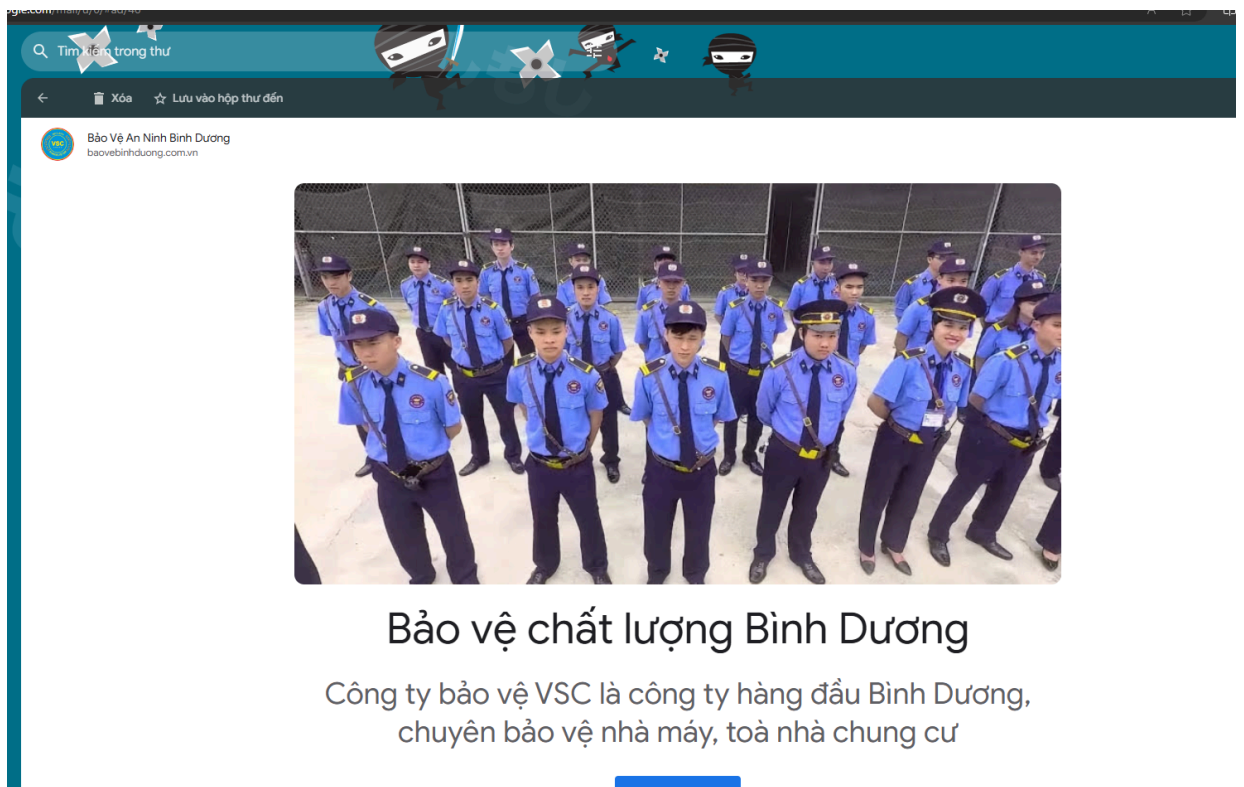


công ty mới



Bài tập 1: Tìm phần tử chính giữa của 1 danh sách liên kết đơn chỉ qua 1 lần lặp.

bài tập 2: Tìm max lớn thứ 2 qua 1 lần lặp.

