ДОДАТОК

Посилання на github проекту https://github.com/YouRlittleBetraYal/Diplomskr

Для успішного запуску необхідно встановити платформу Node.js версії 16.20.0 (це пов'язано з тим що на момент розробки використовувалась саме та версія на пк і програма на тій версії точно запуститься, на інших версіях не перевірено)

для інсталювання залежностей (фреймворків) необхідно як у директорії "server" так і у "client" написати команду таку "npm ci"

для самого запуску програм слід також в терміналі кожної директорії (server, client) написати "npm start"

Лістинг коду серверної частини,

Вміст package.json

```
{
  "name": "yt-clone-own-api",
  "name": "express-not-working-issue",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "nodemon server.js"
  },
  "author": "Harsg",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.0",
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "ffmpeg-static": "^5.1.0",
    "fluent-ffmpeg": "^2.1.2",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^2.0.16"
  }
}
```

Вміст файлу server.js

```
const http = require("http");
const app = require("./App");
const port = process.env.PORT || 5000;
const server = http.createServer(app);
server.listen(port, () => {
  console.log(`Listening on port ${port}!`);
});
```

Вміст файлу App.js

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const path = require("path");
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cors());
app.use("/api/video", require("./api/routes/video"));
app.use("/Uploads", express.static("Uploads"));
if (process.env.NODE_ENV === "production") {
  app.use(express.static("Client/build"));
  app.get("*", (req, res) => {
    res.sendFile(path.resolve(__dirname, "../Client", "build", "index.html"));
  });
}
module.exports = app;
```

віст файлу api\video.js

```
const express = require("express");
const router = express.Router();
const multer = require("multer");
const ffmpeg = require("fluent-ffmpeg");
const fileFilter = (req, file, cb) => {
  console.log("filefilter executed");
  if (file.mimetype === "video/mp4" || file.mimetype === "video/mpeg") {
    cb(null, true);
  } else {
    console.log("image type incorrect");
    cb("only Video files is allowed", false);
  }
};
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "Uploads/");
  },
  filename: function (req, file, cb) {
    const uniqueSuffix = Date.now() + "-" + file.originalname;
    cb(null, file.fieldname + "-" + uniqueSuffix);
  },
});
const upload = multer({ storage: storage, fileFilter: fileFilter }).single(
  "file"
```

```
);
router.post("/uploadfiles", (req, res) => {
  upload(req, res, (err) => {
    if (err) {
      return res.json({
        success: false,
        err,
      });
    }
    return res.json({
      success: true,
      filepath: res.req.file.path,
      fileName: res.req.file.filename,
    });
  });
});
router.post("/uploadThumbnail", (req, res) => {
  const fileFilterThumbnail = (req, file, cb) => {
    if (
      file.mimetype === "image/jpeg" ||
      file.mimetype === "image/apng" ||
      file.mimetype === "image/png"
    ) {
      cb(null, true);
    } else {
      console.log("image type incorrect");
      cb("Sorry bruh You cant have a **not image** thumbnail(for now)*", false);
    }
  };
  const storageThumbnail = multer.diskStorage({
    destination: function (req, file, cb) {
      cb(null, "Uploads/thumbnails/");
    },
    filename: function (req, file, cb) {
      const uniqueSuffix = Date.now() + "-" + file.originalname;
      cb(null, file.fieldname + "-" + uniqueSuffix);
    },
  });
  const uploadThumbnail = multer({
    storage: storageThumbnail,
    fileFilter: fileFilterThumbnail,
  }).single("file");
```

```
  uploadThumbnail(req, res, (err) => {
    if (err) {
      return res.json({
        success: false,
        err,
      });
    }
    return res.json({
      success: true,
      filepath: res.req.file.path,
      fileName: res.req.file.filename,
    });
  });
});
router.post("/thumbnail", (req, res) => {
  let Thumbsfilepath = [];
  let fileDuration = "";
  ffmpeg.ffprobe(req.body.filePath, (err, metadata) => {
    console.log(metadata.format.duration);
    fileDuration = metadata.format.duration;
  });
  ffmpeg(req.body.filePath)
    .on("filenames", function (filenames) {
      for (names in filenames) {
        Thumbsfilepath.push("Uploads/thumbnails/" + filenames[names]);
      }
    })
    .on("end", function () {
      console.log("Screenshots taken");
      return res.json({
        success: true,
        Thumbsfilepath: Thumbsfilepath,
        fileDuration: fileDuration,
      });
    })
    .screenshots({
      // Will take screens at 20%, 40%, 60% and 80% of the video
      count: 1,
      folder: "Uploads/thumbnails",
      size: "320x240",
      filename: "thumbnail-%b.png",
```

```
    });
});
module.exports = router;
```

Лістинг коду до клієнтської частини

Файл Package.json

```
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.2.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^0.27.2",
    "bootstrap": "^5.1.3",
    "firebase": "^9.8.2",
    "framer-motion": "^6.3.4",
    "get-google-fonts": "^1.2.2",
    "moment": "^2.29.3",
    "node-sass": "^7.0.1",
    "numeral": "^2.0.6",
    "react": "^18.1.0",
    "react-bootstrap": "^2.4.0",
    "react-dom": "^18.1.0",
    "react-dropzone": "^14.2.1",
    "react-helmet": "^6.1.0",
    "react-icons": "^4.3.1",
    "react-lazy-load-image-component": "^1.5.4",
    "react-notifications": "^1.7.3",
    "react-redux": "^8.0.2",
    "react-router-dom": "^6.3.0",
    "react-scripts": "5.0.1",
    "react-select": "^5.3.2",
    "react-toastify": "^9.0.1",
    "redux": "^4.2.0",
    "redux-thunk": "^2.4.1",
    "spinners-react": "^1.0.7",
    "styled-components": "^5.3.5",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
```

```
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Вміст файлу index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "bootstrap/dist/css/bootstrap.min.css";
import "react-lazy-load-image-component/src/effects/blur.css";

import App from "./App";
import "./_base.scss";

