**Frontend – React / JS**

**public/**

**server/**

**src/**

**src/assets/**

**src/components/**

***src/components/footer/***

**Footer.js/**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import \* as actions from "../../store/actions";

import Navigator from '../../components/Navigator';

import './Footer.scss';

class Footer extends Component {

   render() {

      const { processLogout } = this.props;

      return (

         <div className="footer-container">

            Hello, this is the footer section.

         </div>

      );

   }

}

const mapStateToProps = state => {

   return {

      isLoggedIn: state.admin.isLoggedIn

   };

};

const mapDispatchToProps = dispatch => {

   return {

      processLogout: () => dispatch(actions.processLogout()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Footer);

**Footer.scss**

***src/components/formating/***

**FormattedDate.js/**

import React, { Component } from 'react';

import moment from 'moment';

/\*\* For valid format please see <a href="https://momentjs.com/docs/#/displaying/">Moment format options</a> \*/

const dateFormat = 'DD/MM/YYYY';

class FormattedDate extends Component {

    render() {

        const { format, value, ...otherProps } = this.props;

        var dFormat = format ? format : dateFormat;

        const formattedValue = value ? moment.utc(value).format(dFormat) : null;

        return (

            <span {...otherProps}>{formattedValue}</span>

        );

    }

}

export default FormattedDate;

**index.js**

export { default as FormattedDate } from './FormattedDate';

***src/components/feader/***

**Header.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import \* as actions from "../../store/actions";

import Navigator from '../../components/Navigator';

import { adminMenu } from './menuApp';

import './Header.scss';

class Header extends Component {

    render() {

        const { processLogout } = this.props;

        return (

            <div className="header-container">

                {/\* thanh navigator \*/}

                <div className="header-tabs-container">

                    <Navigator menus={adminMenu} />

                </div>

                {/\* nút logout \*/}

                <div className="btn btn-logout" onClick={processLogout}>

                    <i className="fas fa-sign-out-alt"></i>

                </div>

            </div>

        );

    }

}

const mapStateToProps = state => {

    return {

        isLoggedIn: state.admin.isLoggedIn

    };

};

const mapDispatchToProps = dispatch => {

    return {

        processLogout: () => dispatch(actions.processLogout()),

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(Header);

**Header.scss**

@import "../../styles/common.scss";

.header-container {

    z-index: 99;

    display: flex;

    justify-content: space-between;

    background-color: $colormain;

    color: $common-text-color;

    height: 40px;

    position: relative;

    .btn-logout {

        color: $common-text-color;

        line-height: 40px;

        height: 40px;

        padding: 0 10px;

        &:hover {

            background-color: darken($colormain, 5);

        }

        i {

            font-size: $base-size + 1px;

        }

    }

}

**menuApp.js**

export const adminMenu = [

    { //hệ thống

        name: 'menu.system.header', menus: [

            {

                name: 'menu.system.system-administrator.header',

                subMenus: [

                    { name: 'menu.system.system-administrator.user-manage', link: '/system/user-manage' },

                    { name: 'menu.system.system-administrator.product-manage', link: '/system/product-manage' },

                    { name: 'menu.system.system-administrator.register-package-group-or-account', link: '/system/register-package-group-or-account' },

                ]

            },

            // { name: 'menu.system.system-parameter.header', link: '/system/system-parameter' },

        ]

    },

];

***src/components/input/***

**DatePicker.js**

import React, { Component } from 'react';

import Flatpickr from 'react-flatpickr';

import moment from 'moment';

import { KeyCodeUtils } from "../../utils";

import './DatePicker.scss';

// const CustomInput = ({ value, defaultValue, inputRef, onInputChange, onInputBlur, ...props }) => {

//     return <input {...props} className='custom-form-control custom-date-input' defaultValue={defaultValue} ref={inputRef}

//         onChange={onInputChange}

//         onBlur={onInputBlur}

//     />;

// };

class DatePicker extends Component {

    flatpickrNode = null;

    nodeRef = element => {

        this.flatpickr = element && element.flatpickr;

        this.flatpickrNode = element && element.node;

        if (this.flatpickrNode) {

            this.flatpickrNode.addEventListener('blur', this.handleBlur);

            this.flatpickrNode.addEventListener('keydown', this.handlerKeyDown);

        }

    };

    handlerKeyDown = (event) => {

        const keyCode = event.which || event.keyCode;

        if (keyCode === KeyCodeUtils.ENTER) {

            event.preventDefault();

            const { onChange } = this.props;

            const value = event.target.value;

            // Take the blur event and process the string value

            const valueMoment = moment(value, 'DD/MM/YYYY');

            onChange([valueMoment.toDate(), valueMoment.toDate()]);

        }

    }

    componentWillUnmount() {

        if (this.flatpickrNode) {

            this.flatpickrNode.removeEventListener('blur', this.handleBlur);

            this.flatpickrNode.removeEventListener('keydown', this.handlerKeyDown);

        }

    }

    handleBlur = (event) => {

        const { onChange } = this.props;

        const value = event.target.value;

        // Take the blur event and process the string value

        event.preventDefault();

        const valueMoment = moment(value, 'DD/MM/YYYY');

        onChange([valueMoment.toDate(), valueMoment.toDate()]);

    };

    onOpen = () => {

        if (this.flatpickrNode) {

            this.flatpickrNode.blur();

        }

    }

    close() {

        this.flatpickr.close();

    }

    checkDateValue = (str, max) => {

        if (str.charAt(0) !== '0' || str === '00') {

            var num = parseInt(str);

            if (isNaN(num) || num <= 0 || num > max) num = 1;

            str = num > parseInt(max.toString().charAt(0)) && num.toString().length === 1 ? '0' + num : num.toString();

        };

        return str;

    }

    // autoFormatonBlur = (value) => {

    //     var input = value;

    //     var values = input.split('/').map(function (v, i) {

    //         return v.replace(/\D/g, '')

    //     });

    //     var output = '';

    //     if (values.length == 3) {

    //         var year = values[2].length !== 4 ? parseInt(values[2]) + 2000 : parseInt(values[2]);

    //         var month = parseInt(values[0]) - 1;

    //         var day = parseInt(values[1]);

    //         var d = new Date(year, month, day);

    //         if (!isNaN(d)) {

    //             //document.getElementById('result').innerText = d.toString();

    //             var dates = [d.getMonth() + 1, d.getDate(), d.getFullYear()];

    //             output = dates.map(function (v) {

    //                 v = v.toString();

    //                 return v.length == 1 ? '0' + v : v;

    //             }).join(' / ');

    //         };

    //     };

    //     // this.value = output;

    //     return output;

    // }

    autoFormatOnChange = (value, seperator) => {

        var input = value;

        let regexForDeleting = new RegExp(`\\D\\${seperator}$`);

        //if (/\D\/$/.test(input)) input = input.substr(0, input.length - 3); // dat.nt: Xóa thêm 1 ký tự nếu xóa dấu cách sau / (VD: 12 / 12 /=> 12 / 1)

        if (regexForDeleting.test(input)) input = input.substr(0, input.length - 3);

        var values = input.split(seperator).map(function (v) {

            return v.replace(/\D/g, '')

        });

        if (values[0]) values[0] = this.checkDateValue(values[0], 31);

        if (values[1]) values[1] = this.checkDateValue(values[1], 12);

        var output = values.map(function (v, i) {

            return v.length === 2 && i < 2 ? v + ' ' + seperator + ' ' : v;

        });

        return output.join('').substr(0, 14);

    }

    onInputChange = (e) => {

        if (this.DISPLAY\_FORMAT === this.DATE\_FORMAT\_AUTO\_FILL) {

            let converted = this.autoFormatOnChange(e.target.value, this.SEPERATOR);

            e.target.value = converted;

        }

    }

    onInputBlur = (e) => {

    }

    //dat.nt : Auto Fill cho dạng ngăn cách và format cụ thể (seperator có thể dc thay thế)

    SEPERATOR = "/";

    DATE\_FORMAT\_AUTO\_FILL = "d/m/Y"; // Format không thay đổi

    // dat.nt : Format ngày hiển thị

    DISPLAY\_FORMAT = "d/m/Y";

    render() {

        const { value, onChange, minDate, onClose, ...otherProps } = this.props;

        const options = {

            dateFormat: this.DISPLAY\_FORMAT,

            allowInput: true,

            disableMobile: true,

            onClose: onClose,

            onOpen: this.onOpen

        };

        if (minDate) {

            options.minDate = minDate;

        }

        return (

            <Flatpickr

                ref={this.nodeRef}

                value={value}

                onChange={onChange}

                options={options}

                // render={

                //     ({ defaultValue, value, ...props }, ref) => {

                //         return <CustomInput defaultValue={defaultValue} inputRef={ref} onInputChange={this.onInputChange} onInputBlur={this.onInputBlur} />

                //     }

                // }

                {...otherProps}

            />

        );

    }

}

export default DatePicker;

**DatePicker.scss**

@import "../../styles/common";

$bezier: cubic-bezier(0.23, 1, 0.32, 1);

$slideTime: 400ms;

$daySize: 39px;

$padding: $daySize / 16;

$dayMargin: 2px;

$daysWidth: $daySize \* 7 + $dayMargin \* 14 + $padding \* 2 + 2;

$calendarWidth: $daysWidth;

$monthNavHeight: 28px !default;

$weekdaysHeight: 28px !default;

$timeHeight: 40px;

$disabledBorderColor: transparent;

$selectedDayForeground: #fff;

$selectedDayBackground: $colormain;

input {

  &.custom-date-input {

    word-spacing:-3px

  }

}

@-webkit-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-moz-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-ms-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@-o-keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

@keyframes fpFadeInDown {

  from {

    opacity: 0;

    transform: translate3d(0, -20px, 0);

  }

  to {

    opacity: 1;

    transform: translate3d(0, 0, 0);

  }

}

.flatpickr-calendar {

  background: transparent;

  opacity: 0;

  display: none;

  text-align: center;

  visibility: hidden;

  padding: 0;

  animation: none;

  direction: ltr;

  border: 0;

  font-size: 14px;

  line-height: 24px;

  border-radius: 5px;

  position: absolute;

  width: $calendarWidth;

  box-sizing: border-box;

  touch-action: manipulation;

  &.open, &.inline {

    opacity: 1;

    max-height: 640px;

    visibility: visible;

  }

  &.open {

    display: inline-block;

    z-index: 99999;

  }

  &.animate.open {

    animation: fpFadeInDown 300ms $bezier;

  }

  &.inline {

    display: block;

    position: relative;

    top: 2px;