Cách 1:

```

public static Integer find_max2(ListNode head) {
    if (head == null || head.next == null) {
        return null;
    }

    Integer max1 = null;
    Integer max2 = null;

    ListNode current = head;
    while (current != null) {
        int value = current.value;

        if (max1 == null || value > max1) {
            max2 = max1; // Cập nhật max2 trước khi max1 thay đổi
            max1 = value;
        } else if (max1 != null && value < max1 && (max2 == null || value > max2)) {
            max2 = value;
        }

        current = current.next;
    }

    return max2;
}

```

Cách 2:

```

public static Integer find_max22(ListNode head) {
    if (head == null || head.next == null) {
        return null;
    }

    Integer firstMax = null;
    Integer secondMax = null;

    ListNode current = head;
    while (current != null) {
        int value = current.value;

        if (firstMax == null || value > firstMax) {
            secondMax = firstMax;
            firstMax = value;
        } else if (value < firstMax && (secondMax == null || value > secondMax)) {
            secondMax = value; // Cập nhật giá trị lớn thứ hai
        }

        current = current.next;
    }

    return secondMax;
}

```

Cách 2:

```

public class vd2 {

    // Tìm phần tử chính giữa bằng cách lưu trữ các phần tử trong danh sách phụ
    public static ListNode findMiddle(ListNode head) {
        if (head == null) {
            return null; // Trả về null nếu danh sách rỗng
        }

        List<ListNode> nodeList = new ArrayList<>();
        ListNode current = head;

        // Bước 1: Lưu tất cả các phần tử vào danh sách phụ
        while (current != null) {
            nodeList.add(current);
            current = current.next;
        }

        // Bước 2: Tìm phần tử chính giữa
        int middleIndex = nodeList.size() / 2;
        return nodeList.get(middleIndex); // Trả về phần tử chính giữa
    }
}

```

Bài tập1: Tìm phần tử
chính giữa của 1 danh sách
liên kết đơn chỉ qua 1 lần lặp

Cách 1:

```

class ListNode {
    int value;
    ListNode next;

    ListNode(int value) {
        this.value = value;
        this.next = null;
    }
}

public class vd {

    public static ListNode findMiddle(ListNode head) {
        if (head == null) {
            return null; // Trả về null nếu danh sách rỗng
        }

        ListNode slow = head;
        ListNode fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;          // Di chuyển con trỏ chậm 1 bước
            fast = fast.next.next;      // Di chuyển con trỏ nhanh 2 bước
        }

        return slow; // Con trỏ chậm đang ở giữa danh sách
    }

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        // Tạo danh sách liên kết đơn ví dụ: 1 -> 2 -> 3 -> 4 -> 5
        ListNode head = new ListNode(value:1);
        head.next = new ListNode(value:2);
        head.next.next = new ListNode(value:3);
        head.next.next.next = new ListNode(value:4);
        head.next.next.next.next = new ListNode(value:5);

        ListNode middle = findMiddle(head);
        System.out.println("Middle element: " + (middle != null ? middle.value : "List is empty"));
    }
}

```



```

src > components > student > Students > Student
8   export default function Student() {
93   },
94   const [searchTerm, setSearchTerm] = useState('');
95   const [searchApi, setSearchApi] = useState('');
96   const filteredStudents = (students || []).filter(student =>
97     student.ten.toLowerCase().includes(searchTerm.toLowerCase())
98   );
99   const handle_search_api = () => {
100     dispatch(searchStudents(searchApi));
101   }
102   const [startYear, setStartYear] = useState(2000);
103   const [endYear, setEndYear] = useState(2001);
104   const handleSearchByYear = () => {
105     if(startYear&&endYear){
106       dispatch(searchStudentsByYear({ startYear, endYear }));
107     }
108   };
109   return (
110     <div className="products">
111       <Container>
112         <div className="my-3 d-flex">
113           <Input
114             type="number"
115             value={startYear}
116             onChange={(e) => setStartYear(e.target.value)}
117             className="mr-2"
118           />
119           <Input
120             type="number"
121             value={endYear}
122             onChange={(e) => setEndYear(e.target.value)}
123             className="mr-2"
124           />
125           <Button onClick={handleSearchByYear}>Search</Button>
126         </div>
127         {showMessage && (
128           <Alert color={status === 200 ? "success" : "danger"}>
129             {message}
130           </Alert>
131         )}
132         {error && (
133           <Alert color="danger">
134             <ul>
135               {error.map((error, index) => (
136                 <li key={index}>{error}</li>
137               ))}
138             </ul>
139           </Alert>
140         )}
141       </Container>
142     </div>
143   );
144 }