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

Вміст файлу categories,js

```
export const categories = [
  { value: 0, label: "Усі" },
  { value: 1, label: "Тваринки" },
  { value: 2, label: "Техніка" },
```

```
  { value: 3, label: "Природа" },
  { value: 4, label: "Навчання" },
  { value: 5, label: "Музика" },
  { value: 7, label: "Собаки" },
  { value: 8, label: "Софт" },
  { value: 9, label: "Сталкер2" },
  { value: 10, label: "Ігри" },
  { value: 11, label: "#programming" },
  { value: 12, label: "Онлайн ігри" },
  { value: 13, label: "Ігри на linux" },
  { value: 14, label: "#coding" },
  { value: 15, label: "Записи стрімів" },
  { value: 16, label: "Відео проходження" },
  { value: 17, label: "#Меми" },
  { value: 18, label: "#рецепти" },
  { value: 19, label: "#trending" },
  { value: 20, label: "#хендмейд" },
  { value: 21, label: "#techtips" },
  { value: 22, label: "#technews" },
  { value: 23, label: "#tech" },
  { value: 24, label: "#блогер" },
  { value: 25, label: "#Закордон" },
  { value: 26, label: "Github" },
  { value: 28, label: "Stackoverflow" },
  { value: 29, label: "Капібара" },
  { value: 27, label: "#різне" },
  { value: 30, label: "Казна що" },
];
```

Вміст файлу app.js

```
import { useState } from "react";
import { Container } from "react-bootstrap";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import "react-notifications/lib/notifications.css";
import { NotificationContainer } from "react-notifications";
import { createStore, combineReducers, applyMiddleware } from "redux";
import { Provider } from "react-redux";
import ReduxThunk from "redux-thunk";
import Header from "./components/header/Header";
import Homescreen from "./Screens/Homescreen/Homescreen";
import Sidebar from "./components/sidebar/Sidebar";
import Login from "./Screens/LoginScreen/Login";
import PagenotFound from "./Screens/PageNotFound/PagenotFound";
```

```
import ProtectedRoute from "./components/ProtectedRoute";
import ForgotPassword from "./Screens/ForgotPassword/ForgotPassword";
import UserActions from "./Screens/UserActions/UserActions";
import Upload from "./Screens/Upload/Uplaod";
import VideosReducer from "./Store/reducers/Videos";
import AuthReducer from "./Store/reducers/Auth";
import { AuthProvider } from "./Context/UserAuthContext";
import "./_app.scss";
import WatchScreen from "./Screens/WatchScreen/WatchScreen";
import SearchScreen from "./Screens/SearchScreen/SearchScreen";
import Subscriptions from "./Screens/subscriptions/subscriptions";
import ChannelScreen from "./Screens/channelScreen/ChannelScreen";
import EditVideo from "./Screens/EditVideoScreen/EditVideo";
const RootReducer = combineReducers({
  Vidoes: VideosReducer,
  auth: AuthReducer,
});
const store = createStore(RootReducer, applyMiddleware(ReduxThunk));
const Layout = ({ children }) => {
  const [toggleSidebar, settoggleSidebar] = useState(false);
  const handleToggleSidebar = () => {
    settoggleSidebar(!toggleSidebar);
  };
  return (
    <ProtectedRoute>
      <Header handleToggleSidebar={handleToggleSidebar} />
      <div className="app_container">
        <Sidebar
          handleToggleSidebar={handleToggleSidebar}
          toggleSidebar={toggleSidebar}
        />
        <Container className="app_main" fluid>
          {children}
        </Container>
      </div>
    </ProtectedRoute>
  );
};
function App() {
  const [containerAuth, togglecontainerAuth] = useState(true);
  const handleContainerLogin = () => {
    togglecontainerAuth(!containerAuth);
```

```
};
return (
  <Provider store={store}>
    <AuthProvider>
      <div className="App">
        <Router>
          <Routes>
            <Route
              path="/"
              element={
                <Layout>
                  <Homescreen />
                </Layout>
              }
            />
            <Route
              path="/login"
              element={
                <Login
                  containerAuth={containerAuth}
                  handleContainerLogin={handleContainerLogin}
                />
              }
            />
            <Route
              path="/search/:query"
              element={
                <Layout>
                  <SearchScreen />
                </Layout>
              }
            />
            <Route path="/forgot-password" element={<ForgotPassword />} />
            <Route path="/UserActions" element={<UserActions />} />
            <Route
              path="/Upload"
              element={
                <Layout>
                  <Upload />
                </Layout>
              }
            />
```

```
            <Route
              path="/watch/:id"
              element={
                <Layout>
                  <WatchScreen />
                </Layout>
              }
            />
            <Route
              path="/channel/:channelId"
              element={
                <Layout>
                  <ChannelScreen />
                </Layout>
              }
            />
            <Route
              path="/EditVideo/:id"
              element={
                <Layout>
                  <EditVideo />
                </Layout>
              }
            />
            <Route
              path="/feed/subscriptions"
              element={
                <Layout>
                  <Subscriptions />
                </Layout>
              }
            />
            <Route path="*" element={<PagenotFound />} />
          </Routes>
        </Router>
        <NotificationContainer />
      </div>
    </AuthProvider>
  </Provider>
 );
}
export default App;
```