  }

  &.static {

    position: absolute;

    top: calc(100% + 2px);

    &.open {

      z-index: 999;

      display: block;

    }

  }

  &.multiMonth {

    .flatpickr-days .dayContainer:nth-child(n+1) {

      & .flatpickr-day.inRange:nth-child(7n+7) {

        box-shadow: none !important;

      }

    }

    .flatpickr-days .dayContainer:nth-child(n+2) {

      & .flatpickr-day.inRange:nth-child(7n+1) {

        box-shadow: -2px 0 0 #e6e6e6, 5px 0 0 #e6e6e6;

      }

    }

  }

  .hasWeeks, .hasTime {

    .dayContainer {

      border-bottom: 0;

      border-bottom-right-radius: 0;

      border-bottom-left-radius: 0;

    }

  }

  &.showTimeInput.hasTime {

    .flatpickr-time {

      height: $timeHeight;

    }

  }

  &.noCalendar.hasTime {

    .flatpickr-time {

      height: auto;

    }

  }

  &:before, &:after {

    position: absolute;

    display: block;

    pointer-events: none;

    border: solid transparent;

    content: '';

    height: 0;

    width: 0;

    left: 22px;

  }

  &.rightMost {

    &:before, &:after {

      left: auto;

      right: 22px;

    }

  }

  &:before {

    border-width: 5px;

    margin: 0 -5px;

  }

  &:after {

    border-width: 4px;

    margin: 0 -4px;

  }

  &.arrowTop {

    &:before, &:after {

      bottom: 100%;

    }

  }

  &.arrowBottom {

    &:before, &:after {

      top: 100%;

    }

  }

  &:focus {

    outline: 0;

  }

  .flatpickr-months {

    display: flex;

    .flatpickr-month {

      height: $monthNavHeight;

      line-height: 1;

      text-align: center;

      position: relative;

      user-select: none;

      overflow: hidden;

      flex: 1;

    }

    .flatpickr-prev-month, .flatpickr-next-month {

      text-decoration: none;

      cursor: pointer;

      position: absolute;

      top: 0;

      line-height: 16px;

      height: $monthNavHeight;

      padding: 10px;

      z-index: 3;

      &.disabled {

        display: none;

      }

      i {

        position: relative;

      }

      &.flatpickr-prev-month {

        left: 0;

      }

      &.flatpickr-next-month {

        right: 0;

      }

      svg {

        width: 14px;

        height: 14px;

        path {

          transition: fill 0.1s;

        }

      }

    }

  }

  background: $date-picker-bg;

  box-shadow: 1px 0 0 $date-picker-border, -1px 0 0 $date-picker-border, 0 1px 0 $date-picker-border, 0 -1px 0 $date-picker-border, 0 3px 13px rgba(black, 0.08);

  &.arrowTop {

    &:before {

      border-bottom-color: $date-picker-border;

    }

    &:after {

      border-bottom-color: $date-picker-bg;

    }

  }

  &.arrowBottom {

    &:before {

      border-top-color: $date-picker-border;

    }

    &:after {

      border-top-color: $date-picker-bg;

    }

  }

  &.showTimeInput.hasTime {

    .flatpickr-time {

      border-top: 1px solid $date-picker-border;

    }

  }

  .flatpickr-months {

    .flatpickr-month {

      background: $date-picker-month-bg;

      color: $date-picker-month-fg;

      fill: $date-picker-month-fg;

    }

    .flatpickr-prev-month, .flatpickr-next-month {

      color: $date-picker-month-fg;

      fill: $date-picker-month-fg;

      &:hover {

        color: $date-picker-today-bg;

        svg {

          fill: $date-picker-arrow-hover;

        }

      }

    }

  }

}

.flatpickr-wrapper {

  position: relative;

  display: inline-block;

}

.numInputWrapper {

  position: relative;

  height: auto;

  input, span {

    display: inline-block;

  }

  input {

    width: 100%;

    &::-ms-clear {

      display: none;

    }

    &::-webkit-outer-spin-button, &::-webkit-inner-spin-button {

      margin: 0;

      -webkit-appearance: none;

    }

  }

  span {

    position: absolute;

    right: 0;

    width: 14px;

    padding: 0 4px 0 2px;

    height: 50%;

    line-height: 50%;

    opacity: 0;

    cursor: pointer;

    box-sizing: border-box;

    &:after {

      display: block;

      content: "";

      position: absolute;

    }

    &.arrowUp {

      top: 0;

      border-bottom: 0;

      &:after {

        top: 26%;

      }

    }

    &.arrowDown {

      top: 50%;

      &:after {

        top: 40%;

      }

    }

    svg {

      width: inherit;

      height: auto;

    }

  }

  &:hover {

    span {

      opacity: 1;

    }

  }

    span {

      border: 1px solid rgba($date-picker-day-fg, 0.15);

      &:hover {

        background: rgba(invert($date-picker-bg), 0.1);

      }

      &:active {

        background: rgba(invert($date-picker-bg), 0.2);

      }

      &.arrowUp {

        &:after {

          border-left: 4px solid transparent;

          border-right: 4px solid transparent;

          border-bottom: 4px solid rgba($date-picker-day-fg, 0.6);

        }

      }

      &.arrowDown {

        &:after {

          border-left: 4px solid transparent;

          border-right: 4px solid transparent;

          border-top: 4px solid rgba($date-picker-day-fg, 0.6);

        }

      }

      svg {

        path {

          fill: rgba($date-picker-month-fg, 0.5);

        }

      }

    }

    &:hover {

      background: rgba(invert($date-picker-bg), 0.05);

    }

}

.flatpickr-current-month {

  font-size: 135%;

  font-weight: 300;

  color: inherit;

  position: absolute;

  width: 75%;

  left: 12.5%;

  padding: 0.22 \* $monthNavHeight 0 0 0;

  line-height: 1;

  height: $monthNavHeight;

  display: inline-block;

  text-align: center;

  transform: translate3d(0px, 0px, 0px);

  span.cur-month {

    font-family: inherit;

    font-weight: 700;

    color: inherit;

    display: inline-block;

    margin-left: 0.5ch;

    padding: 0;

  }

  .numInputWrapper {

    width: 6ch;

    display: inline-block;

  }

  input.cur-year {

    background: transparent;

    box-sizing: border-box;

    color: inherit;

    cursor: text;

    padding: 0 0 0 0.5ch;

    margin: 0;

    display: inline-block;

    font-size: inherit;

    font-family: inherit;

    font-weight: 300;

    line-height: inherit;

    height: auto;

    border: 0;

    border-radius: 0;

    vertical-align: initial;

    -webkit-appearance: textfield;

    -moz-appearance: textfield;

    appearance: textfield;

    &:focus {

      outline: 0;

    }

    &[disabled], &[disabled]:hover {

      font-size: 100%;

      pointer-events: none;

    }

  }

    span.cur-month {

      &:hover {

        background: rgba(invert($date-picker-bg), 0.05);

      }

    }

    .numInputWrapper {

      span.arrowUp:after {

        border-bottom-color: $date-picker-month-fg;

      }

      span.arrowDown:after {

        border-top-color: $date-picker-month-fg;

      }

    }

    input.cur-year {

      &[disabled], &[disabled]:hover {

        color: rgba($date-picker-month-fg, 0.5);

        background: transparent;

      }

    }

}

.flatpickr-monthDropdown-months {

  border: none;

  border-radius: 0;

  box-sizing: border-box;

  color: inherit;

  cursor: pointer;

  font-size: inherit;

  font-family: inherit;

  font-weight: 300;

  height: $monthNavHeight - 6px;

  line-height: inherit;

  margin: -1px 0 0 0;

  outline: none;

  padding: 0 0 0 0.5ch;

  position: relative;

  vertical-align: initial;

  -webkit-box-sizing: border-box;

  -webkit-appearance: none;

  -moz-appearance: none;

  appearance: none;

  width: auto;

  &:focus, &:active {

    outline: none;

  }

  .flatpickr-monthDropdown-month {

    outline: none;

    padding: 0;

  }

    background: $date-picker-month-bg;

    &:hover {

      background: rgba(invert($date-picker-bg), 0.05);

    }

    .flatpickr-monthDropdown-month {

      background-color: $date-picker-month-bg;

    }

}

.flatpickr-weekdays {

  text-align: center;

  overflow: hidden;

  width: 100%;

  display: flex;

  align-items: center;

  height: $weekdaysHeight;

  .flatpickr-weekdaycontainer {

    display: flex;

    flex: 1;

  }

  background: $date-picker-weekdays-bg;

}

span.flatpickr-weekday {

  cursor: default;

  font-size: 90%;

  line-height: 1;

  margin: 0;

  text-align: center;

  display: block;

  flex: 1;

  font-weight: bolder;

    background: $date-picker-month-bg;

    color: $date-picker-weekdays-fg;

}

.dayContainer, .flatpickr-weeks {

  padding: 1px 0 0 0;

}

.flatpickr-days {

  position: relative;

  overflow: hidden;

  display: flex;

  align-items: flex-start;

  width: $daysWidth;

  &:focus {

    outline: 0;

  }

}

.dayContainer {

  padding: 0;

  outline: 0;

  text-align: left;

  width: $daysWidth;

  min-width: $daysWidth;

  max-width: $daysWidth;

  box-sizing: border-box;

  display: inline-block;

  display: -ms-flexbox;

  display: flex;

  flex-wrap: wrap;

  -ms-flex-wrap: wrap;

  -ms-flex-pack: justify;

  justify-content: space-around;

  transform: translate3d(0px, 0px, 0px);

  opacity: 1;

    & + .dayContainer {

      box-shadow: -1px 0 0 $date-picker-border;

    }

}

.flatpickr-day {

  background: none;

  border: 1px solid transparent;

  border-radius: 150px;

  box-sizing: border-box;

  cursor: pointer;

  font-weight: 400;

  width: 14.2857143%;

  flex-basis: 14.2857143%;

  max-width: $daySize;

  height: $daySize;

  line-height: $daySize;

  margin: 0;

  display: inline-block;

  position: relative;

  justify-content: center;

  text-align: center;

  &, &.prevMonthDay, &.nextMonthDay {

    &.inRange, &.today.inRange, &:hover, &:focus {

      cursor: pointer;

      outline: 0;

    }

  }

  &.selected, &.startRange, &.endRange {

    &.startRange {

      border-radius: 50px 0 0 50px;

    }

    &.endRange {

      border-radius: 0 50px 50px 0;

    }

    &.startRange.endRange {

      border-radius: 50px;

    }

  }

  &.inRange {

    border-radius: 0;

