire height but relative on medium and larger screens.

C. Category map:

a. {categories?.map((item) => { ... })}: maps the menu items based on categories data. Each item is different category.

D. <LeftNavMenuItem/>: has attributes key, text, icon and action. The item's key and type are passed as arguments to clickHandle whose result becomes the action, and the app is navigated to home page "/".



LeftNavMenuItem.jsx

```
functionLeftNavMenuItem({ text, icon, className, action }) {
  return (
    <div
      className={
        "text-yellow-500 dark:text-yellow-300 text-sm cursor-pointer h-10 flex
items-center px-3 mb-[4px] rounded-lg hover:dark:bg-white/[0.15]
hover:dark:bg-white[0.15] "+
        className
      }
      onClick={action}
    >
      <spanclassName="text-xl mr-5">{icon}</span>
      {text}
    </div>
  );
}

exportdefaultLeftNavMenuItem;
```

A. Onclick: the action set as the argument to this component is executed when this component is clicked.

B. This component displays the icon and text of the Left Nav Menu.

Feed.jsx

```
import{ useContext, useEffect } from"react";

import{ Context } from"../context/ContextApi";
importLeftNavfrom"./LeftNav";
importVideoCardfrom"./VideoCard";

constFeed= () => {
  const{ loading, searchResult } =useContext(Context);
```

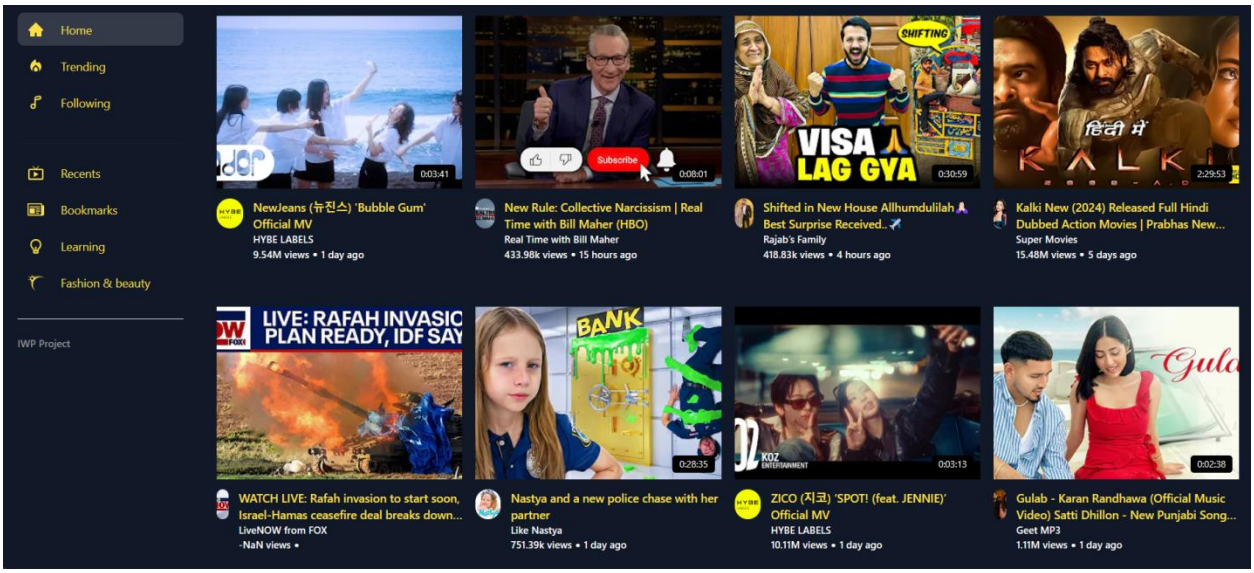
```
  useEffect(() => {
    document.getElementById("root").classList.remove("custom-h");
    window.scrollTo(0, 0);
  }, []);

  return (
    <divclassName="flex flex-row h-[calc(100%-56px)]">
      <LeftNav />
      <divclassName="grow w-[calc(100%-240px)] h-full overflow-y-auto bg-gray-
200 dark:bg-gray-900">

        <divclassName="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-4 p-5">
          {!loading&&
            searchResult.map((item, index) => {
              if (item.type!=="video") returnfalse;
              return<VideoCardkey={index}video={item?.video} />;
            })}
        </div>
      </div>
    </div>
  );
};

exportdefaultFeed;
```

Output:



A. The <Feed> component is rendered when the URL matches "/". It includes the <LeftNav> component.

B. If the data has been fetched or !loading is true then, each result of searchResult array is mapped to a <VideoCard> component. If the item.type is not "video", rendering is skipped.