Вміст файлу \reducers\Videos.js

```
import Video from "../../model/Video";
import Comment from "../../model/Comment";
import {
  CREATE_Video,
  SET_Vidoes,
  SET_VideoDetails,
  SET_Channel_Vidoes,
  SET_OwnerVidDetails,
  SET_RelatedVidoes,
  SET_Suscribers,
  SET_SearchVidoes,
  CREATE_Comment,
  CREATE_Like,
  CREATE_Dislike,
  REMOVE_Like,
  REMOVE_DisLike,
} from "../actions/Videos";
const initialState = {
  availableVideos: [],
  UserVidoes: [],
  relatedVids: [],
  SearchVids: [],
  Suscribers: [],
  video: null,
  VideoOwnerDetails: null,
};
export default (state = initialState, action) => {
  switch (action.type) {
    case CREATE_Video:
      const NewVideo = new Video(
        action.VideoData.id,
        action.VideoData.ownerId,
        action.VideoData.title,
        action.VideoData.thumbnail,
        action.VideoData.descriptions,
        action.VideoData.duration,
        action.VideoData.views,
        action.VideoData.Categories,
        action.VideoData.publicity,
        action.VideoData.filePath,
        action.VideoData.Usersname,
```

```
    action.VideoData.Userpfp,
    action.VideoData.timestamp,
    action.VideoData.comments,
    [],
    []
   );
   return {
    ...state,
    availableVideos: state.availableVideos.concat(NewVideo),
    UserVidoes: state.UserVidoes.concat(NewVideo),
   };
  case SET_Vidoes:
   return {
    ...state,
    availableVideos: action.videos,
    UserVidoes: action.userVidoes,
   };
  case SET_Channel_Vidoes:
   return {
    ...state,
    UserVidoes: action.userVidoes,
   };
  case SET_VideoDetails:
   return {
    ...state,
    video: action.video,
   };
  case SET_OwnerVidDetails:
   return {
    ...state,
    VideoOwnerDetails: action.Userdata,
   };
  case SET_RelatedVidoes:
   return {
    ...state,
    relatedVids: action.relatedvideos,
   };
  case SET_SearchVidoes:
   return {
    ...state,
    SearchVids: action.Searchvideos,
   };
```

```
case SET_Suscribers:
  return {
    ...state,
    Suscribers: action.Suscribers,
  };
case CREATE_Comment:
  const newComment = new Comment(
    action.CommentData.VideoId,
    action.CommentData.userId,
    action.CommentData.name,
    action.CommentData.userPfp,
    action.CommentData.Comment,
    action.CommentData.timestamp
  );

  const updatedVideo = {
    ...state.video,
    comments: state.video.comments.concat(newComment),
  };
  return {
    ...state,
    video: updatedVideo,
  };
case CREATE_Like:
  const updatedLikes = {
    ...state.video,
    likes: state.video.likes.concat(action.userId),
  };
  return {
    ...state,
    video: updatedLikes,
  };
case REMOVE_Like:
  const newlikesArr = [...state.video.likes];
  newlikesArr.splice(newlikesArr.indexOf(action.userId), 1);
  const updatedremovedLikes = {
    ...state.video,
    likes: newlikesArr,
  };
  return {
    ...state,
```

```
      video: updatedremovedLikes,
    };
  case REMOVE_DisLike:
    const newdislikesArr = [...state.video.dislikes];
    newdislikesArr.splice(newdislikesArr.indexOf(action.userId), 1);

    const updatedremovedDisLikes = {
      ...state.video,
      dislikes: newdislikesArr,
    };
    return {
      ...state,
      video: updatedremovedDisLikes,
    };
  case CREATE_Dislike:
    const updatedisLikes = {
      ...state.video,
      dislikes: state.video.dislikes.concat(action.userId),
    };
    return {
      ...state,
      video: updatedisLikes,
    };
  default:
    return state;
 }
};
```

Вміст файлу \reducers\Auth.js

```
import { SET_UserDetails } from "../actions/Auth";
const initialState = {
 userInfo: null,
};
export default (state = initialState, action) => {
 switch (action.type) {
  case SET_UserDetails:
    return {
      userInfo: action.Userdata,
    };
  default:
    return state;
 }
};
```