  }

  &.flatpickr-disabled, &.flatpickr-disabled:hover,

  &.prevMonthDay, &.nextMonthDay,

  &.notAllowed, &.notAllowed.prevMonthDay, &.notAllowed.nextMonthDay {

    cursor: default;

  }

  &.flatpickr-disabled, &.flatpickr-disabled:hover {

    cursor: not-allowed;

  }

  &.week.selected {

    border-radius: 0;

  }

  &.hidden {

    visibility: hidden;

  }

    background: none;

    border: 1px solid transparent;

    color: $date-picker-day-fg;

    &, &.prevMonthDay, &.nextMonthDay {

      &.inRange, &.today.inRange, &:hover, &:focus {

        background: $date-picker-day-hover-bg;

        border-color: $date-picker-day-hover-bg;

      }

    }

    &.today {

      border-color: $date-picker-today-border;

      &:hover, &:focus {

        border-color: $date-picker-today-border;

        background: $date-picker-today-bg;

        color: $date-picker-today-fg;

      }

    }

    &.selected, &.startRange, &.endRange {

      &, &.inRange, &:focus, &:hover, &.prevMonthDay, &.nextMonthDay {

        background: $selectedDayBackground;

        box-shadow: none;

        color: $selectedDayForeground;

        border-color: $selectedDayBackground;

      }

      &.startRange + .endRange:not(:nth-child(7n+1)) {

        box-shadow: -5 \* $dayMargin 0 0 $selectedDayBackground;

      }

    }

    &.inRange {

      border-radius: 0;

      box-shadow: -2.5 \* $dayMargin 0 0 $date-picker-day-hover-bg, 2.5 \* $dayMargin 0 0 $date-picker-day-hover-bg;

    }

    &.flatpickr-disabled, &.flatpickr-disabled:hover,

    &.prevMonthDay, &.nextMonthDay,

    &.notAllowed, &.notAllowed.prevMonthDay, &.notAllowed.nextMonthDay {

      color: rgba($date-picker-day-fg, 0.3);

      background: transparent;

      border-color: $disabledBorderColor;

    }

    &.flatpickr-disabled, &.flatpickr-disabled:hover {

      color: rgba($date-picker-day-fg, 0.1);

    }

}

.rangeMode .flatpickr-day {

  margin-top: 1px;

}

.flatpickr-weekwrapper {

  float: left;

  .flatpickr-weeks {

    padding: 0 12px;

  }

  .flatpickr-weekday {

    float: none;

    width: 100%;

    line-height: $weekdaysHeight;

  }

  span.flatpickr-day {

    &, &:hover {

      display: block;

      width: 100%;

      max-width: none;

      cursor: default;

    }

  }

    .flatpickr-weeks {

      box-shadow: 1px 0 0 $date-picker-border;

    }

    span.flatpickr-day {

      &, &:hover {

        color: rgba($date-picker-day-fg, 0.3);

        background: transparent;

        border: none;

      }

    }

}

.flatpickr-innerContainer {

  display: flex;

  box-sizing: border-box;

  overflow: hidden;

}

.flatpickr-rContainer {

  display: inline-block;

  padding: 0;

  box-sizing: border-box;

}

.flatpickr-time {

  text-align: center;

  outline: 0;

  height: 0;

  line-height: $timeHeight;

  max-height: $timeHeight;

  box-sizing: border-box;

  overflow: hidden;

  display: flex;

  &:after {

    content: "";

    display: table;

    clear: both;

  }

  .numInputWrapper {

    flex: 1;

    width: 40%;

    height: $timeHeight;

    float: left;

  }

  &.hasSeconds .numInputWrapper {

    width: 26%;

  }

  &.time24hr .numInputWrapper {

    width: 49%;

  }

  input {

    background: transparent;

    box-shadow: none;

    border: 0;

    border-radius: 0;

    text-align: center;

    margin: 0;

    padding: 0;

    height: inherit;

    line-height: inherit;

    font-size: 14px;

    position: relative;

    box-sizing: border-box;

    -webkit-appearance: textfield;

    -moz-appearance: textfield;

    appearance: textfield;

    &.flatpickr-hour {

      font-weight: bold;

    }

    &.flatpickr-minute, &.flatpickr-second {

      font-weight: 400;

    }

    &:focus {

      outline: 0;

      border: 0;

    }

  }

  .flatpickr-time-separator, .flatpickr-am-pm {

    height: inherit;

    float: left;

    line-height: inherit;

    font-weight: bold;

    width: 2%;

    user-select: none;

    align-self: center;

  }

  .flatpickr-am-pm {

    outline: 0;

    width: 18%;

    cursor: pointer;

    text-align: center;

    font-weight: 400;

  }

    .numInputWrapper {

      span.arrowUp:after {

        border-bottom-color: $date-picker-day-fg;

      }

      span.arrowDown:after {

        border-top-color: $date-picker-day-fg;

      }

    }

    input {

      color: $date-picker-day-fg;

    }

    .flatpickr-time-separator, .flatpickr-am-pm {

      color: $date-picker-day-fg;

    }

    input, .flatpickr-am-pm {

      &:hover, &:focus {

        background: lighten($date-picker-day-hover-bg, 3);

      }

    }

}

.flatpickr-input[readonly] {

  cursor: pointer;

}

**Index.js**

export { default as DatePicker } from './DatePicker';

export { default as InputSuggest } from './InputSuggest';

**InputSuggest.js**

import React from 'react';

import Autosuggest from 'react-autosuggest';

import { connect } from 'react-redux';

import \_ from 'lodash';

import "./InputSuggest.scss";

const isAlphaNumericChar = keycode => {

    return (keycode >= 48 && keycode <= 57) || (keycode >= 65 && keycode <= 90);

};

class InputSuggestion extends React.Component {

    constructor() {

        super();

        this.state = {

            textInput: '',

            suggestions: []

        };

        document.addEventListener('keydown', this.freeTyping.bind(this), false);

    }

    freeTyping(e) {

        if (!isAlphaNumericChar(e.keyCode)) {

            return;

        }

        if (e.target.value === undefined) {

            this.inputSearch.focus();

        }

    }

    getSuggestions = textInput => {

        // if (!textInput || textInput.length === 0) return []; //chưa nhập -> chưa gợi ý

        const inputValue = textInput.trim().toLowerCase();

        if (!this.props.inputsWithIndex) {

            return [{ textInput: textInput }];

        };

        let inputsWithIndex = this.props.inputsWithIndex;

        let keyArr = Object.keys(inputsWithIndex).filter(

            textInput => {

                return textInput.toLowerCase().indexOf(inputValue) >= 0

            }

        );

        var suggestArr = keyArr.map(function (key) {

            return inputsWithIndex[key];

        });

        return suggestArr;

    };

    storeInputReference = autosuggest => {

        if (autosuggest !== null) {

            this.inputSearch = autosuggest.input;

        }

    };

    shouldRenderSuggestions = value => {

        return true;

    };

    getSuggestionValue = suggestion => {

        this.props.onSelected(suggestion);

        return suggestion.displayName;

    }

    renderSuggestion = suggestion => {

        return (

            <div className="suggest-item">

                {suggestion.displayName}

            </div>

        );

    };

    onSuggestionsFetchRequested = ({ value }) => {

        this.setState({

            suggestions: this.sortSuggestions(this.getSuggestions(value), value)

        });

    };

    sortSuggestions(suggestions, value) {

        var results =  \_.sortBy(suggestions, (element) => {

            return element.displayName

        })

        return results;

    }

    onSuggestionsClearRequested = () => {

        this.setState({

            suggestions: []

        });

    };

    onSuggestionSelected = (event, selected) => {

        this.props.onSelected(selected.suggestion)

        this.setState({

            textInput: selected.suggestion && selected.suggestion.displayName

        });

    };

    handleChangeInput = (event, { newValue }) => {

        this.setState({

            textInput: newValue || ''

        });

    };

    reset() {

        this.setState({

            textInput: ''

        });

        this.onSuggestionsFetchRequested({ value: '' });

    }

    handleBlurInput() {

        // this.setState({

        //     textInput: ''

        // });

    }

    render() {

        const { textInput, suggestions } = this.state;

        const inputProps = {

            value: textInput,

            className: "custom-form-control",

            onChange: this.handleChangeInput,

            onClick: () => {

                this.reset();

            },

            onBlur: () => {

                this.handleBlurInput();

            }

        };

        return (

            <Autosuggest

                suggestions={suggestions}

                onSuggestionsFetchRequested={this.onSuggestionsFetchRequested}

                onSuggestionsClearRequested={this.onSuggestionsClearRequested}

                getSuggestionValue={this.getSuggestionValue}

                renderSuggestion={this.renderSuggestion}

                onSuggestionSelected={this.onSuggestionSelected}

                shouldRenderSuggestions={this.shouldRenderSuggestions}

                highlightFirstSuggestion={true}

                inputProps={inputProps}

                ref={this.storeInputReference}

            />

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

export default connect(mapStateToProps, null)(InputSuggestion);

**InputSuggest.scss**

@import "../../styles/common";

.react-autosuggest\_\_suggestions-container {

    min-width: 90%;

    position: absolute;

    top: 50px;

    z-index: 4;

    max-height: 140px;

    overflow: auto;

    background: $footer-container;

    box-shadow: 2px 2px 4px 0 $box-shadow-color;

    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

}

.react-autosuggest\_\_suggestions-list {

    list-style-type: none;

    margin-bottom: 0;

    padding: 0px;

}

.react-autosuggest\_\_suggestion {

    padding: 0 5px;

    height: 28px;

    line-height: 28px;

    border-bottom: 1px solid $border;

    &:hover {

        cursor: pointer;

        background-color: darken($footer-container, 5);

    }

}

.react-autosuggest\_\_suggestion--highlighted {

    background-color: darken($footer-container, 5);

}

***src/components/Breadcrumb.js***

import React from 'react';

import { useLocation, Link } from 'react-router-dom';

const Breadcrumb = () => {

   const location = useLocation();

   const pathnames = location.pathname.split('/').filter(x => x);

   return (

      <nav className="breadcrumb">

         <Link to="/">Trang chủ</Link>

         {pathnames.map((name, index) => {

            const routeTo = `/${pathnames.slice(0, index + 1).join('/')}`;

            return <span key={index}> / <Link to={routeTo}>{name}</Link></span>;

         })}

      </nav>

   );

};

export default Breadcrumb;