C. It is a dynamically rendered page.

VIdeoCard.jsx

```
import{ Link } from"react-router-dom";
import{ abbreviateNumber } from"js-abbreviation-number";

importVideoLengthfrom"../shared/VideoLength";
functionVideoCard({ video }) {
  return (
    <Linkto={`/video/${video?.videoId}`}>
      <divclassName="flex flex-col mb-8">
        <divclassName="relative h-48 md:h-52 overflow-hidden">
          <img
            src={video?.thumbnails[0]?.url}
            alt="thumbnails"
            className="h-full w-full object-cover"
          />
```

```
          {video.lengthSeconds&&<VideoLengthtime={video?.lengthSeconds} />}
        </div>
        <divclassName="flex text-white mt-3">
          <divclassName="flex items-start">
            <divclassName="flex h-9 w-9 rounded-full overflow-hidden ">
              <img
                src={video?.author?.avatar[0]?.url}
                alt="avatar"
                className="w-full h-full object-cover"
              />
            </div>
            <divclassName="flex flex-col ml-3 overflow-hidden">
              <spanclassName="text-sm font-semibold line-clamp-2 text-black
dark:text-yellow-300">
                {video?.title}
              </span>
              <spanclassName="text-[12px] font-semibold text-black/[0.7]
dark:text-white flex items-center ">
                {video?.author?.title}
              </span>
              <divclassName="flex text-[12px] font-semibold text-black/[0.7]
dark:text-white truncate overflow-hidden">
                <span>{`${abbreviateNumber(
                  video?.stats?.views,
                  2
                )} views`}</span>
                <spanclassName="flex  text-[24px] leading-none font-bold text-
black/[0.7] dark:text-white relative top-[-10px] mx-1">
                  .
                </span>
                <spanclassName="truncate">{video?.publishedTimeText}</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </Link>
  );
}
```

```
exportdefaultVideoCard;
```

A. This component renders the videocard (thumbnail, video-length, author, author-avatar, author name, views, and date of publish).

B. All of the above mentioned is encapsulated inside the <Link> component which navigates to url “/video/${video?.videoId”, which matches the <Route path="/video/:id" element={<VideoDetail />} />, hence <VideoDetail> component is rendered with videoId as an argument.

VideoDetail.jsx

```
import{ useState, useEffect, useContext } from”react”;
import{ useParams } from”react-router-dom”;
importReactPlayerfrom”react-player/youtube”;
import{ AiOutlineLike } from”react-icons/ai”;
import{ abbreviateNumber } from”js-abbreviation-number”;
import{ fetchDataFromApi } from”../utils/Api”;
```

```
import{ Context } from"../context/ContextApi";
importSuggestionVideoCardfrom"./SuggestionVideoCard";

functionVideoDetail() {
  const [video, setVideo] =useState();
  const [relatedVideos, setRelatedVideos] =useState();
  const{ id } =useParams();
  const{ setLoading } =useContext(Context);

  useEffect(() => {
    document.getElementById("root").classList.add("custom-h");
    fetchVideoDetails();
    fetchRelatedVideos();
    window.scrollTo(0, 0);
  }, [id]);

  constfetchVideoDetails= () => {
    setLoading(true);
    fetchDataFromApi(`video/details/?id=${id}`).then((res) => {
      setVideo(res);
      setLoading(false);
    });
  };
  constfetchRelatedVideos= () => {
    setLoading(true);
    fetchDataFromApi(`video/related-contents/?id=${id}`).then((res) => {
      setRelatedVideos(res);
      setLoading(false);
    });
  };
```

A. This component is the video player page, where the <LeftNav> component is still maintained.

B. The useEffect() hook in react executes the first function argument every time the second argument, [id] is changed. So for every<VideoDetail> component rendering, fetchVideoDetails() and fetchRelatedVideos() will be called for the argument “id”.

C. fetchVideoDetails():

   a. This function set the state variable “loading” to true.

b. It calls the fetchDataFromApi() function with the “id” as input, the result is the set to a state variable “Video”.

c. After the loading is done, the state variable “loading” is set to false.