```

redux

```
Terminal  Help  <-->  🔍 pro1

JS App.js  JS index.js  JS studentSlice.js  JS StudentPage.js  JS Student.js  JS Student2.js  # App.css  JS store.js  <img alt="index.html icon" data-bbox="895 105 910 115"/> index.html

src > redux > JS studentSlice.js > studentSlice > extraReducers

35 export const editStudent= createAsyncThunk('student/editProduct', async ({id,student},thunkAPI) => {
36   const url= BASE_URL+'/student/${id}';
37   try {
38     console.log(student)
39     const response = await axios.put(url,student);
40     return response.data; // Trả về dữ liệu từ phản hồi
41   } catch (error) {
42     return thunkAPI.rejectWithValue(error.response.data); // Trả về lỗi nếu có
43   }
44 });
45 export const searchStudents = createAsyncThunk('student/searchStudents', async (searchTerm, thunkAPI) => {
46   const url = `${BASE_URL}/student/search3`;
47   try {
48     const response = await axios.get(url, {
49       params: { "name": searchTerm },
50     });
51     return response.data;
52   } catch (error) {
53     return thunkAPI.rejectWithValue(error.response.data);
54   }
55 });
56 export const searchStudentsByYear = createAsyncThunk('student/searchStudentsByYear',async ({ startYear, endYear }, thunkAPI) => {
57   console.log(startYear)
58   const url = `${BASE_URL}/student/search4?startYear=${startYear}&endYear=${endYear}`;
59   try {
60     const response = await axios.get(url);
61     return response.data; // Trả về dữ liệu từ phản hồi
62   } catch (error) {
63     return thunkAPI.rejectWithValue(error.response.data); // Trả về lỗi nếu có
64   }
65 }
66 );
67 export const searchStudentsXepLoai = createAsyncThunk('student/searchStudentsXepLoai', async (xeploai, thunkAPI) => {
68   const url = `${BASE_URL}/student/searchXepLoai`;
69   try {
70     const response = await axios.get(url, {
71       params: { "name": xeploai },
72     });
73     return response.data;
74   } catch (error) {
75     return thunkAPI.rejectWithValue(error.response.data);
76   }
77 });
78 const studentSlice = createSlice({
```