***src/components/ConfirmModal.js***

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

import { Modal } from 'reactstrap';

import './ConfirmModal.scss';

import \* as actions from "../store/actions";

import { KeyCodeUtils } from "../utils";

class ConfirmModal extends Component {

    constructor(props) {

        super(props);

        this.acceptBtnRef = React.createRef();

    }

    initialState = {

    };

    state = {

        ...this.initialState

    };

    componentDidMount() {

        document.addEventListener('keydown', this.handlerKeyDown);

    }

    componentWillUnmount() {

        document.removeEventListener('keydown', this.handlerKeyDown);

    }

    handlerKeyDown = (event) => {

        const keyCode = event.which || event.keyCode;

        if (keyCode === KeyCodeUtils.ENTER) {

            if (!this.acceptBtnRef.current || this.acceptBtnRef.current.disabled) return;

            this.acceptBtnRef.current.click();

        }

    }

    onAcceptBtnClick = () => {

        const { contentOfConfirmModal } = this.props;

        if (contentOfConfirmModal.handleFunc) {

            contentOfConfirmModal.handleFunc(contentOfConfirmModal.dataFunc);

        }

        this.onClose();

    }

    onClose = () => {

        this.props.setContentOfConfirmModal({

            isOpen: false,

            messageId: "",

            handleFunc: null,

            dataFunc: null

        });

    }

    render() {

        const { contentOfConfirmModal } = this.props;

        return (

            <Modal isOpen={contentOfConfirmModal.isOpen} className='confirm-modal' centered={true}>

                <div className="modal-header">

                    <div className="modal-title">

                        <FormattedMessage id={"common.confirm"} />

                    </div>

                    <div className="col-auto">

                        <button className="btn btn-close" onClick={this.onClose}>

                            <i className="fal fa-times" />

                        </button>

                    </div>

                </div>

                <div className="modal-body">

                    <div className="confirm-modal-content">

                        <div className="row">

                            <div className="col-12">

                                <FormattedMessage id={contentOfConfirmModal.messageId ? contentOfConfirmModal.messageId : "common.confirm-this-task"} />

                            </div>

                            <hr />

                            <div className="col-12">

                                <div className="btn-container text-center">

                                    <button className="btn btn-add" onClick={this.onClose} >

                                        <FormattedMessage id="common.close" />

                                    </button>

                                    <button ref={this.acceptBtnRef} className="btn btn-add" onClick={this.onAcceptBtnClick}>

                                        <FormattedMessage id={"common.accept"} />

                                    </button>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </Modal >

        );

    }

}

const mapStateToProps = state => {

    return {

        lang: state.app.language,

        contentOfConfirmModal: state.app.contentOfConfirmModal

    };

};

const mapDispatchToProps = dispatch => {

    return {

        setContentOfConfirmModal: (contentOfConfirmModal) => dispatch(actions.setContentOfConfirmModal(contentOfConfirmModal))

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(ConfirmModal);

***src/components/ConfirmModal.scss***

@import "../styles/common";

.confirm-modal {

  &.modal-dialog {

    @media (min-width: 766px) {

      min-width: 650px;

    }

  }

  hr {

    border: none;

    border-bottom: 1px solid $border;;

    width: 100%;

    margin: 10px 15px 5px 15px;

  }

  .btn-container {

    margin-top: 10px;

    .btn {

      width: 100px;

      &:first-child {

        margin-right: 10px;

      }

    }

  }

  .custom-form-group {

    margin: 5px 0;

    .custom-form-control.readonly {

      min-height: 28px;

      height: auto;

    }

  }

}

***src/components/CustomScrollbars.js***

import React, { Component } from 'react';

import { Scrollbars } from 'react-custom-scrollbars-2';

import './CustomScrollbars.scss';

class CustomScrollbars extends Component {

    ref = React.createRef();

    getScrollLeft = () => {

        const scrollbars = this.ref.current;

        return scrollbars.getScrollLeft();

    }

    getScrollTop = () => {

        const scrollbars = this.ref.current;

        return scrollbars.getScrollTop();

    }

    scrollToBottom = () => {

        if (!this.ref || !this.ref.current) {

            return;

        }

        const scrollbars = this.ref.current;

        const targetScrollTop = scrollbars.getScrollHeight();

        this.scrollTo(targetScrollTop);

    };

    scrollTo = (targetTop) => {

        const { quickScroll } = this.props;

        if (!this.ref || !this.ref.current) {

            return;

        }

        const scrollbars = this.ref.current;

        const originalTop = scrollbars.getScrollTop();

        let iteration = 0;

        const scroll = () => {

            iteration++;

            if (iteration > 30) {

                return;

            }

            scrollbars.scrollTop(originalTop + (targetTop - originalTop) / 30 \* iteration);

            if (quickScroll && quickScroll === true) {

                scroll();

            } else {

                setTimeout(() => {

                    scroll();

                }, 20);

            }

        };

        scroll();

    };

    renderTrackHorizontal = (props) => {

        return (

            <div {...props} className="track-horizontal" />

        );

    };

    renderTrackVertical = (props) => {

        return (

            <div {...props} className="track-vertical" />

        );

    };

    renderThumbHorizontal = (props) => {

        return (

            <div {...props} className="thumb-horizontal" />

        );

    };

    renderThumbVertical = (props) => {

        return (

            <div {...props} className="thumb-vertical" />

        );

    };

    renderNone = (props) => {

        return (

            <div />

        );

    };

    render() {

        const { className, disableVerticalScroll, disableHorizontalScroll, children, ...otherProps } = this.props;

        return (

            <Scrollbars

                ref={this.ref}

                autoHide={true}

                autoHideTimeout={200}

                hideTracksWhenNotNeeded={true}

                className={className ? className + ' custom-scrollbar' : 'custom-scrollbar'}

                {...otherProps}

                renderTrackHorizontal={disableHorizontalScroll ? this.renderNone : this.renderTrackHorizontal}

                renderTrackVertical={disableVerticalScroll ? this.renderNone : this.renderTrackVertical}

                renderThumbHorizontal={disableHorizontalScroll ? this.renderNone : this.renderThumbHorizontal}

                renderThumbVertical={disableVerticalScroll ? this.renderNone : this.renderThumbVertical}

            >

                {children}

            </Scrollbars>

        );

    }

}

export default CustomScrollbars;

***src/components/CustomScrollbars.scss***

@import "../styles/common.scss";

.custom-scrollbar {

  .thumb-horizontal {

    background-color: rgba($bg-scrollbar, 0.4);

    box-shadow: inset 1px 1px 0 rgba($bg-scrollbar, 0.10), inset 0 -1px 0 rgba($bg-scrollbar, 0.07);

    &:hover {

      background-color: rgba($bg-scrollbar, 0.8);

      box-shadow: inset 1px 1px 1px rgba($bg-scrollbar, 0.50);

    }

    &:active {

      background-color: rgba($bg-scrollbar, 0.9);

      box-shadow: inset 1px 1px 3px rgba($bg-scrollbar, 0.6);

    }

  }

  .thumb-vertical {

    background-color: rgba($bg-scrollbar, 0.4);

    box-shadow: inset 1px 1px 0 rgba($bg-scrollbar, 0.10), inset 0 -1px 0 rgba($bg-scrollbar, 0.07);

    &:hover {

      background-color: rgba($bg-scrollbar, 0.8);

      box-shadow: inset 1px 1px 1px rgba($bg-scrollbar, 0.50);

    }

    &:active {

      background-color: rgba($bg-scrollbar, 0.9);

      box-shadow: inset 1px 1px 3px rgba($bg-scrollbar, 0.6);

    }

  }

  &:hover {

    > .track-horizontal, > .track-vertical {

      opacity: 1 !important;

      > .thumb-horizontal, > .thumb-vertical {

        opacity: 1;

      }

    }

  }

  .track-horizontal {

    right: 0;

    bottom: 0;

    left: 0;

    height: 5px !important;

    &:hover, &:active {

      height: 10px !important;

    }

  }

  .track-vertical {

    right: 0;

    bottom: 0;

    top: 0;

    width: 5px !important;

    &:hover, &:active {

      width: 10px !important;

    }

  }

  .thumb-horizontal {

    cursor: pointer;

    border-radius: 10px;

    background-clip: padding-box;

    min-width: 0px;

  }

  .thumb-vertical {

    cursor: pointer;

    border-radius: 10px;

    background-clip: padding-box;

    min-height: 0px;

  }

}

***src/components/CustomToast.js***

import React, { Component, Fragment } from 'react';

import { FormattedMessage, FormattedTime } from 'react-intl';

import CustomScrollBar from '../components/CustomScrollbars';

import './CustomToast.scss';

class CustomToast extends Component {

    render() {

        const { titleId, message, messageId, time } = this.props;

        return (

            <Fragment>

                <div className="custom-toast">

                    <div className="toast-title">

                        {time && (

                            <span className="date">

                                <FormattedTime hour='numeric' minute='numeric' second='numeric' hour12={true} value={time} />

                            </span>

                        )}

                        <i className="fa fa-fw fa-exclamation-triangle" />

                        <FormattedMessage id={titleId} />

                    </div>

                    {

                        (message && typeof message === 'object') ?

