KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

Đồ án công nghệ phần mềm

CODING STANDARDS

Version 1.0

Bảng ghi nhận thay đổi tài liệu

Ngày	Phiên bản	Mô tả	Người thay đổi
01/4/2024	1.0	Viết code standard cho Backend và Frontend	Nghĩa

CODING STANDARDS

Quy chuẩn mã nguồn cho Backend:

Tổng quát:

- Thống nhất thụt lề (4 dấu cách)
- Sử dụng quy cách đặt tên chuẩn của Java (camelCase cho biến và phương thức, PascalCase cho lớp).
- Sử dụng Comment để giải thích các đoạn code phức tạp
- Đặt tên biến có nghĩa và mô tả tốt các biến, lớp, phương thức, hạn chế viết tắt.

IntelliJ:

- Áp dụng định dạng nhất quán bằng cách sử dụng cài đặt kiểu mã của IntelliJ (ví dụ: Editor > Code Style > Java).
- Sử dụng kiểm tra mã và sửa lỗi nhanh để duy trì chất lượng mã.
- Sử dụng hệ thống kiểm soát phiên bản như Git và tích hợp nó với IntelliJ để theo dõi phiên bản và cộng tác.

Quy chuẩn mã nguồn cho Frontend (ReactJS, Tailwind CSS và HTML):

ReactJS:

- Sử dụng: ReactJS các tính năng như arrow functions, destructuring, template literals, để làm mã nguồn ngắn gọn và hiệu quả.
- Tên file và thư mục: chữ thường đặt tên các folder, tên file component thì viết hoa chữ cái đầu.
- Tổ chức thư mục: Chia ứng dụng thành các component nhỏ với chức năng cụ thể để tăng tính tái sử dụng và dễ bảo trì. Nếu một component trở nên quá lớn, chia nó thành các component nhỏ hơn để dễ bảo trì.
- Sử dụng function component
- Tên component: Sử dụng kiểu PascalCase (chữ cái đầu tiên của mỗi từ được viết hoa) cho tên component. Ví dụ: Header, Sidebar, …
- Tên phương thức: Sử dụng camelCase cho tên biến và hàm (ví dụ: sendMessage, handleUnfriend, …).
- Sử dụng Prettier extension và ESlint giúp thống nhất coding convention

```
const AppContext = createContext()
const AppContextProvider = ({children}) => {
    const [state, dispatch] = useReducer(appReducer, initialState)

    return (
        <AppContext.Provider value = {{state, dispatch}}>
            {children}
        </AppContext.Provider>
    )
}

const useAppContext = () => {
    const context = useContext(AppContext);
  if (!context) {
    throw new Error('useAppContext must be used within an AppContextProvider');
  }
  return context;
}

export { AppContextProvider, useAppContext };
```

- Tách Hành Động ra Khỏi Component: Tạo tệp actions.js để định nghĩa các hành động (actions) và actionTypes để quản lý các type.

```
export const loginRequest = () => {
  return {
    type: actionTypes.LOGIN_REQUEST,
  };
};

export const loginSuccess = (userData) => {
  return {
    type: actionTypes.LOGIN_SUCCESS,
    payload: userData,
  };
};
```

```
// actionTypes.js
export const LOGIN_REQUEST = 'LOGIN_REQUEST';
export const LOGIN_SUCCESS = 'LOGIN_SUCCESS';
export const LOGIN_FAILURE = 'LOGIN_FAILURE';
export const REGISTER_REQUESTE= 'REGISTER_REQUEST';
export const REGISTER_SUCCESS = 'REGISTER_SUCCESS';
export const REGISTER_FAILURE = 'REGISTER_FAILURE';
export const LOGOUT_USER = 'LOGOUT_USER';

export const SET_LOGGED_IN = 'SET_LOGGED_IN';

export const SEARCH_FRIEND = 'SEARCH_FRIEND'

export const GET_CHAT_ROOMS = 'GET_CHAT_ROOMS'
export const GET_REQUESTS = 'GET_REQUESTS'
export const GET_FRIENDS = 'GET_FRIENDS'
export const GET_SENDS = 'GET_SENDS'

export const JOIN_ROOM = 'JOIN_ROOM'
export const USER_ROOM = 'USER_ROOM'
export const GROUP_ROOM = 'GROUP_ROOM'
export const SINGLE_ROOM = 'SINGLE_ROOM'
export const GET_MESSAGES = 'GET_MESSAGES'
export const CREATE_CHAT_ROOM = 'CREATE_CHAT_ROOM'

export const UPDATE_PROFILE = 'UPDATE_PROFILE'
```

- Tạo Reducer: Tạo tệp reducer.js để xử lý các hành động và cập nhật trạng thái.

```
import * as actionTypes from './../actions/actionTypes'
const appReducer = (state, action) => {
    switch (action.type) {
        case actionTypes.SET_LOGGED_IN:
            return {
                ...state,
                user: action.payload
            }
        case actionTypes.LOGIN_REQUEST:
            return {
                ...state,
                loading: true,
                error: null,
            };
        case actionTypes.LOGIN_SUCCESS:
            return {
                ...state,
                user: action.payload,
                isLoggedIn: true,
                loading: false,
                error: null,
            };
```

- Tạo initialState để quản lý state.

```
export const initialState = {
  user: null,
  loading: false,
  error: null,

  isLoggedIn: false,

  room: null,
  group_room: [],
  listMessages: [],
  singleRoom: {},

  listFriends: [],
  listChatrooms: [],
  listRequests: [],
  listSends: [],

  searchFriend: {},

  noti: [],

};
```

- Sử dụng Hook: tạo folder hooks và tách ra thành các file hook truy cập giá trị trạng thái và dispatch từ context tương ứng.

```
const useAuth = () => {
  const { state, dispatch } = useAppContext();
  const { saveToLocalStorage } = useLocalStorage()
  const navigate = useNavigate()

  const handleLogin = async (email, password) => {
    if(state.user===null){
      dispatch(loginRequest());
      try {
        const response = await fetch(baseURL + '/user/sign-in', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({ email, password }),
        });

        if (response.ok) {
          const userData = await response.json();
          dispatch(loginSuccess(userData.data));

          saveToLocalStorage('accessToken', JSON.stringify(userData.data))

          navigate('/messages')

        } else {
          const errorData = await response.json();
          dispatch(loginFailure(errorData.message));
```

Tailwind CSS và HTML:

- Sử dụng Tailwind version 3.4.3.
- Thẻ đóng sẽ có độ thụt đầu dòng ngang với thẻ mở. Nếu một thẻ nằm trong một thẻ khác thì cần được thụt vào sâu hơn.
- Các thuộc tính trong một class name cần cách nhau một khoảng cách trắng và không có khoảng cách giữa dấu “ với kí tự đầu tiên, cùng như giữ ” với kí tự cuối cùng.

```
<div className="text-white">
  <div className="text-white">This is MyMusicPage</div>
  <div className="flex gap-2">
    <button className="bg-blue-300" onClick={() => navigate('/mymusic/song')}>
      Song
    </button>
    <button className="bg-blue-300" onClick={() => navigate('/mymusic/mv')}>
      MV
    </button>
    <button className="bg-blue-300" onClick={() => navigate('/mymusic/album')}>
      Album
    </button>
  </div>
  <Outlet />
</div>
```