```jsx
import React, { Fragment, useEffect, useState } from "react";
import { DatePicker, Input, Select, Row, Col } from "antd";
import Flex from "components/shared-components/Flex";
import { isEmpty } from "lodash";
import { SearchOutlined } from "@ant-design/icons";
import { MediSelect } from "components/Custom";

const FilterComponentCustom = (props) => {
  const {
    children,
    datasource,
    onSearchImmediate,
    // onClearFilter,
    renderFilterMoreView,
    renderInputSearch,
    renderDatePicker,
    placeholderSearh = "Tìm kiếm",
  } = props;

  const dateFormat = "DD/MM/YYYY";

  const [formData, setFormData] = useState({ ...datasource });
  const [keyDate, setKeyDate] = useState(Date.now());
  const [keySearchText, setKeySearchText] = useState("searchText");

  useEffect(() => {
    if(datasource.startDate===null){
      setKeyDate(Date.now())
    }
    if(datasource?.searchText==null){
      setKeySearchText(Date.now().toString())
    }
    setFormData({ ...datasource });
  }, [datasource]);
  const onItemValueChanged = (child) => {
    const handleValueChange = (value, dataField) => {
      const newData = {
        ...formData,
        [dataField]: value,
      };
      setFormData(newData);
      if (onSearchImmediate) {
        onSearchImmediate(newData, dataField);
      }
    };

    if (child.type === Select || child.type === DatePicker || child.type === MediSelect) {
      return (value) => {
        handleValueChange(value, child.props["data-field"]);
      };
    }
    // else if to-do
    else {
      return (e) => {
        let { value } = e.target;
        handleValueChange(value, child.props["data-field"]);
      };
    }
  };


  const onChangeSearch = (text) => {
    let newData;
    newData = {
      ...formData,
      searchText: text,
    };

    setFormData(newData);

    if (onSearchImmediate) {
```

```jsx
      onSearchImmediate(newData);
    }
  };

  const handleSearchChange = (e) => {
    const { value } = e.target;
    console.log(value,"value")
    if (!value) {
      onChangeSearch(value);
    }
  };

  const onDefaultItemValueChange = (field) => (e) => {
    let newData;
    if (field === "DatePicker") {
      const [startDate, endDate] = isEmpty(e)
        ? [null, null]
        : [e[0].toISOString(), e[1].toISOString()];
      newData = {
        ...formData,
        startDate,
        endDate,
      };
    } else {
      const { target } = e;
      newData = {
        ...formData,
        [field]: target.value,
      };
    }

    setFormData(newData);

    if (onSearchImmediate) {
      onSearchImmediate(newData);
    }
  };

  //Custom render children
  const renderChildren = (item) => {
    if (!item) return null;

    const { "data-field": dataField, label } = item.props;

    //Add event listener to children
    let child;

    child = React.cloneElement(item, {
      value: formData[dataField],
      className: "mb-1",
      onChange: onItemValueChanged(item),
    });

    //add custom title
    if (label && label.trim().length > 0) {
      return (
        <Fragment key={item.key}>
          <Col>
            <label className="d-block mb-1">{label}</label>
            {child}
          </Col>
        </Fragment>
      );
    }

    return (
      <Fragment key={item.key}>
        <Col>{child}</Col>
      </Fragment>
    );
  };

  return (
    <Flex>
```

```jsx
      <Row gutter={16} align="bottom">
        {renderInputSearch && (
          <Col className="mb-1">
            <Input
              type="string"
              key={keySearchText}
              data-field="searchText"
              defaultValue={formData.searchText}
              placeholder={placeholderSearh}
              allowClear
              style={{ width: "224px" }}
              prefix={<SearchOutlined />}
              onPressEnter={onDefaultItemValueChange("searchText")}
              onChange={handleSearchChange}
              // onSearch={onChangeSearch}
            />
          </Col>
        )}
        {renderDatePicker && (
          <Col className="mb-1">
            <DatePicker.RangePicker
              defaultValue={[
                formData.startDate,
                formData.endDate,
              ]}
              key={keyDate}
              format={dateFormat}
              onChange={onDefaultItemValueChange("DatePicker")}
            />
          </Col>
        )}
        {React.Children.map(children, (childItem) => renderChildren(childItem))}
        <Col className="mb-1">
          {renderFilterMoreView && renderFilterMoreView()}
        </Col>
      </Row>
    </Flex>
  );
};

export default FilterComponentCustom;
```