                            <CustomScrollBar autoHeight={true} autoHeightMin={50} autoHeightMax={100}>

                                {

                                    message.map((msg, index) => {

                                        return (

                                            <Fragment key={index}>

                                                <div className="toast-content">{msg}</div>

                                            </Fragment>

                                        )

                                    })

                                }

                            </CustomScrollBar> :

                            <div className="toast-content">

                                {message ? message : (messageId ? (<FormattedMessage id={messageId} />) : null)}

                            </div>

                    }

                </div>

            </Fragment>

        );

    }

}

export class CustomToastCloseButton extends Component {

    render() {

        return (

            <button type="button" className="toast-close" onClick={this.props.closeToast}>

                <i className="fa fa-fw fa-times-circle" />

            </button>

        );

    }

}

export default CustomToast;

***src/components/CustomToast.scss***

@import "../styles/common.scss";

.toast-container {

  .toast-item {

    white-space: pre-wrap;

    box-shadow: 0 0 5px 5px $box-shadow-color;

    -webkit-box-shadow: 0 0 5px 5px $box-shadow-color;

    &.Toastify\_\_toast--success {

      background: $common-green;

    }

    &.Toastify\_\_toast--error {

      background: $common-red;

    }

    &.Toastify\_\_toast--warn {

      background: $colormain;

    }

    &.Toastify\_\_toast--info {

      background: $colormain;

    }

    .toast-close {

      position: absolute;

      right: 10px;

      top:10px;

      color: $common-text-color;

      font-size: $base-size;

      padding: 0;

      cursor: pointer;

      background: transparent;

      border: 0;

      -webkit-appearance: none;

      transition: opacity 0.2s ease-out;

    }

    .toast-item-body {

      color: $common-text-color;

      display: block;

      flex: none;

      width: 100%;

      .toast-title {

        font-weight: bold;

        font-size: $base-size;

        margin: 0 20px 5px 0;

        .fixed-scroll-bar {

          height: 50px;

        }

        .date {

          float: right;

          font-size: $base-size - 2px;

          vertical-align: middle;

          margin-right: 5px;

          margin-bottom: 0;

          padding: 2px 5px;

        }

        i {

          position: relative;

          margin-right: 3px;

        }

      }

      .toast-content {

        font-size: $base-size - 2px;

      }

    }

  }

}

***src/components/Navigator.js***

import React, { Component, Fragment } from 'react';

import { Link, withRouter } from 'react-router-dom';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

import './Navigator.scss';

class MenuGroup extends Component {

    render() {

        const { name, children } = this.props;

        return (

            <li className="menu-group">

                <div className="menu-group-name">

                    <FormattedMessage id={name} />

                </div>

                <ul className="menu-list list-unstyled">

                    {children}

                </ul>

            </li>

        );

    }

}

class Menu extends Component {

    render() {

        const { name, active, link, children, onClick, hasSubMenu, onLinkClick } = this.props;

        return (

            <li className={"menu" + (hasSubMenu ? " has-sub-menu" : "") + ("") + (active ? " active" : "")}>

                {hasSubMenu ? (

                    <Fragment>

                        <span

                            data-toggle="collapse"

                            className={"menu-link collapsed"}

                            onClick={onClick}

                            aria-expanded={"false"}

                        >

                            <FormattedMessage id={name} />

                            <div className="icon-right">

                                <i className={"far fa-angle-right"} />

                            </div>

                        </span>

                        <div>

                            <ul className="sub-menu-list list-unstyled">

                                {children}

                            </ul>

                        </div>

                    </Fragment>

                ) : (

                        <Link to={link} className="menu-link" onClick={onLinkClick}>

                            <FormattedMessage id={name} />

                        </Link>

                    )}

            </li>

        );

    }

}

class SubMenu extends Component {

    getItemClass = path => {

        return this.props.location.pathname === path ? "active" : "";

    };

    render() {

        const { name, link, onLinkClick } = this.props;

        return (

            <li className={"sub-menu " + this.getItemClass(link)}>

                <Link to={link} className="sub-menu-link" onClick={onLinkClick}>

                    <FormattedMessage id={name} />

                </Link>

            </li>

        );

    }

}

const MenuGroupWithRouter = withRouter(MenuGroup);

const MenuWithRouter = withRouter(Menu);

const SubMenuWithRouter = withRouter(SubMenu);

const withRouterInnerRef = (WrappedComponent) => {

    class InnerComponentWithRef extends React.Component {

        render() {

            const { forwardRef, ...rest } = this.props;

            return <WrappedComponent {...rest} ref={forwardRef} />;

        }

    }

    const ComponentWithRef = withRouter(InnerComponentWithRef, { withRef: true });

    return React.forwardRef((props, ref) => {

        return <ComponentWithRef {...props} forwardRef={ref} />;

    });

};

class Navigator extends Component {

    state = {

        expandedMenu: {}

    };

    toggle = (groupIndex, menuIndex) => {

        const expandedMenu = {};

        const needExpand = !(this.state.expandedMenu[groupIndex + '\_' + menuIndex] === true);

        if (needExpand) {

            expandedMenu[groupIndex + '\_' + menuIndex] = true;

        }

        this.setState({

            expandedMenu: expandedMenu

        });

    };

    isMenuHasSubMenuActive = (location, subMenus, link) => {

        if (subMenus) {

            if (subMenus.length === 0) {

                return false;

            }

            const currentPath = location.pathname;

            for (let i = 0; i < subMenus.length; i++) {

                const subMenu = subMenus[i];

                if (subMenu.link === currentPath) {

                    return true;

                }

            }

        }

        if (link) {

            return this.props.location.pathname === link;

        }

        return false;

    };

    checkActiveMenu = () => {

        const { menus, location } = this.props;

        outerLoop:

        for (let i = 0; i < menus.length; i++) {

            const group = menus[i];

            if (group.menus && group.menus.length > 0) {

                for (let j = 0; j < group.menus.length; j++) {

                    const menu = group.menus[j];

                    if (menu.subMenus && menu.subMenus.length > 0) {

                        if (this.isMenuHasSubMenuActive(location, menu.subMenus, null)) {

                            const key = i + '\_' + j;

                            this.setState({

                                expandedMenu: {

                                    [key]: true

                                }

                            });

                            break outerLoop;

                        }

                    }

                }

            }

        }

    };

    componentDidMount() {

        this.checkActiveMenu();

    };

    // componentWillReceiveProps(nextProps, prevState) {

    //     const { location, setAccountMenuPath, setSettingMenuPath } = this.props;

    //     const { location: nextLocation } = nextProps;

    //     if (location !== nextLocation) {

    //         let pathname = nextLocation && nextLocation.pathname;

    //         if ((pathname.startsWith('/account/') || pathname.startsWith('/fds/account/'))) {

    //             setAccountMenuPath(pathname);

    //         }

    //         if (pathname.startsWith('/settings/')) {

    //             setSettingMenuPath(pathname);

    //         };

    //     };

    // };

    componentDidUpdate(prevProps, prevState) {

        const { location } = this.props;

        const { location: prevLocation } = prevProps;

        if (location !== prevLocation) {

            this.checkActiveMenu();

        };

    };

    render() {

        const { menus, location, onLinkClick } = this.props;

        return (

            <Fragment>

                <ul className="navigator-menu list-unstyled">

                    {

                        menus.map((group, groupIndex) => {

                            return (

                                <Fragment key={groupIndex}>

                                    <MenuGroupWithRouter name={group.name}>

                                        {group.menus ? (

                                            group.menus.map((menu, menuIndex) => {

                                                const isMenuHasSubMenuActive = this.isMenuHasSubMenuActive(location, menu.subMenus, menu.link);

                                                const isSubMenuOpen = this.state.expandedMenu[groupIndex + '\_' + menuIndex] === true;

                                                return (

                                                    <MenuWithRouter

                                                        key={menuIndex}

                                                        active={isMenuHasSubMenuActive}

                                                        name={menu.name}

                                                        link={menu.link}

                                                        hasSubMenu={menu.subMenus}

                                                        isOpen={isSubMenuOpen}

                                                        onClick={() => this.toggle(groupIndex, menuIndex)}

                                                        onLinkClick={onLinkClick}

                                                    >

                                                        {menu.subMenus && menu.subMenus.map((subMenu, subMenuIndex) => (

                                                            <SubMenuWithRouter

                                                                key={subMenuIndex}

                                                                name={subMenu.name}

                                                                link={subMenu.link}

                                                                onClick={this.closeOtherExpand}

                                                                onLinkClick={onLinkClick}

                                                            />

                                                        ))}

                                                    </MenuWithRouter>

                                                );

                                            })

                                        ) : null}

                                    </MenuGroupWithRouter>

                                </Fragment>

                            );

                        })

                    }

                </ul>

            </Fragment>

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    }

}

export default withRouterInnerRef(connect(mapStateToProps, mapDispatchToProps)(Navigator));

***src/components/Navigator.scss***

@import "../styles/common.scss";

.navigator-menu {

    display: flex;

    margin: 0;

    padding: 0;

    &.list-unstyled,

    .list-unstyled {

        list-style-type: none;

    }

    .menu-group {

        &:hover {

            cursor: pointer;

            background-color: darken($colormain, 5);

        }

        .menu-group-name {

            line-height: 40px;

            padding: 0 15px;

            border-right: 1px solid $border;

            &:first-child {

                border-left: 1px solid $border;

            }

        }

        .menu-list {

            display: none;

            background-color: $bg-menu-color;

            box-shadow: 2px 2px 4px 0 $box-shadow-color;

            -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

            color: $text-in-light;

            position: absolute;

            padding: 0;

            .menu {

                width: 205px;

                padding: 0 15px;

                height: 35px;

                line-height: 35px;

                text-transform: none;

                .menu-link {

                    text-decoration: none;

                    color: $text-in-light;

                }

                .sub-menu-list {

                    display: none;

                    background-color: $bg-menu-color;

                    box-shadow: 2px 2px 4px 0 $box-shadow-color;

                    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

                    position: absolute;

                    top: 0;

                    left: 205px;

                    padding: 0;

                    .sub-menu {

                        padding: 0 15px;

                        height: 35px;

                        line-height: 35px;

                        white-space: nowrap;

                        &:hover {

                            background-color: darken($bg-menu-color, 5);

                        }

                        .sub-menu-link {

                            text-decoration: none;

                            color: $text-in-light;

                        }

                        a {

                            display: block;

                        }

                        &.active a {

                            font-weight: 500;

                            color: $colormain;

                        }

                    }

                }

                &.active span {

                    font-weight: 500;

                    color: $colormain;

                }

                &:hover {

                    background-color: darken($bg-menu-color, 3);

                    .sub-menu-list {

                        display: block;

                    }

                }

                .icon-right {

                    display: block;

                    position: absolute;

                    top: 0;

                    right: 10px;

                }

            }

        }

        &:hover {

            .menu-list {

                display: block;

            }

        }

    }

}

**src/hoc/**

**Authentication.js**

import locationHelperBuilder from "redux-auth-wrapper/history4/locationHelper";

import { connectedRouterRedirect } from "redux-auth-wrapper/history4/redirect";

const locationHelper = locationHelperBuilder({});

export const userIsAuthenticated = connectedRouterRedirect({

    authenticatedSelector: state => state.admin.isLoggedIn,

    wrapperDisplayName: 'UserIsAuthenticated',

    redirectPath: '/login'

});

export const userIsNotAuthenticated = connectedRouterRedirect({

    // Want to redirect the user when they are authenticated

    authenticatedSelector: state => !state.admin.isLoggedIn,

    wrapperDisplayName: 'UserIsNotAuthenticated',

    redirectPath: (state, ownProps) => locationHelper.getRedirectQueryParam(ownProps) || '/',

    allowRedirectBack: false

});