D. fetchRelatedVideos():

a. This function retrieves the data about the videos related to the current video that is being played.

```
return (
    <divclassName="flex justify-center flex-row h-[calc(100%-56px)] bg-white
dark:bg-gray-900">
      <divclassName="w-full max-w-[1280px] flex flex-col lg:flex-row">
        <divclassName="flex flex-col lg:w-[calc(100%-350px)] xl:w-[calc(100%-
400px)] px-4 py-3 lg:py-6 overflow-y-auto">
          <divclassName="h-[200px] md:h-[400px] lg:h-[400px] xl:h-[550px] ml-
[-16px] lg:ml-0 mr-[-16px] lg:mr-0">
            <ReactPlayer
              url={`https://www.youtube.com/watch?v=${id}`}
              controls
              width="100%"
              height="100%"
              style={{ backgroundColor: "#00000" }}
              playing={true}
            />
          </div>
          <divclassName="text-black dark:text-yellow-300 font-semibold text-
smmd:text-xl mt-4 line-clamp-2">
            {video?.title}
          </div>
          <divclassName="flex justify-between flex-col md:flex-row mt-4">

            <divclassName="flex text-black dark:text-white mt-4 md:mt-0">
              <divclassName="flex items-center justify-center h-11 px-6
dark:bg-white/[0.15] bg-black/[0.2]">
```

```
                <AiOutlineLikeclassName="text-xl dark:text-white text-black
mr-2" />
                {`${abbreviateNumber(video?.stats?.views, 2)} Likes`}
              </div>
              <divclassName="flex items-center justify-center h-11 px-6
dark:bg-white/[0.15] bg-black/[0.2] ml-4">
                {`${abbreviateNumber(video?.stats?.views, 2)} Views`}
              </div>
            </div>
            <divclassName="flex">
              <divclassName="flex items-start ">
                <divclassName="flex h-11 w-11 rounded-full overflow-hidden">
                  <img
                    src={video?.author?.avatar[0]?.url}
                    alt="avatar"
                    className="h-full w-full object-cover"
                  />
                </div>
              </div>
              <divclassName="flex flex-col ml-3 ">
                <divclassName="text-black dark:text-white text-md font-
semibold flex items-center">
                  {video?.author?.title}
                </div>
                <divclassName="text-black/[0.7] dark:text-white text-sm">
                  {video?.author?.stats?.subscribersText}
                </div>
              </div>
            </div>
          </div>
        </div>
        <divclassName="flex flex-col py-6 px-4 overflow-y-auto lg:w-[350px]
xl:w-[400px] hide">
          {relatedVideos?.contents?.map((item, index) => {
            if (item?.type!=="video") returnfalse;
            return<SuggestionVideoCardkey={index}video={item?.video} />;
          })}
        </div>
      </div>
```

```
    </div>
  );
}

exportdefaultVideoDetail;
```

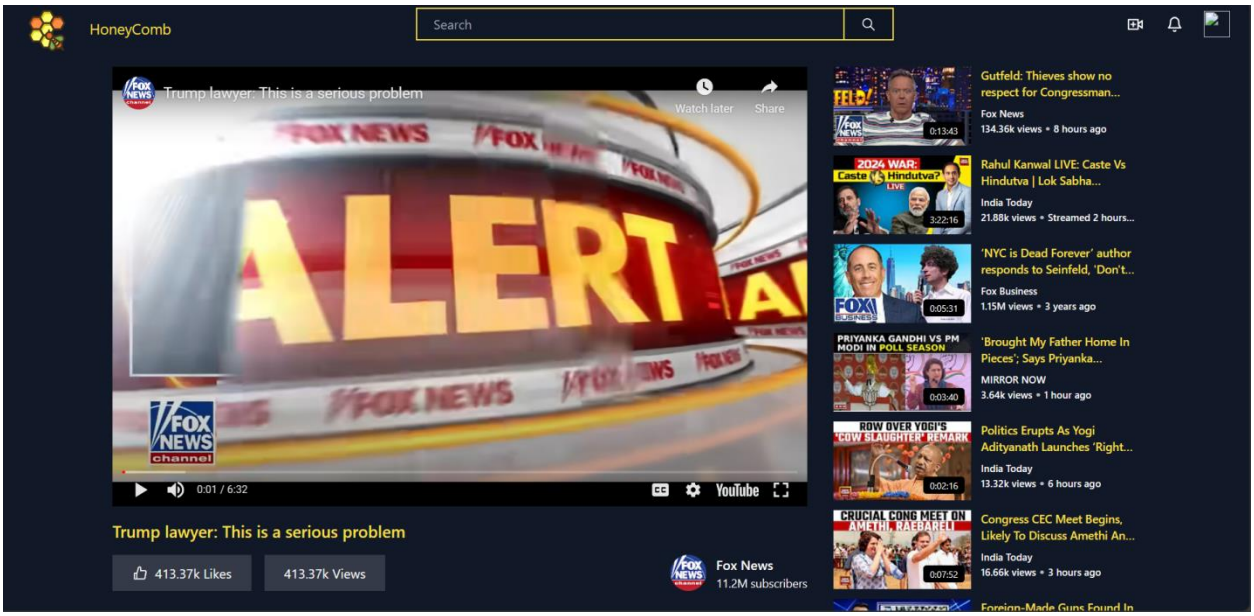
E. <ReactPlayer>:

   a. This renders a video player, the video url is dynamically generated based on the id.

   b. The state variable “playing” is set to true.

F. The video title, stats, author-avatar, author title are displayed.

G. The data fetched from fetchRelatedVideos() is mapped over <SuggestionVideoCard> component.



SuggestionVideoCard.jsx

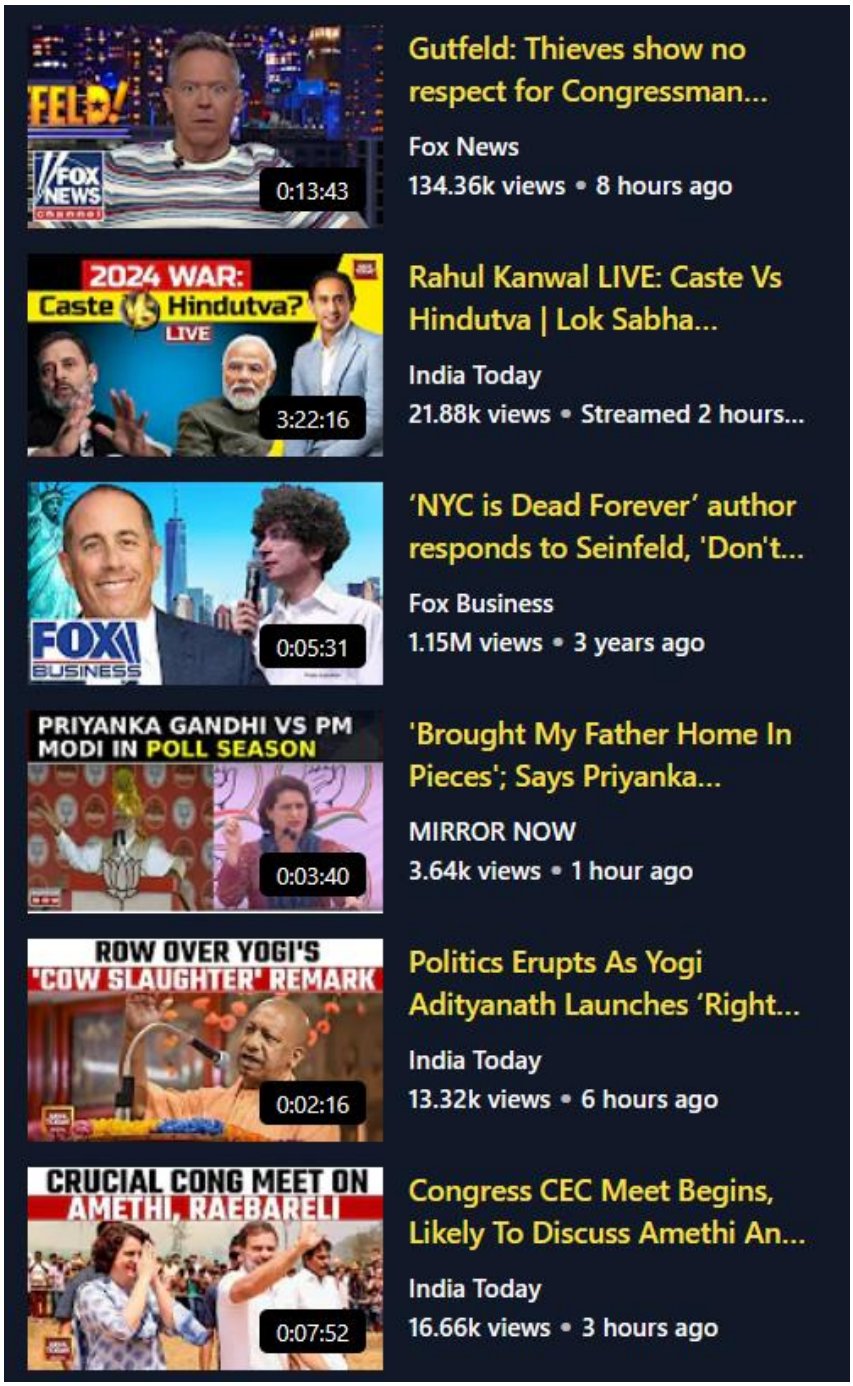
```
/* eslint-disable react/prop-types */
import{ abbreviateNumber } from"js-abbreviation-number";
import{ BsFillCheckCircleFill } from"react-icons/bs";
import{ Link } from"react-router-dom";
```

```
importVideoLengthfrom"../shared/VideoLength";

constSuggestionVideoCard=({ video }) => {
  return (
    <Linkto={`/video/${video?.videoId}`}>
      <divclassName="mb-3 flex flex-col md:flex-row items-start">
        <divclassName="relative h-56 md:h-24 lg:h-20 xl:h-24 w-full md:w-40
min-w-[168px] lg:w-32 lg:min-w-[128px] xl:w-40 xl:min-w-[168px] bg-slate-800
overflow-hidden">
          <img
            className="h-full w-full object-cover"
            src={video?.thumbnails[0]?.url}
          />
          {video?.lengthSeconds&&<VideoLengthtime={video?.lengthSeconds} />}
        </div>
        <divclassName="flex flex-col ml-3 overflow-hidden">
          <spanclassName="text-md lg:text-xsxl:text-sm font-semibold line-
clamp-2 text-black dark:text-yellow-300">
            {video?.title}
          </span>
          <spanclassName="text-[12px] lg:text-[10px] xl:text-[12px] font-
semibold mt-2 text-black/[0.7] dark:text-white flex items-center">
            {video?.author?.title}
          </span>
          <divclassName="flex text-[12px] lg:text-[10px] xl:text-[12px] font-
semibold text-black/[0.7] dark:text-white truncate overflow-hidden">
            <span>{`${abbreviateNumber(video?.stats?.views, 2)} views`}</span>
            <spanclassName="flex text-[24px] leading-none font-bold text-
black/[0.7] dark:text-white/[0.7] relative top-[-10px] mx-1">
              .
            </span>
            <spanclassName="truncate">{video?.publishedTimeText}</span>
          </div>
        </div>
      </div>
    </Link>
  );
};
```

```
exportdefaultSuggestionVideoCard;
```