Вміст файлу actions\Videos.js

```
import { db, auth } from "../../Config/FirebaseConfig";
import {
  collection,
  addDoc,
  getDocs,
  getDoc,
  doc,
  arrayUnion,
  arrayRemove,
  updateDoc,
  serverTimestamp,
  query,
  where,
} from "firebase/firestore";
import Video from "../../model/Video";
import User from "../../model/UserDetails";
export const CREATE_Video = "CREATE_Video";
export const CREATE_Comment = "CREATE_Comment";
export const CREATE_Like = "CREATE_Like";
export const REMOVE_Like = "REMOVE_Like";
export const CREATE_Dislike = "CREATE_Dislike";
export const REMOVE_DisLike = "REMOVE_DisLike";
export const SET_Vidoes = "SET_Video";
export const SET_Channel_Vidoes = "SET_Channel_Vidoes";
export const SET_RelatedVidoes = "SET_RelatedVidoes";
export const SET_SearchVidoes = "SET_SearchVidoes";
export const SET_Suscribers = "SET_Suscribers";
export const SET_VideoDetails = "SET_VideoDetails";
export const SET_OwnerVidDetails = "SET_OwnerVidDetails";
const capitalizeFirstWords = (string) => {
  return string.replace(/(?:^|\s)\S/g, function (a) {
    return a.toUpperCase();
  });
};
export const Create_Video = (
  title,
  descriptions,
  publicity,
  Categories,
  duration,
  thumbnail,
```

```
  filePath,
  Usersname,
  Userpfp
) => {
  return async (dispatch) => {
    try {
      const VideoRef = addDoc(collection(db, "Videos"), {
        ownerId: auth.currentUser.uid,
        title: capitalizeFirstWords(title),
        descriptions: descriptions,
        duration: duration,
        thumbnail: thumbnail,
        filePath: filePath,
        publicity: publicity.value,
        Categories: Categories,
        Usersname: Usersname,
        Userpfp: Userpfp,
        views: [],
        timestamp: serverTimestamp(),
        comments: [],
        likes: [],
        dislikes: [],
      });
      dispatch({
        type: Create_Video,
        VideoData: {
          id: VideoRef.id,
          ownerId: auth.currentUser.uid,
          title: title,
          descriptions: descriptions,
          duration: duration,
          thumbnail: thumbnail[0],
          filePath: filePath,
          publicity: publicity.value,
          Categories: Categories,
          Usersname: Usersname,
          Userpfp: Userpfp,
          views: [],
          timestamp: new Date(),
          comments: [],
          likes: [],
          dislikes: [],
```

```
        },
      });
    } catch {
      throw new Error("Something went wrong!");
    }
  };
};
export const Update_Video = (
  name,
  description,
  publicity,
  thumbnail,
  VideoId
) => {
  return async (dispatch) => {
    try {
      console.log(thumbnail);
      const VidRef = doc(db, "Videos", VideoId);
      await updateDoc(VidRef, {
        title: name,
        descriptions: description,
        publicity: publicity,
        thumbnail: thumbnail,
      });
    } catch (error) {
      console.error(error);
      throw new Error("Something went wrong!");
    }
  };
};
export const Create_Comment = (VideoId, Comment, userPfp, name) => {
  return async (dispatch) => {
    try {
      const userId = auth.currentUser.uid;
      const VidRef = doc(db, "Videos", VideoId);
      updateDoc(VidRef, {
        comments: arrayUnion({
          VideoId,
          userId,
          userPfp,
          name,
          Comment,
```

```
        timestamp: new Date(),
      }),
    });
    dispatch({
      type: CREATE_Comment,
      CommentData: {
        VideoId,
        userId,
        userPfp,
        name,
        Comment,
        timestamp: new Date(),
      },
    });
  } catch (err) {
    console.log(err);
    throw new Error("Something went wrong!");
  }
 };
};
export const Create_Like = (VideoId) => {
 return async (dispatch) => {
  try {
    const userId = auth.currentUser.uid;
    const VidRef = doc(db, "Videos", VideoId);
    updateDoc(VidRef, {
      likes: arrayUnion(userId),
    });
    dispatch({
      type: CREATE_Like,
      userId: userId,
    });
  } catch (err) {
    console.log(err);
    throw new Error("Something went wrong!");
  }
 };
};
export const Remove_Like = (VideoId) => {
 return async (dispatch) => {
  try {
    const userId = auth.currentUser.uid;
```

```
      const VidRef = doc(db, "Videos", VideoId);
      updateDoc(VidRef, {
        likes: arrayRemove(userId),
      });
      dispatch({
        type: REMOVE_Like,
        userId: userId,
      });
    } catch (err) {
      console.log(err);
      throw new Error("Something went wrong!");
    }
  };
};
export const Remove_DisLike = (VideoId) => {
  return async (dispatch) => {
    try {
      const userId = auth.currentUser.uid;
      const VidRef = doc(db, "Videos", VideoId);
      updateDoc(VidRef, {
        dislikes: arrayRemove(userId),
      });
      dispatch({
        type: REMOVE_DisLike,
        userId: userId,
      });
    } catch (err) {
      console.log(err);
      throw new Error("Something went wrong!");
    }
  };
};
export const Create_DisLike = (VideoId) => {
  return async (dispatch) => {
    try {
      const userId = auth.currentUser.uid;
      const VidRef = doc(db, "Videos", VideoId);
      updateDoc(VidRef, {
        dislikes: arrayUnion(userId),
      });
      dispatch({
        type: CREATE_Dislike,
```

```
        userId: userId,
      });
    } catch (err) {
      console.log(err);
      throw new Error("Something went wrong!");
    }
  };
};
export const fetchvidoes = (category) => {
  return async (dispatch) => {
    try {
      const userId = auth.currentUser.uid;
      const VideosRef = collection(db, "Videos");
      let VideoSnapshot;
      if (!category || category === "All") {
        const q = query(VideosRef, where("publicity", "==", "Public"));
        VideoSnapshot = await getDocs(q);
      } else {
        const q = query(
          VideosRef,
          where("Categories", "array-contains", category),
          where("publicity", "==", "Public")
        );
        VideoSnapshot = await getDocs(q);
      }
      const videostobeloaded = [];
      VideoSnapshot.forEach((doc) => {
        videostobeloaded.push(
          new Video(
            doc.id,
            doc.data().ownerId,
            doc.data().title,
            doc.data().thumbnail,
            doc.data().descriptions,
            doc.data().duration,
            doc.data().views,
            doc.data().Categories,
            doc.data().publicity,
            doc.data().filePath,
            doc.data().Usersname,
            doc.data().Userpfp,
            doc.data().timestamp,
```

```
            doc.data().comments,
            doc.data().likes,
            doc.data().dislikes
          )
        );
      });
      dispatch({
        type: SET_Vidoes,
        videos: videostobeloaded,
        userVidoes: videostobeloaded.filter((vid) => vid.OwnerId == userId),
      });
    } catch (error) {
      throw error;
    }
  };
};
export const fetchChannelVideos = (ChannelId) => {
  return async (dispatch) => {
    try {
      const VideosRef = collection(db, "Videos");
      let que;
      if (ChannelId === auth.currentUser.uid) {