**IntProviderWrapper.js**

import React, { Component } from "react";

import { connect } from 'react-redux';

import { IntlProvider } from "react-intl";

import '@formatjs/intl-pluralrules/polyfill';

import '@formatjs/intl-pluralrules/locale-data/en';

import '@formatjs/intl-pluralrules/locale-data/vi';

import '@formatjs/intl-relativetimeformat/polyfill';

import '@formatjs/intl-relativetimeformat/locale-data/en';

import '@formatjs/intl-relativetimeformat/locale-data/vi';

import { LanguageUtils } from '../utils'

const messages = LanguageUtils.getFlattenedMessages();

class IntlProviderWrapper extends Component {

    render() {

        const { children, language } = this.props;

        return (

            <IntlProvider

                locale={language}

                messages={messages[language]}

                defaultLocale="vi">

                {children}

            </IntlProvider>

        );

    }

}

const mapStateToProps = state => {

    return {

        language: state.app.language

    };

};

export default connect(mapStateToProps, null)(IntlProviderWrapper);

**PrivateRoute.js**

import React from 'react';

import { Route, Redirect } from 'react-router-dom';

const PrivateRoute = ({ component: Component, role, render, ...rest }) => {

   const token = localStorage.getItem('token');

   const roleId = localStorage.getItem('roleId');

   return (

      <Route

         {...rest}

         render={(props) => {

            if (!token) return <Redirect to="/login" />;

            if (role && roleId !== role) return <Redirect to="/" />;

            return render ? render(props) : <Component {...props} />;

         }}

      />

   );

};

export default PrivateRoute;

**src/layouts/**

**AdminLayout.js**

Đang phát triển…

**PublicLayout.js**

import React from 'react';

import Header from '../components/Header/Header';

import Footer from '../components/Footer/Footer';

const PublicLayout = (props) => {

   return (

      <>

         <Header />

         <main className="main-content">

            {props.children}

         </main>

         <Footer />

      </>

   );

};

export default PublicLayout;

**src/pages/**

***src/admin/***

**ProductManager.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class ProductManage extends Component {

    state = {

    }

    componentDidMount() {

    }

    render() {

        return (

            <div className="text-center" >Manage products</div>

        )

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(ProductManage);

**RegisterPackageGroupOrAcc.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class RegisterPackageGroupOrAcc extends Component {

    constructor(props) {

        super(props);

    }

    render() {

        return (

            <div className="text-center">

                register package group or account

            </div>)

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(RegisterPackageGroupOrAcc);

**UserManager.js**

import React, { Component } from 'react';

import { FormattedMessage } from 'react-intl';

import { connect } from 'react-redux';

class UserManage extends Component {

    state = {

    }

    componentDidMount() {

    }

    render() {

        return (

            <div className="text-center">Manage users</div>

        );

    }

}

const mapStateToProps = state => {

    return {

    };

};

const mapDispatchToProps = dispatch => {

    return {

    };

};

export default connect(mapStateToProps, mapDispatchToProps)(UserManage);

***src/auth/***

**Login.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Login.scss';

import { FormattedMessage } from 'react-intl';

class Login extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>login</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Login);

**Login.scss**

***src/home/***

**Home.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Home.scss';

import { FormattedMessage } from 'react-intl';

class Home extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>Home</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Home);

**Home.scss**

***src/user/***

**Profile.js**

import React, { Component } from 'react';

import { connect } from 'react-redux';

import { push } from "connected-react-router";

import \* as actions from "../../store/actions";

import './Profile.scss';

import { FormattedMessage } from 'react-intl';

class Profile extends Component {

   constructor(props) {

      super(props);

   }

   render() {

      return (

         <div>Profile</div>

      )

   }

}

const mapStateToProps = state => {

   return {

      lang: state.app.language

   };

};

const mapDispatchToProps = dispatch => {

   return {

      navigate: (path) => dispatch(push(path)),

      adminLoginSuccess: (adminInfo) => dispatch(actions.adminLoginSuccess(adminInfo)),

      adminLoginFail: () => dispatch(actions.adminLoginFail()),

   };

};

export default connect(mapStateToProps, mapDispatchToProps)(Profile);

**Profile.scss**

**src/routes/**

***AppRoutes.js***

import React from 'react';

import { Switch, Route, Redirect } from 'react-router-dom';

import PublicLayout from '../layouts/PublicLayout';

import AdminLayout from '../layouts/AdminLayout';

import PrivateRoute from '../hoc/PrivateRoute';

import Home from '../pages/home/Home';

import Login from '../pages/auth/Login';

import Profile from '../pages/user/Profile';

const AppRoutes = () => {

   return (

      <Switch>

         {/\* Trang HOME - không cần đăng nhập \*/}

         <Route

            path="/"

            exact

            render={(props) => (

               <PublicLayout>

                  <Home {...props} />

               </PublicLayout>

            )}

         />

         {/\* Login \*/}

         <Route

            path="/login"

            render={(props) => (

               <PublicLayout>

                  <Login {...props} />

               </PublicLayout>

            )}

         />

         {/\* Profile - cần login với role = 2 (user) \*/}

         <PrivateRoute

            path="/profile"

            role="2"

            render={(props) => (

               <PublicLayout>

                  <Profile {...props} />

               </PublicLayout>

            )}

         />

         {/\* Admin - cần login với role = 1 (admin) \*/}

         <PrivateRoute

            path="/admin"

            role="1"

            render={(props) => (

               <AdminLayout>

                  <div>Trang admin</div>

               </AdminLayout>

            )}

         />

         {/\* Redirect fallback \*/}

         <Redirect to="/" />

      </Switch>

   );

};

export default AppRoutes;

**src/services/**

***adminService.js***

import axios from '../axios';

import \* as queryString from 'query-string';

const adminService = {

    /\*\*

     \* Đăng nhập hệ thống

     \* {

     \*  "username": "string",

     \*  "password": "string"

     \* }

     \*/

    login(loginBody) {

        return axios.post(`/admin/login`, loginBody)

    },

};

export default adminService;

***index.js***

export { default as adminService } from './adminService';

**src/store/**

***src/store/actions/***

**actionType.js**

const actionTypes = Object.freeze({

    //app

    APP\_START\_UP\_COMPLETE: 'APP\_START\_UP\_COMPLETE',

    SET\_CONTENT\_OF\_CONFIRM\_MODAL: 'SET\_CONTENT\_OF\_CONFIRM\_MODAL',

    //admin

    ADMIN\_LOGIN\_SUCCESS: 'ADMIN\_LOGIN\_SUCCESS',

    ADMIN\_LOGIN\_FAIL: 'ADMIN\_LOGIN\_FAIL',

    PROCESS\_LOGOUT: 'PROCESS\_LOGOUT',

    //user

    ADD\_USER\_SUCCESS: 'ADD\_USER\_SUCCESS',

})

export default actionTypes;

**adminActions.js**

import actionTypes from './actionTypes';

export const adminLoginSuccess = (adminInfo) => ({

    type: actionTypes.ADMIN\_LOGIN\_SUCCESS,

    adminInfo: adminInfo

})

export const adminLoginFail = () => ({

    type: actionTypes.ADMIN\_LOGIN\_FAIL

})

export const processLogout = () => ({

    type: actionTypes.PROCESS\_LOGOUT

})

**appActions.js**

import actionTypes from './actionTypes';

export const appStartUpComplete = () => ({

    type: actionTypes.APP\_START\_UP\_COMPLETE

});

export const setContentOfConfirmModal = (contentOfConfirmModal) => ({

    type: actionTypes.SET\_CONTENT\_OF\_CONFIRM\_MODAL,

    contentOfConfirmModal: contentOfConfirmModal

});

**index.js**

export \* from './appActions'

export \* from './adminActions'

export \* from './userActions'

**userActions.js**

import actionTypes from './actionTypes';

export const addUserSuccess = () => ({

    type: actionTypes.ADD\_USER\_SUCCESS

})

***src/store/reducers/***

**adminReducer.js**

import actionTypes from '../actions/actionTypes';

const initialState = {

    isLoggedIn: false,

    adminInfo: null

}

const appReducer = (state = initialState, action) => {

    switch (action.type) {

        case actionTypes.ADMIN\_LOGIN\_SUCCESS:

            return {

                ...state,

                isLoggedIn: true,

                adminInfo: action.adminInfo

            }

        case actionTypes.ADMIN\_LOGIN\_FAIL:

            return {

                ...state,

                isLoggedIn: false,

                adminInfo: null

            }

        case actionTypes.PROCESS\_LOGOUT:

            return {

                ...state,

                isLoggedIn: false,

                adminInfo: null

            }

        default:

            return state;

    }

}

export default appReducer;

**appReducer.js**

import actionTypes from '../actions/actionTypes';

const initContentOfConfirmModal = {

    isOpen: false,

    messageId: "",

    handleFunc: null,

    dataFunc: null

}

const initialState = {

    started: true,

    language: 'vi',

    systemMenuPath: '/system/user-manage',

    contentOfConfirmModal: {

        ...initContentOfConfirmModal

    }

}

const appReducer = (state = initialState, action) => {

    switch (action.type) {

        case actionTypes.APP\_START\_UP\_COMPLETE:

            return {

                ...state,

                started: true

            }

        case actionTypes.SET\_CONTENT\_OF\_CONFIRM\_MODAL:

            return {

                ...state,

                contentOfConfirmModal: {

                    ...state.contentOfConfirmModal,

                    ...action.contentOfConfirmModal

                }

            }

        default:

            return state;

    }

}

export default appReducer;

**rootReducer.js**

import {combineReducers} from 'redux';

import { connectRouter } from 'connected-react-router';

import appReducer from "./appReducer";

import adminReducer from "./adminReducer";

import userReducer from "./userReducer";

import autoMergeLevel2 from 'redux-persist/lib/stateReconciler/autoMergeLevel2';

import storage from 'redux-persist/lib/storage';

import { persistReducer } from 'redux-persist';

const persistCommonConfig = {

    storage: storage,

    stateReconciler: autoMergeLevel2,

};

const adminPersistConfig = {

    ...persistCommonConfig,

    key: 'admin',

    whitelist: ['isLoggedIn', 'adminInfo']

};

export default (history) => combineReducers({

    router: connectRouter(history),

    admin: persistReducer(adminPersistConfig, adminReducer),

    user: userReducer,

    app: appReducer

})

**userReducer.js**

Đang phát triển…

**src/styles/**

***\_base.scss***

/\*

 \* General styles

 \*/

 body, html {

    background-image: "#fff";

    background-repeat: no-repeat;

    background-size: cover;

    font-family: $fontmain;

    font-size: $base-size - 2px;

    height: 100vh;

    overflow: hidden;

}

.btn {

    padding: 0;

    border: none;

    cursor: pointer;

    font-weight: 700;

    height: 30px;

    line-height: 30px;

    -moz-user-select: none;