A. This is similar to VideoCard but with different layout and



SearchResult.jsx

```
import{ useState, useEffect, useContext } from"react";
import{ useParams } from"react-router-dom";

import{ fetchDataFromApi } from"../utils/Api";
import{ Context } from"../context/ContextApi";
importLeftNavfrom"./LeftNav";
importSearchResultVideoCardfrom"./SearchResultVideoCard";
functionSearchResult() {
  const [result, setResult] =useState();
  const{ searchQuery } =useParams();
  const{ setLoading } =useContext(Context);

  useEffect(() => {
    document.getElementById("root").classList.remove("custom-h");
    fetchSearchResult();
    window.scrollTo(0, 0);
  }, [searchQuery]);

  constfetchSearchResult= () => {
    setLoading(true);
    fetchDataFromApi(`search/?q=${searchQuery}`).then((res) => {
      setResult(res.contents);
      setLoading(false);
    });
  };
```

A. When <SearchResult> component is rendered, fetchSearchResult() is called every time the dependency [searchQuery] is changed.

B. fetchSearchResult():

   a. “loading” is set to true

   b. Data is fetched using the API call, fetchDataFromApi(‘search/?q=${searchQuery}’) which returns the search result for the state variable “searchQuery”.

   c. The result content is set to a state variable Result and loading is set to false.