        que = query(VideosRef, where("ownerId", "==", ChannelId));
      } else {
        que = query(
          VideosRef,
          where("ownerId", "==", ChannelId),
          where("publicity", "==", "Public")
        );
      }
      const querySnapshot = await getDocs(que);
      const ChannelVidstobeloaded = [];
      querySnapshot.forEach((doc) => {
        ChannelVidstobeloaded.push(
          new Video(
            doc.id,
            doc.data().ownerId,
            doc.data().title,
            doc.data().thumbnail,
            doc.data().descriptions,
            doc.data().duration,
            doc.data().views,
```

```
            doc.data().Categories,
            doc.data().publicity,
            doc.data().filePath,
            doc.data().Usersname,
            doc.data().Userpfp,
            doc.data().timestamp,
            doc.data().comments,
            doc.data().likes,
            doc.data().dislikes
          )
        );
      });
      dispatch({
        type: SET_Channel_Vidoes,
        userVidoes: ChannelVidstobeloaded,
      });
    } catch (error) {
      throw error;
    }
  };
};
export const fetchSearchvidoes = (searchquery) => {
  return async (dispatch) => {
    const CaseSearchquery = capitalizeFirstWords(searchquery);
    try {
      const VideosRef = collection(db, "Videos");
      const q = query(
        VideosRef,
        where("title", ">=", CaseSearchquery),
        where("title", "<=", CaseSearchquery + "\uf8ff"),
        where("publicity", "==", "Public")
      );
      const querySnapshot = await getDocs(q);
      const Searchvideostobeloaded = [];
      querySnapshot.forEach((doc) => {
        Searchvideostobeloaded.push(
          new Video(
            doc.id,
            doc.data().ownerId,
            doc.data().title,
            doc.data().thumbnail,
            doc.data().descriptions,
```

```
            doc.data().duration,
            doc.data().views,
            doc.data().Categories,
            doc.data().publicity,
            doc.data().filePath,
            doc.data().Usersname,
            doc.data().Userpfp,
            doc.data().timestamp,
            doc.data().comments,
            doc.data().likes,
            doc.data().dislikes
          )
        );
      });
      dispatch({
        type: SET_SearchVidoes,
        Searchvideos: Searchvideostobeloaded,
      });
    } catch (error) {
      console.log(error);
      throw error;
    }
  };
};
export const fetchrelatedvideos = (category) => {
  return async (dispatch) => {
    try {
      const VideosRef = collection(db, "Videos");
      const q = query(
        VideosRef,
        where("Categories", "array-contains-any", category),
        where("publicity", "==", "Public")
      );
      const querySnapshot = await getDocs(q);
      const relatedvideostobeloaded = [];
      querySnapshot.forEach((doc) => {
        relatedvideostobeloaded.push(
          new Video(
            doc.id,
            doc.data().ownerId,
            doc.data().title,
            doc.data().thumbnail,
```

```
            doc.data().descriptions,
            doc.data().duration,
            doc.data().views,
            doc.data().Categories,
            doc.data().publicity,
            doc.data().filePath,
            doc.data().Usersname,
            doc.data().Userpfp,
            doc.data().timestamp,
            doc.data().comments,
            doc.data().likes,
            doc.data().dislikes
          )
        );
      });
      dispatch({
        type: SET_RelatedVidoes,
        relatedvideos: relatedvideostobeloaded,
      });
    } catch (error) {
      console.log(error);
      throw error;
    }
  };
};
export const fetchvideo = (id) => {
  return async (dispatch) => {
    try {
      const VideoRef = doc(db, "Videos", id);
      const VideoSnap = await getDoc(VideoRef);
      const VideoDetail = new Video(
        id,
        VideoSnap.data().ownerId,
        VideoSnap.data().title,
        VideoSnap.data().thumbnail,
        VideoSnap.data().descriptions,
        VideoSnap.data().duration,
        VideoSnap.data().views,
        VideoSnap.data().Categories,
        VideoSnap.data().publicity,
        VideoSnap.data().filePath,
        VideoSnap.data().Usersname,
```

```
        VideoSnap.data().Userpfp,
        VideoSnap.data().timestamp,
        VideoSnap.data().comments,
        VideoSnap.data().likes,
        VideoSnap.data().dislikes
      );
      dispatch({
        type: SET_VideoDetails,
        video: VideoDetail,
        id: id,
      });
    } catch (error) {
      throw error;
    }
  };
};
export const fetchUserDetails = (OwnerId) => {
  return async (dispatch) => {
    try {
      const UserDocRef = doc(db, "Users", OwnerId);
      const Usersnap = await getDoc(UserDocRef);
      const Usertobeloaded = new User(
        OwnerId,
        Usersnap.data().name,
        Usersnap.data().profileImage,
        Usersnap.data().initials,
        Usersnap.data().Suscribers
      );
      dispatch({
        type: SET_OwnerVidDetails,
        Userdata: Usertobeloaded,
      });
    } catch (error) {
      throw error;
    }
  };
};

export const loadSubscriptions = () => {
  return async (dispatch) => {
    try {
      const userId = auth.currentUser.uid;
```

```
      const UsersRef = collection(db, "Users");
      const q = query(UsersRef, where("Suscribers", "array-contains", userId));
      const querySnapshot = await getDocs(q);
      const Suscriberstobeloaded = [];
      querySnapshot.forEach((doc) => {
        Suscriberstobeloaded.push(
          new User(
            doc.id,
            doc.data().name,
            doc.data().profileImage,
            doc.data().initials,
            doc.data().Suscribers
          )
        );
      });
      dispatch({
        type: SET_Suscribers,
        Suscribers: Suscriberstobeloaded,
      });
    } catch (error) {
      throw error;
    }
  };
};
export const Subscribe = (OwnerId) => {
  return new Promise((resolve, reject) => {
    try {
      const userId = auth.currentUser.uid;
      const VidOwnerRef = doc(db, "Users", OwnerId);
      updateDoc(VidOwnerRef, {
        Suscribers: arrayUnion(userId),
      });
      resolve("Suscribed !!!");
    } catch (error) {
      reject(Error("Failed :< didn't work!"));
    }
  });
};
export const UnSubscribe = (OwnerId) => {
  return new Promise((resolve, reject) => {
    try {
      const userId = auth.currentUser.uid;
```

```
      const VidOwnerRef = doc(db, "Users", OwnerId);
      updateDoc(VidOwnerRef, {
        Suscribers: arrayRemove(userId),
      });
      console.log("Unsubscribed");
      resolve("UnSuscribed !!!");
    } catch (error) {
      reject(Error("Failed :< didn't work!"));
    }
  });
};
export const SetView = (VideoId, Ip) => {
  return new Promise((resolve, reject) => {
    try {
      const VidRef = doc(db, "Videos", VideoId);
      updateDoc(VidRef, {
        views: arrayUnion(Ip.ip),
      });
      resolve("Suscribed !!!");
    } catch (error) {
      reject(Error("Failed :< didn't work!"));
    }
  });
};
```