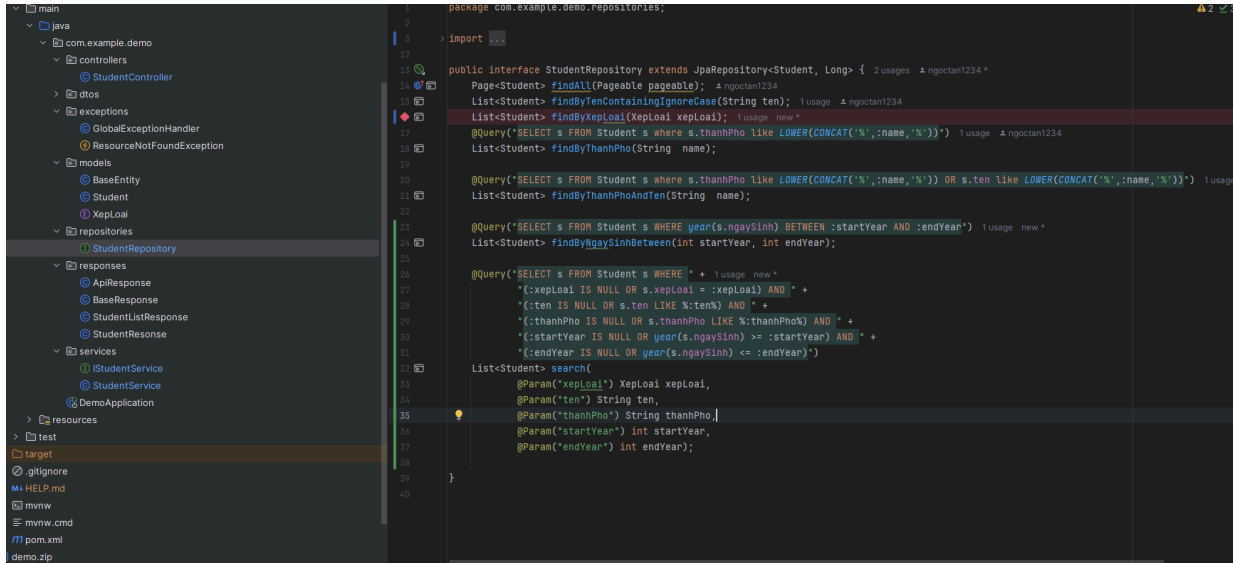


```

78  const studentsSlice = createSlice({
94    extraReducers: (builder) => {
121      .addCase(editStudent.fulfilled, (state, action) => {
122        state.status=action.payload.status
123        state.message=action.payload.message
124        state.students = state.students.map(student =>
125          student.id === action.payload.data.id ? action.payload.data : student
126        );
127      })
128      .addCase(editStudent.rejected, (state, action) => {
129        state.status=action.payload.status
130        state.message=action.payload.message
131        state.error=action.payload.data
132      })
133      .addCase(searchStudents.fulfilled, (state, action) => {
134        state.students = action.payload.data;
135        state.status=action.payload.status
136      })
137      .addCase(searchStudents.rejected, (state, action) => {
138        state.status=action.payload.status
139        state.message=action.payload.message
140        state.error = action.payload.data;
141      })
142      .addCase(searchStudentsByYear.fulfilled, (state, action) => {
143        state.students = action.payload.data;
144        state.status=action.payload.status
145      })
146      .addCase(searchStudentsByYear.rejected, (state, action) => {
147        state.status=action.payload.status
148        state.message=action.payload.message
149        state.error = action.payload.data;
150      })
151      .addCase(searchStudentsXepLoai.fulfilled, (state, action) => {
152        state.students = action.payload.data;
153        state.status=action.payload.status
154      })
155      .addCase(searchStudentsXepLoai.rejected, (state, action) => {
156        state.status=action.payload.status
157        state.message=action.payload.message
158        state.error = action.payload.data;
159      })
160    };
161  },
162  });