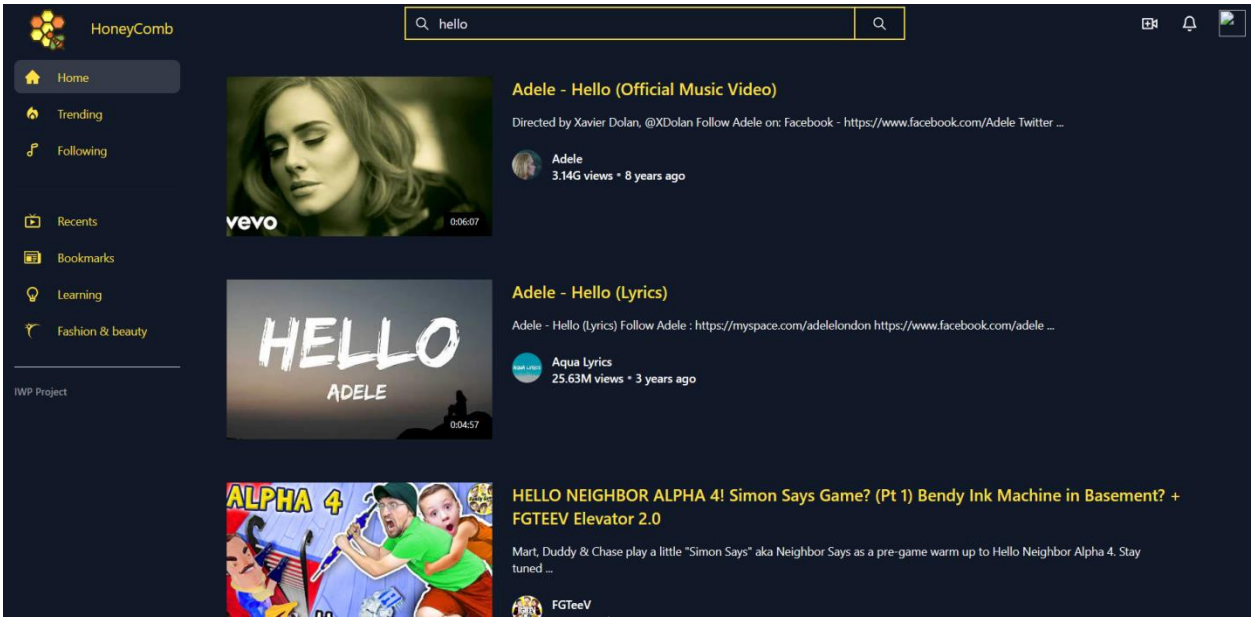
```
return (
```

```
    <divclassName="flex flex-row  h-[calc(100%-56px)]">
      <LeftNav />
      <divclassName="grow w-[calc(100%-240px)] h-full overflow-y-auto bg-white
dark:bg-gray-900">
        <divclassName="grid grid-cols-1 gap-2 p-5">
          {result?.map((item) => {
            if (item.type!=="video") returnfalse;
            return (
              <SearchResultVideoCard
                key={item?.video?.videoId}
                video={item?.video}
              />
            );
          })}
        </div>
      </div>
    </div>
  );
}

exportdefaultSearchResult;
```

