MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Redux form
- ✓ Rest-based app react

NỘI DUNG

- Sử dụng project streams của Lab6 để thực hiện tiếp tục bài Lab7

Bài 1: Redux form

- Cài đặt redux-form
- Kết nối redux-form

client > src > reducers > JS index.js > ...

```
1  import { combineReducers } from 'redux';
2  import { reducer as formReducer } from 'redux-form';
3  import authReducer from './authReducer';
4
5  export default combineReducers({
6    auth: authReducer,
7    form: formReducer
8  });
```

- Tạo form và xử lý sự kiện

```
class StreamCreate extends React.Component {
  renderInput({ input, label }) {
    return (
      <div className="field">
        <label>{label}</label>
        <input {...input} />
      </div>
    );
  }

  onSubmit(formValues) {
    console.log(formValues);
  }

  render() {
    return (
      <form
        onSubmit={this.props.handleSubmit(this.onSubmit)}
        className="ui form"
      >
        <Field name="title" component={this.renderInput} label="Enter Title" />
        <Field
          name="description"
          component={this.renderInput}
          label="Enter Description"
        />
        <button className="ui button primary">Submit</button>
      </form>
    );
  }
}
```

- Form validate

client > src > components > streams > JS StreamCreate.js > ...

```
46
47    const validate = formValues => {
48      const errors = {};
49
50      if (!formValues.title) {
51        errors.title = 'You must enter a title';
52      }
53
54      if (!formValues.description) {
55        errors.description = 'You must enter a description';
56      }
57
58      return errors;
59    };
60
61    export default reduxForm({
62      form: 'streamCreate',
63      validate
64    })(StreamCreate);
```

- Show error

client > src > components > streams > JS StreamCreate.js > default

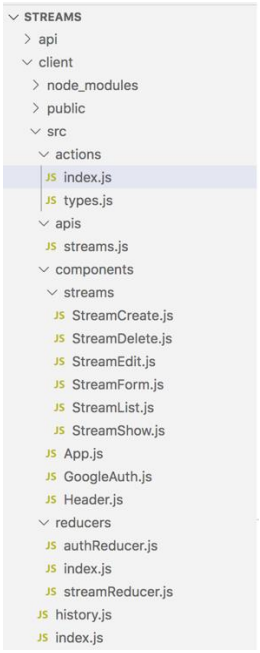
```
1     import React from 'react';
2     import { Field, reduxForm } from 'redux-form';
3
4     class StreamCreate extends React.Component {
5       renderError({ error, touched }) {
6         if (touched && error) {
7           return (
8             <div className="ui error message">
9               <div className="header">{error}</div>
10            </div>
11          );
12        }
13      }
14
15      renderInput = ({ input, label, meta }) => {
16        const className = `field ${meta.error && meta.touched ? 'error' : ''}`;
17        return (
18          <div className={className}>
19            <label>{label}</label>
20            <input {...input} autoComplete="off" />
21            {this.renderError(meta)}
22          </div>
23        );
24      };
```

Bài 2: Rest-based app react

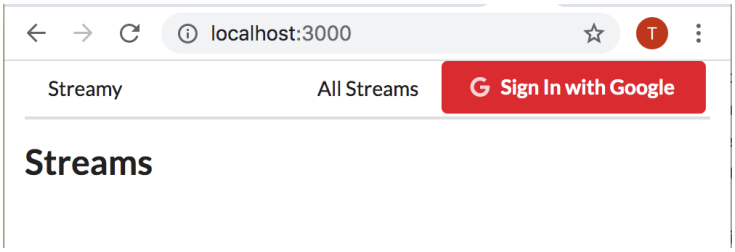
- Tạo api server trên streams project với cấu trúc như hình sau

Action	Method	Route	Response
List all records	GET	/streams	Array of records
Get one particular record	GET	/streams/:id	Single record
Create record	POST	/streams	Single record
Update a record	PUT	/streams/:id	Single record
Delete a record	DELETE	/streams/:id	Nothing