Вміст файлу \actions\auth.js

```
import { db, auth } from "../../Config/FirebaseConfig";
import { getDoc, doc, setDoc } from "firebase/firestore";
import User from "../../model/UserDetails";
export const SET_UserDetails = "SET_UserDetails";
export const SET_Suscibers = "SET_Suscibers";
export const fetchUserDetails = () => {
  return async (dispatch) => {
    const userId = auth.currentUser.uid;
    const UserDocRef = doc(db, "Users", userId);
    const Usersnap = await getDoc(UserDocRef);
    if (Usersnap.exists()) {
      const Usertobeloaded = new User(
        userId,
        Usersnap.data().name,
        Usersnap.data().profileImage,
        Usersnap.data().initials,
        Usersnap.data().Suscribers
```

```
      );
      dispatch({
        type: SET_UserDetails,
        Userdata: Usertobeloaded,
      });
    } else {
      console.log("No such User!");
    }
  };
};
```

Вміст файлу \model\Videos,js

```
class Video {
  constructor(
    id,
    OwnerId,
    name,
    thumbnail,
    description,
    duration,
    views,
    category,
    publicity,
    filepath,
    Usersname,
    Userpfp,
    timestamp,
    comments,
    likes,
    dislikes
  ) {
    this.id = id;
    this.name = name;
    this.OwnerId = OwnerId;
    this.thumbnail = thumbnail;
    this.description = description;
    this.duration = duration;
    this.views = views;
    this.category = category;
    this.publicity = publicity;
    this.filepath = filepath;
    this.Usersname = Usersname;
    this.Userpfp = Userpfp;
```

```
    this.timestamp = timestamp;
    this.comments = comments;
    this.likes = likes;
    this.dislikes = dislikes;
  }
}
export default Video;
```

Вміст файлу \model\UserDetails.js

```
class User {
  constructor(id, name, profileImage, initials, Suscribers) {
    this.id = id;
    this.name = name;
    this.profileImage = profileImage;
    this.initials = initials;
    this.Suscribers = Suscribers;
  }
}
export default User;
```

Вміст файлу \model\comments.js

```
class Comment {
  constructor(VideoId, userId, name, profileImage, Comment, timestamp) {
    this.name = name;
    this.userPfp = profileImage;
    this.Comment = Comment;
    this.userId = userId;
    this.VideoId = VideoId;
    this.timestamp = timestamp;
  }
}
export default Comment;
```

Вміст файлу \contextl\UserAuthContext.js

```
import React, { useContext, useState, useEffect } from "react";
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
  signOut,
  GoogleAuthProvider,
  signInWithPopup,
  sendPasswordResetEmail,
  confirmPasswordReset,
  sendSignInLinkToEmail,
```

```
  isSignInWithEmailLink,
  signInWithEmailLink,
} from "firebase/auth";
import { doc, setDoc } from "firebase/firestore";
import { db, auth } from "../Config/FirebaseConfig";
const AuthContext = React.createContext();
export const useAuth = () => {
  const auth = useContext(AuthContext);
  return { ...auth, isAuthenticated: auth.CurrentUser != null };
};
export function AuthProvider({ children }) {
  const [CurrentUser, setCurrentUser] = useState();
  const [loading, setloading] = useState(true);

  useEffect(() => {
    const unsuscribe = onAuthStateChanged(auth, (user) => {
      console.log(user);
      setCurrentUser(user);
      setloading(false);
    });
    return unsuscribe;
  }, []);
  const SignupUser = (email, password, name) => {
    return createUserWithEmailAndPassword(auth, email, password).then(
      (resp) => {
        if (resp.user) {
          CreateFireStoreUser(
            resp.user,
            name,
            "https://cdn-icons-png.flaticon.com/512/149/149071.png"
          );
        }
      }
    );
  };
  const CreateFireStoreUser = (user, name, profileImage) => {
    const UsersRef = doc(db, "Users", user.uid);
    setDoc(UsersRef, {
      name: name,
      profileImage: profileImage,
      initials: name[0] + name[name.length - 1],
      Suscribers: [],
```

```
  }).then(() => {
    console.log("added User");
  });
};
const LogInUser = (email, password) => {
  return signInWithEmailAndPassword(auth, email, password);
};
const SendEmailVerification = () => {
  const actionCodeSettings = {
    url: "http://localhost:3000",
    handleCodeInApp: true,
  };
  return sendSignInLinkToEmail(auth, CurrentUser.email, actionCodeSettings);
};
const VerifyEmailLink = () => {
  if (isSignInWithEmailLink(auth, window.location.href)) {
    return signInWithEmailLink(auth, CurrentUser.email);
  }
};
const SignInWithGoogle = () => {
  const provider = new GoogleAuthProvider();
  return signInWithPopup(auth, provider);
};
const forgotPassword = (email) => {
  // This function is not used in any files till YET
  return sendPasswordResetEmail(auth, email, {
    url: "http://localhost:3000/login",
  });
};
const ResetPassword = (oobcode, newPasword) => {
  return confirmPasswordReset(auth, oobcode, newPasword);
};
const logout = () => {
  return signOut(auth);
};
const value = {
  CurrentUser,
  SignupUser,
  LogInUser,
  logout,
  SignInWithGoogle,
  forgotPassword,
```

```
    ResetPassword,
    CreateFireStoreUser,
    SendEmailVerification,
    VerifyEmailLink,
  };
  return (
    <AuthContext.Provider value={value}>
      {!loading && children}
    </AuthContext.Provider>
  );
}
```