    -webkit-user-select: none;

    user-select: none;

    i {

      font-size: $base-size - 3px;

    }

}

.btn-edit,

.btn-add,

.btn-delete,

.btn-assign {

  background-color: $colormain;

  color: $common-text-color;

  &:hover {

    color: $common-text-color;

    background-color: darken($colormain, 5);

  }

}

#password:invalid {

  font-family: $fontmain; // Fixed TH hien thi placeholder cho EDGE

}

#password:-webkit-input-placeholder {

  font-family: $fontmain; // CHROME 1

}

#password::-webkit-input-placeholder {

  font-family: $fontmain; //CHROME 2   EDGE 1

}

#password:-moz-placeholder {

  font-family: $fontmain; // Firefox 4-18

}

#password::-moz-placeholder {

  font-family: $fontmain; // Firefox 19+

}

#password:-ms-input-placeholder {

  font-family: $fontmain; // IE 10-11

}

#password::-ms-input-placeholder {

  font-family: $fontmain; // EDGE 2

}

#password::placeholder {

  font-family: $fontmain;

}

.content-container {

  .title {

    text-align: center;

    text-transform: uppercase;

    font-weight: bold;

    font-size: $base-size + 4px;

    margin-top: 15px;

    color: $colormain;

  }

  .content {

    margin: 15px 30px;

    box-shadow: 2px 2px 4px 0 $box-shadow-color;

    -webkit-box-shadow: 2px 2px 4px 0 $box-shadow-color;

  }

}

.modal {

  .modal-dialog {

    .modal-content {

      .modal-header {

        padding: 0 10px;

        height: 35px;

        background-color: $colormain;

        color: $common-text-color;

        .modal-title {

          line-height: 35px;

          font-weight: 500;

          font-size: $base-size + 1px;

        }

        .btn-close {

          position: absolute;

          color: $common-text-color;

          i {

            line-height: 35px;

            font-size: $base-size + 2px;

          }

        }

      }

    }

  }

}

***\_form.scss***

@import "common";

.custom-form-group {

  .custom-form-control {

    background-color: $common-white;

    border: 1px solid $border;

    &.readonly {

      cursor: not-allowed;

      background-color: darken($common-white, 5);

    }

    &:focus {

      outline: none;

      border: 1px solid darken($border, 50);

    }

  }

  select.custom-form-control {

    background-image: $dropdown-image;

  }

  margin-bottom: 10px;

  label {

    margin-bottom: 1px;

  }

  .custom-form-control {

    display: block;

    width: 100%;

    height: 28px;

    padding: 0 5px;

    line-height: 26px;

    border-radius: 5px;

  }

  textarea.custom-form-control {

    min-height: 28px;

    height: auto;

  }

  select.custom-form-control {

    -webkit-appearance: none;

    -moz-appearance: none;

    padding-right: 15px;

    background-position: right top;

    background-repeat: no-repeat;

    background-size: 15px 100%;

    cursor: pointer;

    &:disabled {

      cursor: not-allowed;

    }

  }

  .custom-checkbox-control {

    line-height: 28px;

    input {

      cursor: pointer;

    }

    label {

      margin-left: 5px;

    }

  }

}

***\_variables.scss***

$base-size: 16px;

$lh: 1.313;

$fontmain:"Helvetica", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";

$colormain:#0071ba;

$colorsub:#bb8d09;

$common-text-color:white;

$common-btn-confirm: #185ba8;

$common-btn-deny: #bb8d09;

$common-error: #ff9600;

$common-notify: greenyellow;

$common-green: #00C087;

$common-red: #d01a1d;

$common-white: #fff;

$common-orange: #FFAC42;

$border: #ccc;

$text-in-light: #181818;

$input-focus: #FFAC42;

$footer-container: #f5f5f5;

$footer-text: #262626;

$footer-next-disable: #ccc;

$bg-menu-color: #f5f5f5;

$bg-scrollbar: #000;

$bg-table-color: #f5f5f5;

$box-shadow-color: #0000004d;

$dropdown-image: url("../assets/images/dropdown.svg");

//date picker

$date-picker-bg: $common-white;

$date-picker-border: darken(#3f4458, 50%);

$date-picker-arrow-hover: $common-orange;

$date-picker-month-fg: #000;

$date-picker-month-bg: $common-white;

$date-picker-weekdays-fg: #000;

$date-picker-weekdays-bg: transparent;

$date-picker-day-fg: #000;

$date-picker-day-hover-bg: darken($common-white, 25%);

$date-picker-today-border: $common-orange;

$date-picker-today-bg: $common-orange;

$date-picker-today-fg: #000;

$date-picker-day-selected-fg: $common-white;

***common.scss***

@import "variables";

.text-center {

    text-align: center !important;

}

.text-left {

    text-align: left !important;

}

.text-right {

    text-align: right !important;

}

.text-red {

    color: $common-red !important;

}

***styles.scss***

@import "variables";

@import "../../node\_modules/bootstrap/scss/bootstrap";

$fa-font-path : "~@fortawesome/fontawesome-free-webfonts/webfonts";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fontawesome.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-solid.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-regular.scss";

@import "~@fortawesome/fontawesome-free-webfonts/scss/fa-brands.scss";

@import "base";

@import "form";

**src/translations/**

***en.json***

{

    "common": {

        "add": "Add",

        "edit": "Edit",

        "delete": "Delete"

    },

    "login": {

        "login": "Login",

        "username": "Username",

        "password": "Password"

    }

}

***vi.json***

{

    "common": {

        "add": "Thêm",

        "edit": "Sửa",

        "delete": "Xóa"

    },

    "login": {

        "login": "Đăng nhập",

        "username": "Tên đăng nhập",

        "password": "Mật khẩu"

    }

}

**src/utils/**

***CommonUtils.js***

class CommonUtils {

    static isNumber1 (number) {

        if (number === 1) return true;

        return false;

    }

}

export default CommonUtils;

***Constant.js***

export const path = {

    HOME: '/',

    LOGIN: '/login',

    LOG\_OUT: '/logout',

    SYSTEM: '/system'

};

export const languages = {

    VI: 'vi',

    EN: 'en'

};

export const manageActions = {

    ADD: "ADD",

    EDIT: "EDIT",

    DELETE: "DELETE"

};

export const dateFormat = {

    SEND\_TO\_SERVER: 'DD/MM/YYYY'

};

export const YesNoObj = {

    YES: 'Y',

    NO: 'N'

}

***index.js***

export \* from './constant';

export {default as CommonUtils} from './CommonUtils';

export {default as KeyCodeUtils} from './KeyCodeUtils';

export {default as LanguageUtils} from './LanguageUtils';

export {default as ToastUtil} from './ToastUtil';

***KeyCodeUtils.js***

class KeyCodeUtils {

    static UP = 38;

    static DOWN = 40;

    static TAB = 9;

    static ENTER = 13;

    static E = 69;

    static ESCAPE = 27;

    static isNavigation(e) {

        return (e >= 33 && e <= 40) || e === 9 || e === 8 || e === 46 || e === 14 || e === 13;

    }

    static isNumeric(e) {

        return (e >= 48 && e <= 57) || (e >= 96 && e <= 105);

    }

    static isAlphabetic(e) {

        return (e >= 65 && e <= 90);

    }

    static isDecimal(e) {

        return e === 190 || e === 188 || e === 108 || e === 110;

    }

    static isDash(e) {

        return e === 109 || e === 189;

    }

}

export default KeyCodeUtils;

***LanguageUtils.js***

import messages\_vi from '../translations/vi.json';

import messages\_en from '../translations/en.json';

const flattenMessages = ((nestedMessages, prefix = '') => {

    if (nestedMessages == null) {

        return {}

    }

    return Object.keys(nestedMessages).reduce((messages, key) => {

        const value = nestedMessages[key];

        const prefixedKey = prefix ? `${prefix}.${key}` : key;

        if (typeof value === 'string') {

            Object.assign(messages, {[prefixedKey]: value})

        } else {

            Object.assign(messages, flattenMessages(value, prefixedKey))

        }

        return messages

    }, {})

});

const messages = {

    'vi': flattenMessages(messages\_vi),

    'en': flattenMessages(messages\_en),

};

export default class LanguageUtils {

    static getMessageByKey(key, lang) {

        return messages[lang][key]

    }

    static getFlattenedMessages() {

        return messages;

    }

}

***ToastUtil.js***

import React from 'react';

import { toast } from 'react-toastify';

import axios from 'axios';

import CustomToast from "../components/CustomToast";

const TYPE\_SUCCESS = 'SUCCESS';

const TYPE\_INFO = 'INFO';

const TYPE\_WARN = 'WARN';

const TYPE\_ERROR = 'ERROR';

class ToastUtil {

    static success(title, message) {

        this.show(TYPE\_SUCCESS, title, message);

    }

    static info(title, message) {

        this.show(TYPE\_INFO, title, message);

    }

    static warn(title, message) {

        this.show(TYPE\_WARN, title, message);

    }

    static error(title, message) {

        this.show(TYPE\_ERROR, title, message);

    }

    static successRaw(title, message) {

        this.show(TYPE\_SUCCESS, title, message, true);

    }

    static errorRaw(title, message, autoCloseDelay = 3000) {

        this.show(TYPE\_ERROR, title, message, true, autoCloseDelay);

    }

    static errorApi(error, title = 'common.fail-to-load-data', autoCloseDelay = 3000) {

        if (axios.isCancel(error)) {

            // Do nothing if request was cancelled

            return;

        }

        let message = null;

        let messageId = 'common.unknown-error';

        if (error.httpStatusCode >= 500) {

            messageId = 'common.internal-server-error';

        } else if (error.httpStatusCode < 500 && error.httpStatusCode >= 400) {

            if (error.httpStatusCode === 400) {

                messageId = 'common.bad-request';

            } else if (error.httpStatusCode === 403) {

                messageId = 'common.forbiden-request';

            }

        } else {

            // Request fail even server was returned a success response

            if (error.errorMessage) {

                message = error.errorMessage

            }

        }

        toast.error(<CustomToast titleId={title} message={message} messageId={messageId} time={new Date()} />, {

            position: toast.POSITION.BOTTOM\_RIGHT,

            pauseOnHover: true,

            autoClose: autoCloseDelay

        });