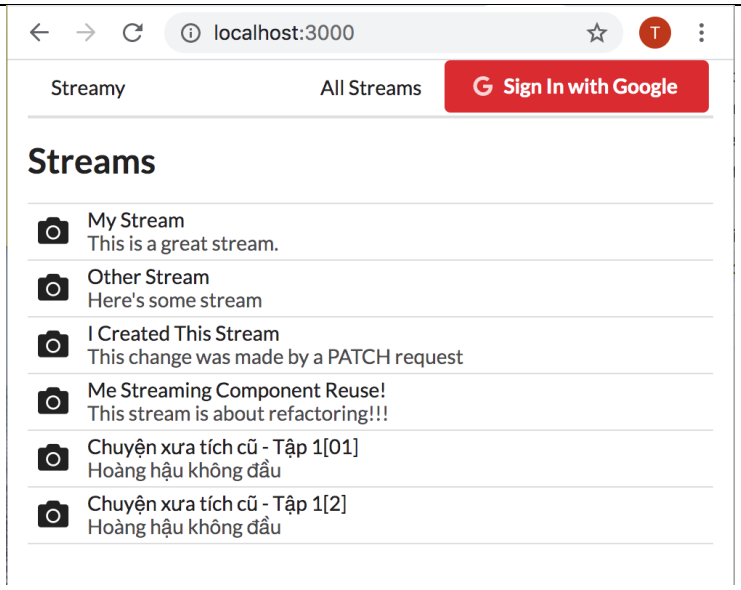
- Cấu trúc project



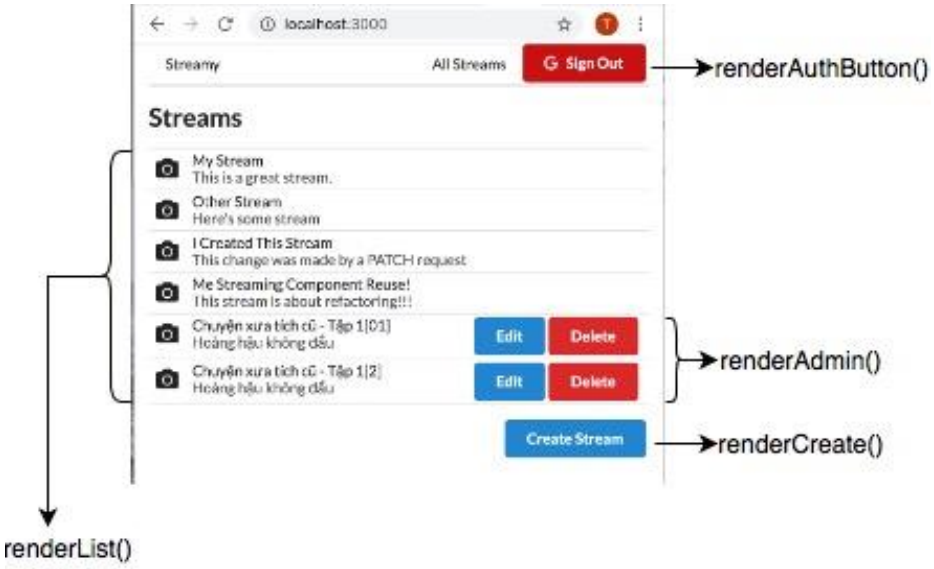
- Mô tả
  - API server chưa start



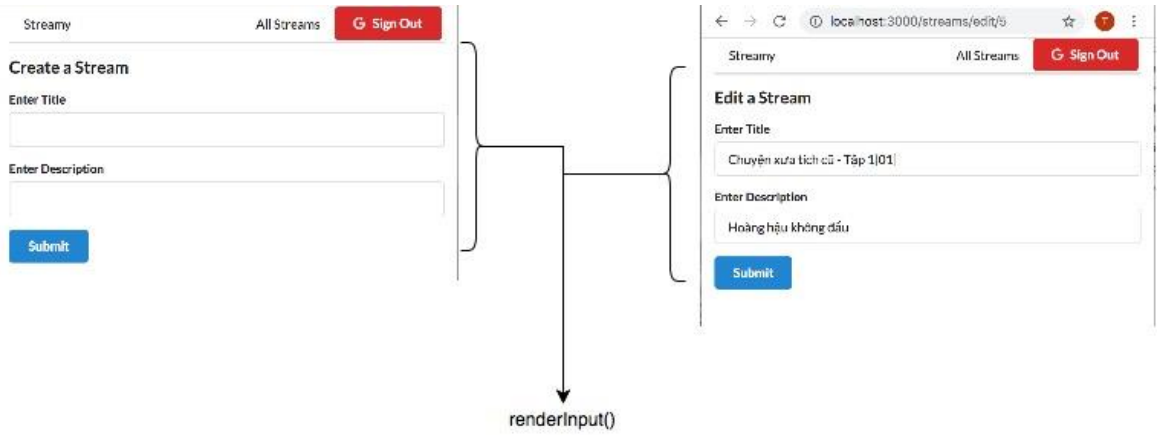
  - API server start



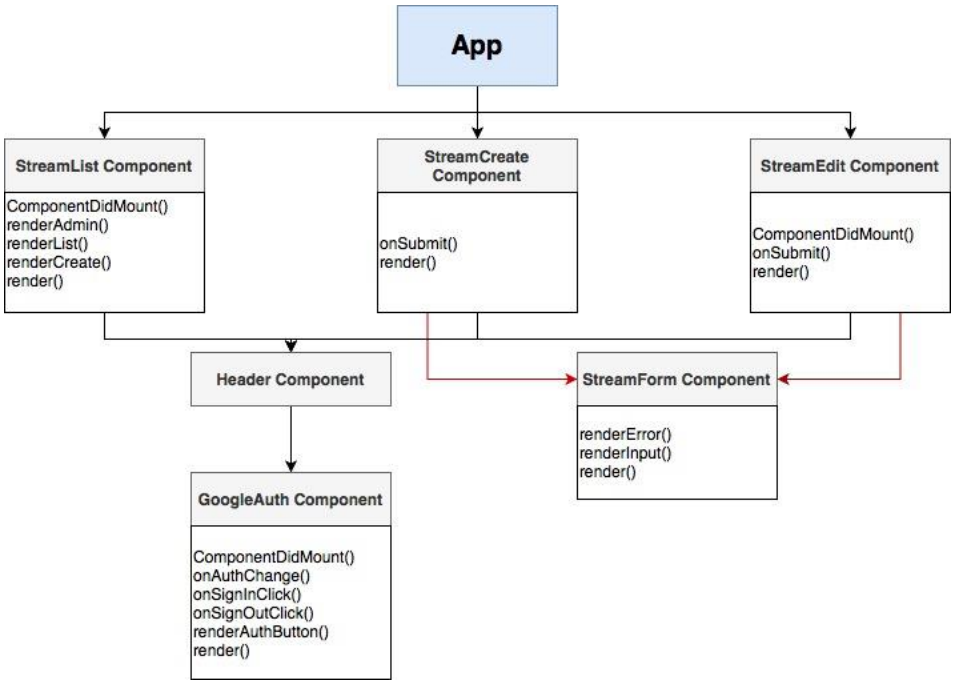
- StreamList



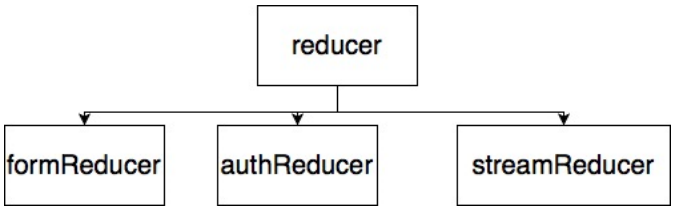
- Create & Edit



- Cấu trúc Components



- Cấu trúc reducer



client > src > reducers > JS authReducer.js > ...

```
1   import { SIGN_IN, SIGN_OUT } from '../actions/types';
2
3   const INTIAL_STATE = {
4     isSignedIn: null,
5     userId: null
6   };
7
8   export default (state = INTIAL_STATE, action) => {
9     switch (action.type) {
10      case SIGN_IN:
11        return { ...state, isSignedIn: true, userId: action.payload };
12      case SIGN_OUT:
13        return { ...state, isSignedIn: false, userId: null };
14      default:
15        return state;
16    }
17  };
```

client > src > reducers > JS streamReducer.js > ...

```
import _ from 'lodash';
import {
  FETCH_STREAM,
  FETCH_STREAMS,
  CREATE_STREAM,
  EDIT_STREAM,
  DELETE_STREAM
} from '../actions/types';

export default (state = {}, action) => {
  switch (action.type) {
    case FETCH_STREAMS:
      return { ...state, ..._.mapKeys(action.payload, 'id') };
    case FETCH_STREAM:
      return { ...state, [action.payload.id]: action.payload };
    case CREATE_STREAM:
      return { ...state, [action.payload.id]: action.payload };
    case EDIT_STREAM:
      return { ...state, [action.payload.id]: action.payload };
    case DELETE_STREAM:
      return _.omit(state, action.payload);
    default:
      return state;
  }
};
```

client > src > reducers > JS index.js > ...

```
import { combineReducers } from 'redux';
import { reducer as formReducer } from 'redux-form';
import authReducer from './authReducer';
import streamReducer from './streamReducer';

export default combineReducers({
  auth: authReducer,
  form: formReducer,
  streams: streamReducer
});
```