```


Search Sương sương



This screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer shows a package structure for 'com.example.demo' with sub-packages like controllers, dtos, exceptions, models, repositories, responses, and services. The 'repositories' package is selected, and 'StudentRepository' is highlighted. The code editor displays the 'StudentRepository' interface, which extends 'JpaRepository<Student, Long>'. It includes several methods for finding students, such as 'findAll', 'findByName', 'findByTen', 'findByXepLoai', 'findByThanhPho', 'findByThanhPhoAndTen', 'findByNgaySinhBetween', and 'search'. The 'search' method is highlighted with a yellow cursor.

```
package com.example.demo.repositories;

import java.util.List;
import java.util.Optional;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

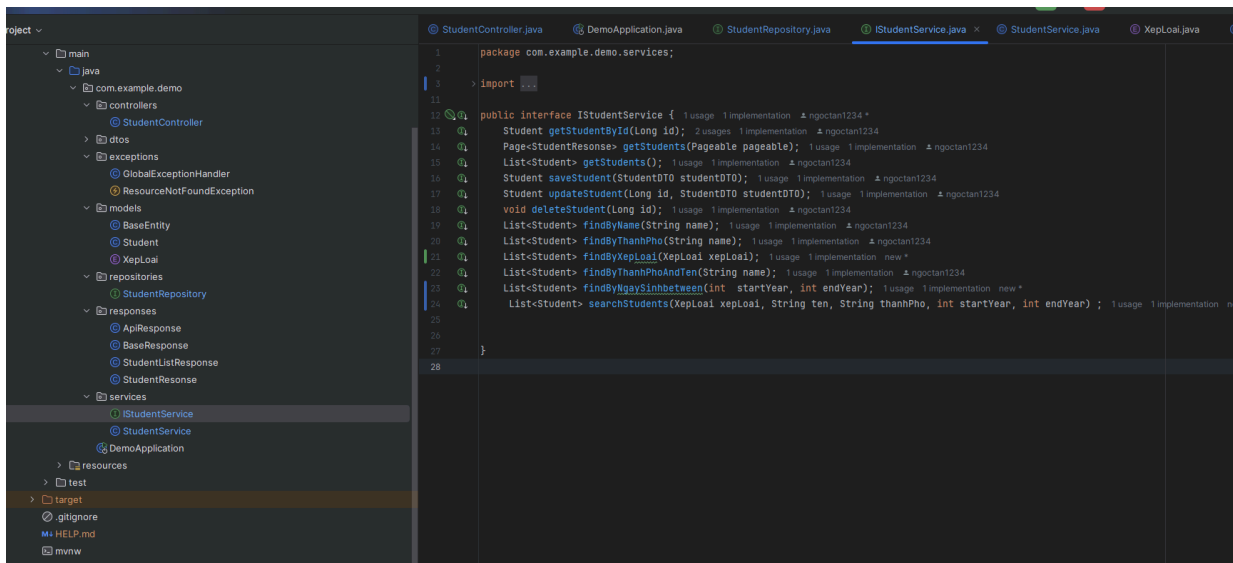
public interface StudentRepository extends JpaRepository<Student, Long> {

    Page<Student> findAll(Pageable pageable);
    List<Student> findByName(String name);
    List<Student> findByTenContainingIgnoreCase(String ten);
    List<Student> findByXepLoai(XepLoai xepLoai);
    List<Student> findByThanhPhoLike LOWER(CONCAT('%',:name,'%')));
    List<Student> findByThanhPhoAndTen(String name);

    @Query("SELECT s FROM Student s WHERE s.thanhPho LIKE LOWER(CONCAT('%',:name,'%')) OR s.ten LIKE LOWER(CONCAT('%',:name,'%'))")
    List<Student> findByThanhPhoAndTen(String name);

    @Query("SELECT s FROM Student s WHERE year(s.ngaySinh) BETWEEN :startYear AND :endYear")
    List<Student> findByNgaySinhBetween(int startYear, int endYear);

    @Query("SELECT s FROM Student s WHERE (:xepLoai IS NULL OR s.xepLoai = :xepLoai) AND (:ten IS NULL OR s.ten LIKE %:ten%) AND (:thanhPho IS NULL OR s.thanhPho LIKE %:thanhPho%) AND (:startYear IS NULL OR year(s.ngaySinh) >= :startYear) AND (:endYear IS NULL OR year(s.ngaySinh) <= :endYear)")
    List<Student> search(
        @Param("xepLoai") XepLoai xepLoai,
        @Param("ten") String ten,
        @Param("thanhPho") String thanhPho,
        @Param("startYear") int startYear,
        @Param("endYear") int endYear);
}
```



This screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer shows the same package structure as the first screenshot. The 'services' package is selected, and 'IstudentService' is highlighted. The code editor displays the 'IstudentService' interface, which includes methods for getting students by ID, getting students by pageable, saving students, updating students, deleting students, and finding students by name, ThanhPho, XepLoai, ThanhPhoAndTen, and NgaySinhBetween. The 'searchStudents' method is highlighted with a yellow cursor.

```
package com.example.demo.services;

import java.util.List;
import java.util.Optional;
import org.springframework.data.domain.Pageable;

public interface IstudentService {

    Student getStudentById(Long id);
    Page<StudentResponse> getStudents(Pageable pageable);
    List<Student> getStudents();
    Student saveStudent(StudentDTO studentDTO);
    Student updateStudent(Long id, StudentDTO studentDTO);
    void deleteStudent(Long id);
    List<Student> findByName(String name);
    List<Student> findByThanhPho(String thanhPho);
    List<Student> findByXepLoai(XepLoai xepLoai);
    List<Student> findByThanhPhoAndTen(String name);
    List<Student> findByNgaySinhBetween(int startYear, int endYear);
    List<Student> searchStudents(XepLoai xepLoai, String ten, String thanhPho, int startYear, int endYear);
}
```

```
17 public class StudentService implements IStudentService {
48     public Student updateStudent(Long id, StudentDTO studentDTO) {
52         studentExisting.setXepLoai(XepLoai.fromTen(studentDTO.getXepLoai()));
53         studentExisting.setNgaySinh(studentDTO.getNgaySinh());
54         return studentRepository.save(studentExisting);
55     }
56
57     @Override @usage 1 ngocntan1234
58     public void deleteStudent(Long id) { studentRepository.deleteById(id); }
59
60     @Override @usage 1 ngocntan1234
61     public List<Student> findByName(String name) { return studentRepository.findByTenContainingIgnoreCase(name); }
62
63     @Override @usage 1 ngocntan1234
64     public List<Student> findByThanhPho(String name) { return studentRepository.findByThanhPho(name); }
65
66     @Override @usage new *
67     public List<Student> findByXepLoai(XepLoai xepLoai) {
68         return studentRepository.findByXepLoai(xepLoai);
69     }
70
71     @Override @usage 1 ngocntan1234
72     public List<Student> findByThanhPhoAndTen(String name) { return studentRepository.findByThanhPhoAndTen(name); }
73
74     @Override @usage new *
75     public List<Student> findByNgaySinhBetween(int startYear, int endYear) {
76         return studentRepository.findByNgaySinhBetween(startYear, endYear);
77     }
78
79     @Override @usage new *
80     public List<Student> searchStudents(XepLoai xepLoai, String ten, String thanhPho, int startYear, int endYear) {
81         return studentRepository.search(xepLoai, ten, thanhPho, startYear, endYear);
82     }
83 }
84
85
86
87
88
89
90
91
```

```
30 public class StudentController {
148     public ResponseEntity<ApiResponse> search3(@RequestParam String name) {
149         return ResponseEntity.ok(apiResponse);
150     }
151
152     @GetMapping("/search4") new *
153     public ResponseEntity<ApiResponse> search3(@RequestParam int startYear, int endYear) {
154         ApiResponse apiResponse = ApiResponse.builder()
155             .data(studentService.findByNgaySinhBetween(startYear, endYear))
156             .message("Search successfully")
157             .status(HttpStatus.OK.value())
158             .build();
159         return ResponseEntity.ok(apiResponse);
160     }
161
162     @GetMapping("/searchXepLoai") new *
163     public ResponseEntity<ApiResponse> searchXepLoai(@RequestParam("xepLoai") String xepLoaiStr) {
164         ApiResponse apiResponse = ApiResponse.builder()
165             .data(studentService.findByXepLoai(XepLoai.fromTen(xepLoaiStr)))
166             .message("Search successfully")
167             .status(HttpStatus.OK.value())
168             .build();
169         return ResponseEntity.ok(apiResponse);
170     }
171
172     @GetMapping("/search") new *
173     public ResponseEntity<ApiResponse> searchStudents(
174         @RequestParam(value = "xepLoai", required = false) String xepLoai,
175         @RequestParam(value = "ten", required = false) String ten,
176         @RequestParam(value = "thanhPho", required = false) String thanhPho,
177         @RequestParam(value = "startYear", required = false) int startYear,
178         @RequestParam(value = "endYear", required = false) int endYear) {
179         ApiResponse apiResponse = ApiResponse.builder()
180             .data(studentService.searchStudents(XepLoai.fromTen(xepLoai), ten, thanhPho, startYear, endYear))
181             .message("Search successfully")
182             .status(HttpStatus.OK.value())
183             .build();
184         return ResponseEntity.ok(apiResponse);
185     }
186 }
187
```