Вміст файлу \config\ FirebaseConfig.js

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";
const firebaseConfig = {
  apiKey: "AIzaSyCoqaTfAPXTyB9okrKNN8opZNLOlshC7ac",
  authDomain: "mydemoapp-c9a6a.firebaseapp.com",
  projectId: "mydemoapp-c9a6a",
  storageBucket: "mydemoapp-c9a6a.appspot.com",
  messagingSenderId: "708439279811",
  appId: "1:708439279811:web:d1298c8a144e8d2f43fa76"
};
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);
const auth = getAuth(app);
export { auth, db };
```

Вміст файлу \screens\watchscreen\watchscreen.js

```
import React, { useState, useEffect, useCallback } from "react";
import { Col, Row } from "react-bootstrap";
import { useSelector, useDispatch } from "react-redux";
import axios from "axios";
import { NotificationManager } from "react-notifications";
import { useParams } from "react-router-dom";
import { SpinnerCircular } from "spinners-react";
import { Helmet } from "react-helmet";
import VideoMetaData from "../../components/VideoMetaData/VideoMetaData";
import VideoHorizontal from "../../components/VideoHorizontal/VideoHorizontal";
import Comments from "../../components/comments/Comments";
import Videoplayer from "../../components/VideoPlayer/Videoplayer";
import * as VideoActions from "../../Store/actions/Videos";
```

```
function WatchScreen() {
  const [error, seterror] = useState();
  const [loading, setloading] = useState(false);
  const dispatch = useDispatch();
  const { id } = useParams();
  const videoDetails = useSelector((state) => state.Vidoes.video);
  const centeredItem = {
    height: "100vh",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
  };
  const loadVideoDetails = useCallback(async () => {
    seterror(null);
    try {
      await dispatch(VideoActions.fetchvideo(id));
    } catch (error) {
      seterror(error.message);
      console.log(error.message);
    }
  }, [dispatch, seterror]);
  useEffect(() => {
    if (error) {
      NotificationManager.error(error, "Error", 10000);
    }
  }, [error]);
  useEffect(() => {
    setloading(true);
    loadVideoDetails().then(() => {
      setloading(false);
    });
  }, [dispatch, loadVideoDetails, setloading]);
  const SetViews = useCallback(async () => {
    axios
      .get("https://api.ipify.org/?format=json")
      .then(async (res) => {
        seterror(null);
        VideoActions.SetView(id, res.data).catch((error) => {
          seterror(error.message);
        });
      })
      .catch((error) => {
```

```
      seterror(error);
      console.log(error);
    });
}, [dispatch, seterror, videoDetails]);
useEffect(() => {
  SetViews();
}, []);
if (loading) {
  return (
    <div style={centeredItem}>
      <SpinnerCircular color="#00BFFF" size={30} />
    </div>
  );
}
if (!loading && videoDetails === null) {
  return <div style={centeredItem}>Ми не можемо завантажити відео вибачте</div>;
}
if (error) {
  return (
    <div style={centeredItem}>
      Помилка!
      <button onClick={loadVideoDetails}>Спробуйте знову</button>
    </div>
  );
}
return (
  <div>
    <Helmet>
      <title>{videoDetails.name}</title>
    </Helmet>
    <Videoplayer
      vidsrc={videoDetails.filepath}
      duration={videoDetails.duration}
    />
    <Row>
      <Col lg={8}>
        <VideoMetaData
          views={videoDetails.views}
          title={videoDetails.name}
          description={videoDetails.description}
          OwnerId={videoDetails.OwnerId}
          timestamp={new Date(videoDetails.timestamp.toDate()).toUTCString()}
```

```
          VideoId={id}
        />
        <Comments
          VideoId={id}
          userPfp={videoDetails.Userpfp}
          name={videoDetails.Usersname}
        />
      </Col>
      <Col lg={4}>
        <VideoHorizontal category={videoDetails.category} VideoId={id} />
      </Col>
    </Row>
  </div>
 );
}
export default WatchScreen;
```

Вміст файлу \screens\verifyEmail\ verifyEmail.js

```
import React, { useEffect } from "react";
import { useLocation } from "react-router-dom";
import { useAuth } from "../../Contexts/Context";
const VeryifyEmail = () => {
 const { SendEmailVerification } = useAuth();
 useEffect(() => {
  SendEmailVerification()
   .then(() => {
    alert("An email has been sent");
   })
   .catch((e) => {
    console.log(e);
   });
 }, []);
 return <div>Ви не верифіковані</div>;
};
export default VeryifyEmail;
```

Вміст файлу \screens\UserActions\ UserActions.js

```
import React from "react";
import { useLocation } from "react-router-dom";
import ResetPassword from "../../components/ResetPassword/ResetPassword";
import EmailVerification from "../../components/EmailVerification/EmailVerification";
const useQuery = () => {
 return new URLSearchParams(useLocation().search);
};
```

```
const UserActions = () => {
  const query = useQuery();
  const mode = query.get("mode");
  const actionCode = query.get("oobCode");
  const continueUrl = new URL(query.get("continueUrl")).pathname;
  const lang = query.get("lang") || "en";
  const renderSwitch = () => {
    switch (mode) {
      case "resetPassword":
        return (
          <ResetPassword actionCode={actionCode} continueUrl={continueUrl} />
        );
      case "signIn":
        return (
          <EmailVerification
            actionCode={actionCode}
            continueUrl={continueUrl}
          />
        );
    }
  };
  return <div>{renderSwitch()}</div>;
};
export default UserActions;
```

Вміст файлу \screens\Upload\upload.js

```
import React, { useReducer, useCallback, useState, useMemo } from "react";
import { useDropzone } from "react-dropzone";
import { AiOutlineProfile } from "react-icons/ai";
import Select from "react-select";
import makeAnimated from "react-select/animated";
import axios from "axios";
import { NotificationManager } from "react-notifications";
import { useDispatch, useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import "./_upload.scss";
import * as VideoActions from "../../Store/actions/Videos";
import Input from "../../components/Input/Input";
import { categories } from "../../categories";
const Private = [
  { value: "Public", label: "Публічне" },
  { value: "Private", label: "Приватне" },
];
```