- Hướng dẫn
  - Apis

client > src > apis > JS streams.js > ...

```
import axios from 'axios';

export default axios.create({
  baseURL: 'http://localhost:3001'
});
```

- Action

client > src > actions > JS types.js > ...

```
export const SIGN_IN = 'SIGN_IN';
export const SIGN_OUT = 'SIGN_OUT';
export const CREATE_STREAM = 'CREATE_STREAM';
export const FETCH_STREAMS = 'FETCH_STREAMS';
export const FETCH_STREAM = 'FETCH_STREAM';
export const DELETE_STREAM = 'DELETE_STREAM';
export const EDIT_STREAM = 'EDIT_STREAM';
```

client > src > actions > JS index.js > ...

```
import streams from '../apis/streams';
import history from '../history';
import {
  SIGN_IN,
  SIGN_OUT,
  CREATE_STREAM,
  FETCH_STREAMS,
  FETCH_STREAM,
  DELETE_STREAM,
  EDIT_STREAM
} from './types';

export const signIn = userId => {
  return {
    type: SIGN_IN,
    payload: userId
  };
};

export const signOut = () => {
  return {
    type: SIGN_OUT
  };
};

export const createStream = formValues => async (dispatch, getState) => {
  const { userId } = getState().auth;
  const response = await streams.post('/streams', { ...formValues, userId });

  dispatch({ type: CREATE_STREAM, payload: response.data });
  history.push('/');
};

export const fetchStreams = () => async dispatch => {
  const response = await streams.get('/streams');

  dispatch({ type: FETCH_STREAMS, payload: response.data });
};

export const fetchStream = id => async dispatch => {
  const response = await streams.get(`/streams/${id}`);

  dispatch({ type: FETCH_STREAM, payload: response.data });
};

export const editStream = (id, formValues) => async dispatch => {
  const response = await streams.patch(`/streams/${id}`, formValues);

  dispatch({ type: EDIT_STREAM, payload: response.data });
  history.push('/');
};

export const deleteStream = id => async dispatch => {
  await streams.delete(`/streams/${id}`);

  dispatch({ type: DELETE_STREAM, payload: id });
};

```

- Component

client > src > components > JS GoogleAuth.js > ...

```
import React from 'react';…

class GoogleAuth extends React.Component {
  componentDidMount() {…
  }

  onAuthChange = isSignedIn => {…
  };

  onSignInClick = () => {…
  };

  onSignOutClick = () => {…
  };

  renderAuthButton() {
    if (this.props.isSignedIn === null) {
      return null;
    } else if (this.props.isSignedIn) {
      return (
        <button onClick={this.onSignOutClick} className="ui red google button">
          <i className="google icon" />
          Sign Out
        </button>
      );
    } else {
      return (
        <button onClick={this.onSignInClick} className="ui red google button">
          <i className="google icon" />
          Sign In with Google
        </button>
      );
    }
  }

  render() {
    return <div>{this.renderAuthButton()}</div>;
  }
}

const mapStateToProps = state => {
  return { isSignedIn: state.auth.isSignedIn };
};

export default connect(
  mapStateToProps,
  { signIn, signOut }
)(GoogleAuth);
```

client > src > components > JS Header.js > ...

```
import React from 'react';…

  const Header = () => {
    return (
      <div className="ui secondary pointing menu">
        <Link to="/" className="item">
          Streamy
        </Link>
        <div className="right menu">
          <Link to="/" className="item">
            All Streams
          </Link>
          <GoogleAuth />
        </div>
      </div>
    );
  };

  export default Header;
```

client > src > components > streams > JS StreamCreate.js > ...

```
1  > import React from 'react';…
5
6    class StreamCreate extends React.Component {
7      onSubmit = formValues => {
8        this.props.createStream(formValues);
9      };
10
11     render() {
12       return (
13         <div>
14           <h3>Create a Stream</h3>
15           <StreamForm onSubmit={this.onSubmit} />
16         </div>
17       );
18     }
19   }
20
21   export default connect(
22     null,
23     { createStream }
24   )(StreamCreate);
```