C. Each item in the state variable “result” array is mapped to <SearchResultVIdeoCard> component.

D. In brief, this component results the video cards of the search query.



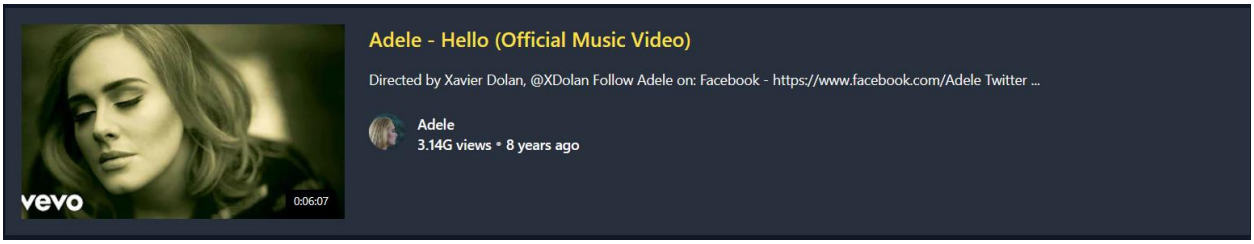
SearchResultVideoCard

```
import{ Link } from"react-router-dom";
import{ abbreviateNumber } from"js-abbreviation-number";
importVideoLengthfrom"../shared/VideoLength";
functionSearchResultVideoCard({ video }) {
  return (
    <Linkto={`/video/${video?.videoId}`}>
      <divclassName="flex flex-col md:flex-row mb-8 md:mb-3 dark:lg:hover:bg-
white/[0.1]  lg:hover:bg-black/[0.1] md:p-4">
        <divclassName="relative flex shrink-0 h-48 md:h-28 lg:h-40 xl:h-48 w-
full md:w-48 lg:w-64 xl:w-80 bg-slate-800 overflow-hidden">
          <img
            className="h-full w-full object-cover"
            src={video?.thumbnails[0]?.url}
          />
          {video?.lengthSeconds&&<VideoLengthtime={video?.lengthSeconds} />}
        </div>
        <divclassName="flex  flex-col ml-4 md:ml-6 mt-4 md:mt-0 overflow-
hidden">
          <spanclassName="text-lg md:text-xl font-semibold line-clamp-2 text-
black dark:text-yellow-300">
            {video?.title}
```

```
          </span>
          <spanclassName="empty:hidden text-sm line-clamp-1 md:line-clamp-2
text-black/[0.7] dark:text-white md:pr-24 md:my-4">
            {video?.descriptionSnippet}
          </span>
          <divclassName="flex items-center">
            <divclassName="flex items-start mr-3">
              <divclassName="flex h-9 w-9 rounded-full overflow-hidden">
                <img
                  className="h-full w-full object-cover"
                  src={video?.author?.avatar[0]?.url}
                />
              </div>
            </div>
            <divclassName="flex flex-col">
              <spanclassName="text-sm font-semibold mt-2 text-black/[0.7]
dark:text-white flex items-center">
                {video?.author?.title}
              </span>
              <divclassName="flex text-sm font-semibold text-black/[0.7]
dark:text-white truncate overflow-hidden">
                <span>{`${abbreviateNumber(
                  video?.stats?.views,
                  2
                )} views`}</span>
                <spanclassName="flex text-[24px] leading-none font-bold text-
black/[0.7] dark:text-white/[0.7] relative top-[-10px] mx-1">
                  .
                </span>
                <spanclassName="truncate">{video?.publishedTimeText}</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </Link>
  );
}
```

```
exportdefaultSearchResultVideoCard;
```

A. This is similar to <VideoCard> component but <SearchResultVideoCard> component contains video description.



Api.jsx

```
importaxiosfrom"axios";

constBASE_URL="https://youtube138.p.rapidapi.com";
constoptions= {
  params: {
    hl: "en",
    gl: "in",
  },
  headers: {
    "X-RapidAPI-Key": "6a42b79c08msha6243efef7889a4p1b0eefjsne2c76a30d966",
    "X-RapidAPI-Host": "youtube138.p.rapidapi.com",
  },
};

exportconstfetchDataFromApi=async (url) => {
  const{ data } =awaitaxios.get(`${BASE_URL}/${url}`, options);
  returndata;
};
```