```
const formReducer = (state, action) => {
  // It's just just like a redux reducer but not not connected to redux at all
  if (action.type === "FORM_INPUT_UPDATE") {
    // when we change any input
    const updatedValues = {
      // Change our inputValues state
      ...state.inputValues, // Return all other inputs values
      [action.input]: action.value, // But change the value of that specific input
    };
    const updateValidities = {
      // Change our inputValidities state
      ...state.inputValidities,
      [action.input]: action.isValid,
    };
    let FormisValid = false; // Initiallt its false
    for (const key in updateValidities) {
      //  If all inputs are valid then change it to true
      FormisValid = FormisValid && updateValidities[key];
    }
    return {
      formIsValid: FormisValid,
      inputValidities: updateValidities,
      inputValues: updatedValues,
    };
  }
  return state; // return our state
};
const baseStyle = {
  flex: 1,
  display: "flex",
  flexDirection: "column",
  alignItems: "center",
  padding: "20px",
  borderWidth: 2,
  borderRadius: 2,
  borderColor: "#eeeeee",
  borderStyle: "dashed",
  backgroundColor: "#fafafa",
  color: "#bdbdbd",
  outline: "none",
  transition: "border .24s ease-in-out",
};
```

```
const focusedStyle = {
  borderColor: "#2196f3",
};
const acceptStyle = {
  borderColor: "#00e676",
};
const rejectStyle = {
  borderColor: "#ff1744",
};
function Uplaod() {
  const animatedComponents = makeAnimated();
  const [publicity, setpublicity] = useState();
  const [Categories, setCategories] = useState();
  const [Files, setFiles] = useState();
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const Userinfo = useSelector((state) => state.auth.userInfo);
  const {
    getRootProps,
    getInputProps,
    isFocused,
    isDragAccept,
    isDragReject,
    acceptedFiles,
  } = useDropzone({
    multiple: false,
    maxSize: 800000000,
    onDrop: (acceptedfile) => {
      setFiles(acceptedfile);
    },
    accept: { "video/*": [] },
  });
  const [formState, dispatchformState] = useReducer(formReducer, {
    // Our useReducer describing all our states
    inputValues: {
      Title: "",
      descriptions: "",
    },
    inputValidities: {
      Title: false,
      descriptions: false,
    },
```

```
  formIsValid: false,
});
const inputChangeHandler = useCallback(
  // Our inputChangeHandler func
  (inputIdentifier, inputvalue, inputisValid) => {
    dispatchformState({
      type: "FORM_INPUT_UPDATE",
      value: inputvalue,
      isValid: inputisValid,
      input: inputIdentifier,
    });
  },
  [dispatchformState]
);
const UplaodHandler = async () => {
  let variable;
  let thumbnail;
  let duration;
  let formData = new FormData();
  formData.append("file", Files[0]);
  axios({
    method: "POST",
    url: "http://localhost:5000/api/video/uploadfiles",
    headers: { "Content-Type": "multipart/form-data" },
    data: formData,
  }).then((response) => {
    if (response.data.success) {
      variable = {
        filePath: response.data.filepath,
        fileName: response.data.fileName,
      };
      NotificationManager.success(
        "Файл завантажено....Генерація мініатюри...",
        "успіх"
      );
      axios({
        method: "POST",
        url: "http://localhost:5000/api/video/thumbnail",
        data: variable,
      })
        .then((response) => {
          if (response.data.success) {
```

```
            thumbnail = response.data.Thumbsfilepath;
            duration = response.data.fileDuration;
            NotificationManager.success(
              "Мініатюру згенеровано успішно!",
              "успіх"
            );
          } else {
            NotificationManager.error(response.data.err, "Error", 10000);
          }
        })
        .finally(() => {
          onclickHandler(variable, thumbnail, duration);
        });
    } else {
      NotificationManager.error(response.data.err, "Error", 10000);
    }
  });
};
const onclickHandler = async (variable, thumbnail, duration) => {
  let categories = Categories.map((category) => category["label"]);
  await dispatch(
    VideoActions.Create_Video(
      formState.inputValues.Title,
      formState.inputValues.descriptions,
      publicity,
      categories,
      duration,
      `http://localhost:5000/${thumbnail}`,
      variable.filePath,
      Userinfo.name,
      Userinfo.profileImage
    )
  );
  navigate("/", { replace: true });
};
const style = useMemo(
  () => ({
    ...baseStyle,
    ...(isFocused ? focusedStyle : {}),
    ...(isDragAccept ? acceptStyle : {}),
    ...(isDragReject ? rejectStyle : {}),
  }),
```

```
  [isFocused, isDragAccept, isDragReject]
);

return (
  <div className="main">
    <div className="main_Container">
      <section>
        <div
          {...getRootProps({ style })}
          className="main_file-upload-wrapper"
        >
          <input {...getInputProps()} />
          {!isDragReject &&
            "Перетягніть сюди файл або натисніть та виберіть"}
          {isDragReject && "увага! формат файлу не підтримується"}
        </div>
        <ul className="list-group mt-2">
          {acceptedFiles.length > 0 &&
            acceptedFiles.map((acceptedFile) => (
              <li className="list-group-item list-group-item-success">
                {acceptedFile.name}
              </li>
            ))}
        </ul>
        <Input
          id="Title" // field Id
          label="Введіть назву" // Label for our Input
          minLength={5} // A minimum length of 5 or else error
          initialvalue="" // Initial Value for our input. i.e ""
          required // It is a required field
          inputchange={inputChangeHandler} // Do this when text inout value changes
          initiallyvalid="false"
          Icon={AiOutlineProfile}
        />
        <Input
          id="descriptions" // field Id
          label="Напишіть опис, не менше 15 символів" // Label for our Input
          textarea
          initialvalue="" // I    nitial Value for our input. i.e ""
          minLength={15} // A minimum length of 10 or else error
          required // It is a required field
          inputchange={inputChangeHandler} // // Do this when text inout value changes
```