client > src > components > streams > JS StreamEdit.js > ...

```
1  > import _ from 'lodash';…
6
7    class StreamEdit extends React.Component {
8      componentDidMount() {
9        this.props.fetchStream(this.props.match.params.id);
10     }
11
12     onSubmit = formValues => {
13       this.props.editStream(this.props.match.params.id, formValues);
14     };
15
16     render() {
17       if (!this.props.stream) {
18         return <div>Loading...</div>;
19       }
20
21       return (
22         <div>
23           <h3>Edit a Stream</h3>
24           <StreamForm
25             initialValues={_.pick(this.props.stream, 'title', 'description')}
26             onSubmit={this.onSubmit}
27           />
28         </div>
29       );
30     }
31   }
32
33   const mapStateToProps = (state, ownProps) => {
34     return { stream: state.streams[ownProps.match.params.id] };
35   };
36
37   export default connect(
38     mapStateToProps,
39     { fetchStream, editStream }
40   )(StreamEdit);
```

client > src > components > streams > JS StreamForm.js > ...

```
1   import React from 'react';
2   import { Field, reduxForm } from 'redux-form';
3
4   class StreamForm extends React.Component {
5     renderError({ error, touched }) {
6       if (touched && error) {
7         return (
8           <div className="ui error message">
9             <div className="header">{error}</div>
10          </div>
11        );
12      }
13    }
14
15    renderInput = ({ input, label, meta }) => {
16      const className = `field ${meta.error && meta.touched ? 'error' : ''}`;
17      return (
18        <div className={className}>
19          <label>{label}</label>
20          <input {...input} autoComplete="off" />
21          {this.renderError(meta)}
22        </div>
23      );
24    };
25
26    onSubmit = formValues => {
27      this.props.onSubmit(formValues);
28    };
29
30    render() {…
45    }
46  }
47
48  const validate = formValues => {…
60  };
61
62  export default reduxForm({
63    form: 'streamForm',
64    validate
65  })(StreamForm);
66
```

client > src > components > streams > JS StreamList.js > StreamList

```
1  > import React from 'react';…
5
6  class StreamList extends React.Component {
7    componentDidMount() {
8      this.props.fetchStreams();
9    }
10
11   renderAdmin(stream) {
12     if (stream.userId === this.props.currentUserId) {
13       return (
14         <div className="right floated content">
15           <Link to={`/streams/edit/${stream.id}`} className="ui button primary">
16             Edit
17           </Link>
18           <button className="ui button negative">Delete</button>
19         </div>
20       );
21     }
22   }
23
24   renderList() {
25     return this.props.streams.map(stream => {
26       return (
27         <div className="item" key={stream.id}>
28           {this.renderAdmin(stream)}
29           <i className="large middle aligned icon camera" />
30           <div className="content">
31             {stream.title}
32             <div className="description">{stream.description}</div>
33           </div>
34         </div>
35       );
36     });
37   }
38
39   renderCreate() {
40     if (this.props.isSignedIn) {
41       return (
42         <div style={{ textAlign: 'right' }}>
43           <Link to="/streams/new" className="ui button primary">
44             Create Stream
45           </Link>
46         </div>
47       );
48     }
49   }
50
51   render() {
52     return (
53       <div>
54         <h2>Streams</h2>
55         <div className="ui celled list">{this.renderList()}</div>
56         {this.renderCreate()}
57       </div>
58     );
59   }
60 }
61
62 const mapStateToProps = state => {
63   return {
64     streams: Object.values(state.streams),
65     currentUserId: state.auth.userId,
66     isSignedIn: state.auth.isSignedIn
67   };
68 };
69
70 export default connect(
71   mapStateToProps,
72   { fetchStreams }
73 )(StreamList);
74
```

Bài 3: Browser history

client > src > components > JS App.js > ...

```
1  > import React from 'react';…
10
11   const App = () => {
12     return (
13       <div className="ui container">
14         <Router history={history}>
15   >       <div>…
23         </Router>
24       </div>
25     );
26   };
27
28   export default App;
```

client > src > JS history.js > ...

```
1   import { createBrowserHistory } from 'history';
2   export default createBrowserHistory();
3
```

*** Yêu cầu nộp bài:

SV nén file (*hoặc share thư mục google drive*) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---