A. Axios is a library for making HTTP requests, and it's commonly used in React applications for data fetching.

B. Axios provides methods for different HTTP request types (GET, POST, PUT, DELETE).

C. RapidAPI provides API for youtube data.

D. hl: Language parameter set to English.

E. gl: Region parameter set to India.

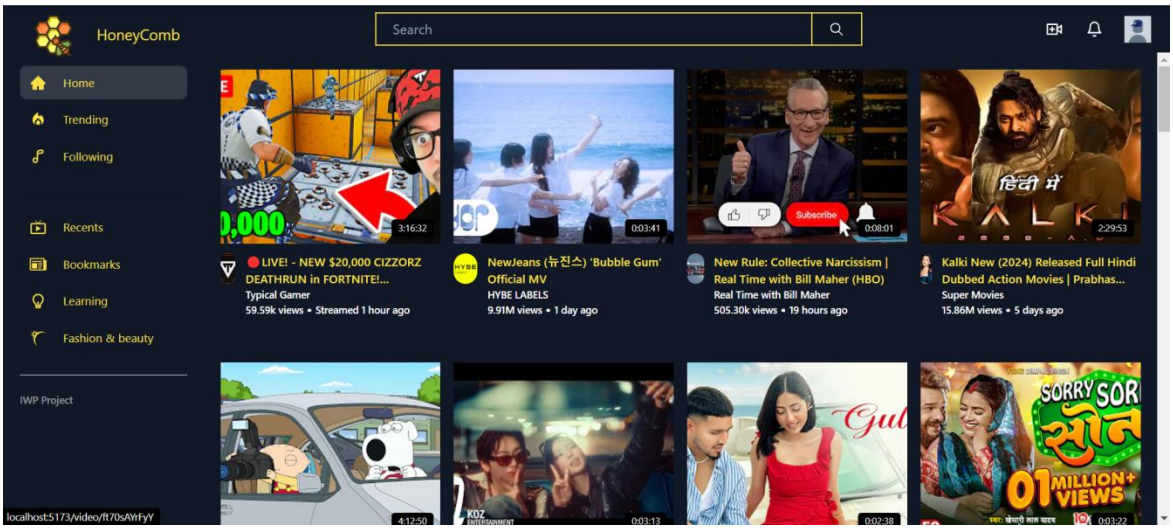
F. X-RapidAPI-Key: API key for authentication.

G. X-RapidAPI-Host: Host for the RapidAPI service.

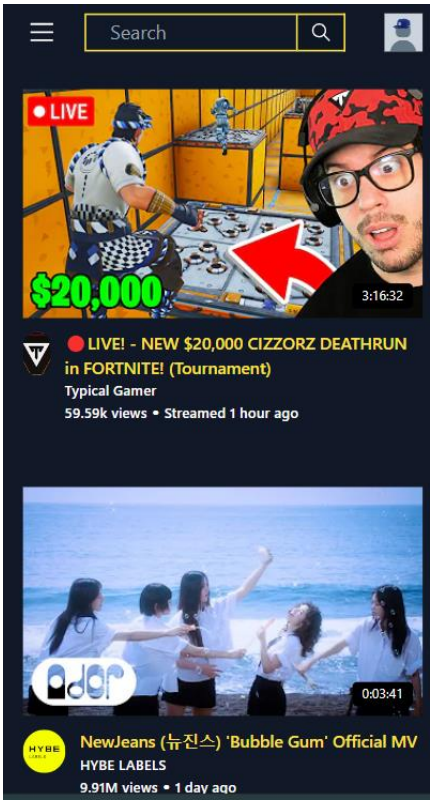
H. fetchDataFromApi(): This function makes an asynchronous GET request to the required URL.

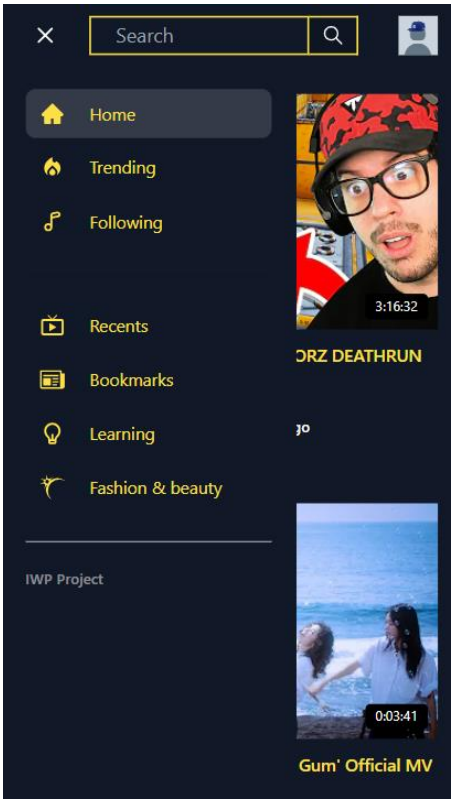
Screenshot Showcase

Landscape:



Portrait:





Conclusion

In conclusion, HoneyComb stands as a testament to the relentless pursuit of excellence in the realm of online video streaming. Through meticulous attention to detail and a steadfast commitment to innovation, HoneyComb has succeeded in delivering a comprehensive and immersive user experience. By harnessing the power of modern web technologies and incorporating user-centric features, HoneyComb has positioned itself as a formidable contender in the competitive landscape of video streaming platforms. As we continue to embrace the digital age, HoneyComb remains steadfast in its mission to redefine the way users engage with online video content. With a focus on user engagement, accessibility, and personalized features, HoneyComb is poised to shape the future of online video streaming, setting new standards for excellence and innovation in the process.