    }

    static show(type, title, message, rawMessage = false, autoCloseDelay = 3000) {

        const content = <CustomToast titleId={title} messageId={rawMessage ? null : message} message={rawMessage ? message : null} time={new Date()} />;

        const options = {

            position: toast.POSITION.BOTTOM\_RIGHT,

            pauseOnHover: true,

            autoClose: autoCloseDelay

        };

        switch (type) {

            case TYPE\_SUCCESS:

                toast.success(content, options);

                break;

            case TYPE\_INFO:

                toast.info(content, options);

                break;

            case TYPE\_WARN:

                toast.warn(content, options);

                break;

            case TYPE\_ERROR:

                toast.error(content, options);

                break;

            default:

                break;

        }

    }

}

export default ToastUtil;

**src/App.js/**

import React from 'react';

import { BrowserRouter } from 'react-router-dom';

import AppRoutes from './routes/AppRoutes';

import { Provider } from 'react-redux';

import store from './redux';

function App() {

    return (

        <Provider store={store}>

            <BrowserRouter>

                <AppRoutes />

            </BrowserRouter>

        </Provider>

    );

}

export default App;

**src/App.scss/**

@import "../styles/common.scss";

**src/axios.js/**

import axios from 'axios';

import \_ from 'lodash';

const instance = axios.create({

    baseURL: process.env.REACT\_APP\_BACKEND\_URL,

    withCredentials: true,

});

const createError = (httpStatusCode, statusCode, errorMessage, problems, errorCode = '') => {

    const error = new Error();

    error.httpStatusCode = httpStatusCode;

    error.statusCode = statusCode;

    error.errorMessage = errorMessage;

    error.problems = problems;

    error.errorCode = String(errorCode);

    return error;

};

export const isSuccessStatusCode = (s) => {

    return (typeof s === 'number' && s === 0) || (typeof s === 'string' && s.toUpperCase() === 'OK');

};

instance.interceptors.response.use(

    (response) => {

        const { data } = response;

        if (data.hasOwnProperty('s') && !isSuccessStatusCode(data.s) && data.hasOwnProperty('errmsg')) {

            return Promise.reject(createError(response.status, data.s, data.errmsg, null, data.errcode || ""));

        }

        if (data.hasOwnProperty('s') && data.hasOwnProperty('d')) return data.d;

        if (data.hasOwnProperty('s') && \_.keys(data).length === 1) return null;

        return data;

    },

    (error) => {

        const { response } = error;

        if (!response) return Promise.reject(error);

        const { data } = response;

        if (data?.s && data?.errmsg) {

            return Promise.reject(createError(response.status, data.s, data.errmsg));

        }

        if (data?.code && data?.message) {

            return Promise.reject(createError(response.status, data.code, data.message, data.problems));

        }

        return Promise.reject(createError(response.status));

    }

);

export default instance;

**src/index.js/**

import React from 'react';

import ReactDOM from 'react-dom';

import 'react-toastify/dist/ReactToastify.css';

import './styles/styles.scss';

import App from './App';

import \* as serviceWorker from './serviceWorker';

import IntlProviderWrapper from "./hoc/IntlProviderWrapper";

import { Provider } from 'react-redux';

import reduxStore, { persistor } from './redux';

const renderApp = () => {

    ReactDOM.render(

        <Provider store={reduxStore}>

            <IntlProviderWrapper>

                <App persistor={persistor} />

            </IntlProviderWrapper>

        </Provider>,

        document.getElementById('root')

    );

};

renderApp();

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

**src/redux.js/**

import { logger } from "redux-logger";

import thunkMiddleware from "redux-thunk";

import { routerMiddleware } from 'connected-react-router';

import { createBrowserHistory } from 'history';

import { createStore, applyMiddleware, compose } from 'redux';

import { createStateSyncMiddleware } from 'redux-state-sync';

import { persistStore } from 'redux-persist';

import createRootReducer from './store/reducers/rootReducer';

import actionTypes from './store/actions/actionTypes';

const environment = process.env.NODE\_ENV || "development";

let isDevelopment = environment === "development";

//hide redux logs

isDevelopment = false;

export const history = createBrowserHistory({ basename: process.env.REACT\_APP\_ROUTER\_BASE\_NAME });

const reduxStateSyncConfig = {

    whitelist: [

        actionTypes.APP\_START\_UP\_COMPLETE,

    ]

}

const rootReducer = createRootReducer(history);

const middleware = [

    routerMiddleware(history),

    thunkMiddleware,

    createStateSyncMiddleware(reduxStateSyncConfig),

]

if (isDevelopment) middleware.push(logger);

const composeEnhancers = (isDevelopment && window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_) ? window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ : compose;

const reduxStore = createStore(

    rootReducer,

    composeEnhancers(applyMiddleware(...middleware)),

)

export const dispatch = reduxStore.dispatch;

export const persistor = persistStore(reduxStore);

export default reduxStore;

**src/serviceWorker.js/**

// This optional code is used to register a service worker.

// register() is not called by default.

// This lets the app load faster on subsequent visits in production, and gives

// it offline capabilities. However, it also means that developers (and users)

// will only see deployed updates on subsequent visits to a page, after all the

// existing tabs open on the page have been closed, since previously cached

// resources are updated in the background.

// To learn more about the benefits of this model and instructions on how to

// opt-in, read https://bit.ly/CRA-PWA

const isLocalhost = Boolean(

  window.location.hostname === 'localhost' ||

    // [::1] is the IPv6 localhost address.

    window.location.hostname === '[::1]' ||

    // 127.0.0.1/8 is considered localhost for IPv4.

    window.location.hostname.match(

      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/

    )

);

export function register(config) {

  if (process.env.NODE\_ENV === 'production' && 'serviceWorker' in navigator) {

    // The URL constructor is available in all browsers that support SW.

    const publicUrl = new URL(process.env.PUBLIC\_URL, window.location.href);

    if (publicUrl.origin !== window.location.origin) {

      // Our service worker won't work if PUBLIC\_URL is on a different origin

      // from what our page is served on. This might happen if a CDN is used to

      // serve assets; see https://github.com/facebook/create-react-app/issues/2374

      return;

    }

    window.addEventListener('load', () => {

      const swUrl = `${process.env.PUBLIC\_URL}/service-worker.js`;

      if (isLocalhost) {

        // This is running on localhost. Let's check if a service worker still exists or not.

        checkValidServiceWorker(swUrl, config);

        // Add some additional logging to localhost, pointing developers to the

        // service worker/PWA documentation.

        navigator.serviceWorker.ready.then(() => {

          console.log(

            'This web app is being served cache-first by a service ' +

              'worker. To learn more, visit https://bit.ly/CRA-PWA'

          );

        });

      } else {

        // Is not localhost. Just register service worker

        registerValidSW(swUrl, config);

      }

    });

  }

}

function registerValidSW(swUrl, config) {

  navigator.serviceWorker

    .register(swUrl)

    .then(registration => {

      registration.onupdatefound = () => {

        const installingWorker = registration.installing;

        if (installingWorker == null) {

          return;

        }

        installingWorker.onstatechange = () => {

          if (installingWorker.state === 'installed') {

            if (navigator.serviceWorker.controller) {

              // At this point, the updated precached content has been fetched,

              // but the previous service worker will still serve the older

              // content until all client tabs are closed.

              console.log(

                'New content is available and will be used when all ' +

                  'tabs for this page are closed. See https://bit.ly/CRA-PWA.'

              );

              // Execute callback

              if (config && config.onUpdate) {

                config.onUpdate(registration);

              }

            } else {

              // At this point, everything has been precached.

              // It's the perfect time to display a

              // "Content is cached for offline use." message.

              console.log('Content is cached for offline use.');

              // Execute callback

              if (config && config.onSuccess) {

                config.onSuccess(registration);

              }

            }

          }

        };

      };

    })

    .catch(error => {

      console.error('Error during service worker registration:', error);

    });

}

function checkValidServiceWorker(swUrl, config) {

  // Check if the service worker can be found. If it can't reload the page.

  fetch(swUrl)

    .then(response => {

      // Ensure service worker exists, and that we really are getting a JS file.

      const contentType = response.headers.get('content-type');

      if (

        response.status === 404 ||

        (contentType != null && contentType.indexOf('javascript') === -1)

      ) {

        // No service worker found. Probably a different app. Reload the page.

        navigator.serviceWorker.ready.then(registration => {

          registration.unregister().then(() => {

            window.location.reload();

          });

        });

      } else {

        // Service worker found. Proceed as normal.

        registerValidSW(swUrl, config);

      }

    })

    .catch(() => {

      console.log(

        'No internet connection found. App is running in offline mode.'

      );

    });

}

export function unregister() {

  if ('serviceWorker' in navigator) {

    navigator.serviceWorker.ready.then(registration => {

      registration.unregister();

    });

  }

}

**.env/**

PORT=3000

NODE\_ENV=development

REACT\_APP\_BACKEND\_URL=http://localhost:8080

#The base URL for all locations. If your app is served from a sub-directory on your server, you'll want to set

#this to the sub-directory. A properly formatted basename should have a leading slash, but no trailing slash.

REACT\_APP\_ROUTER\_BASE\_NAME=

**.gitignore/**

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies

/node\_modules

/.pnp

.pnp.js

# testing

/coverage

# production

/build

# misc

.DS\_Store

.env.local

.env.development.local

.env.test.local

.env.production.local

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

**package.json/**

{

  "name": "nguyenlien",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@formatjs/intl-pluralrules": "^3.5.6",

    "@formatjs/intl-relativetimeformat": "^7.3.6",

    "@fortawesome/fontawesome-free-webfonts": "^1.0.9",

    "axios": "^0.21.1",

    "bootstrap": "^5.0.1",

    "connected-react-router": "^6.9.1",

    "lodash": "^4.17.21",

    "moment": "^2.29.0",

    "react": "^17.0.2",

    "react-auth-wrapper": "^1.0.0",

    "react-autosuggest": "^10.1.0",

    "react-custom-scrollbars-2": "^4.5.0",

    "react-dom": "^17.0.2",

    "react-flatpickr": "^3.10.7",

    "react-intl": "^5.20.2",

    "react-redux": "^7.2.4",

    "react-router": "^5.2.0",

    "react-router-dom": "^5.2.0",

    "react-scripts": "^4.0.3",

    "react-toastify": "^5.5.0",

    "reactstrap": "^8.9.0",

    "redux": "^4.1.0",

    "redux-auth-wrapper": "^2.1.0",

    "redux-logger": "^3.0.6",

    "redux-persist": "^5.10.0",

    "redux-state-sync": "^2.1.0",

    "redux-thunk": "^2.3.0",

    "typescript": "^4.3.2"

  },

  "scripts": {

    "start": "set NODE\_OPTIONS=--openssl-legacy-provider && react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": "react-app"

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  },

  "devDependencies": {

    "sass": "^1.89.2"

  }

}