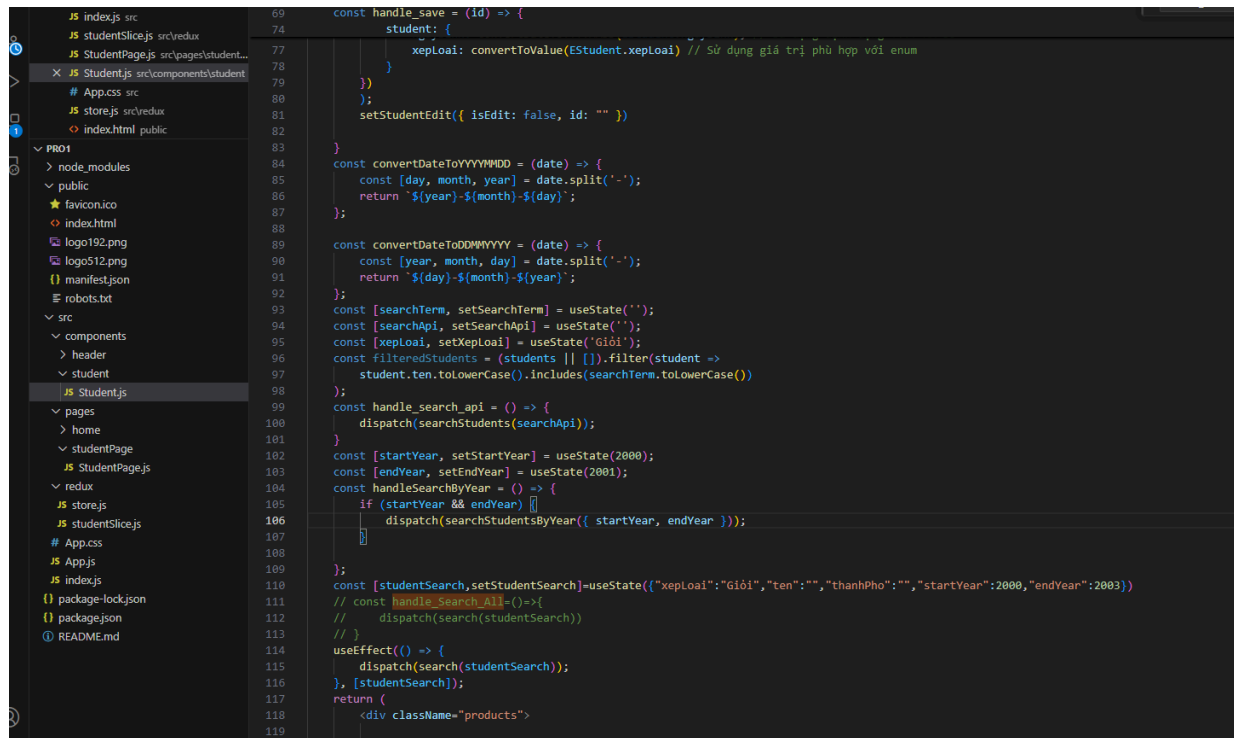
Search frontend

Reducer

```
54 }
55 });
56 export const searchStudentsByYear = createAsyncThunk('student/searchStudentsByYear', async ({ startYear, endYear }, thunkAPI) => {
57   console.log(startYear)
58   const url = `${BASE_URL}/student/search4?startYear=${startYear}&endYear=${endYear}`;
59   try {
60     const response = await axios.get(url);
61     return response.data; // Trả về dữ liệu từ phản hồi
62   } catch (error) {
63     return thunkAPI.rejectWithValue(error.response.data); // Trả về lỗi nếu có
64   }
65 });
66
67 export const searchStudentsXepLoai = createAsyncThunk('student/searchStudentsXepLoai', async (xepLoai, thunkAPI) => {
68   const url = `${BASE_URL}/student/searchXepLoai`;
69   try {
70     const response = await axios.get(url, {
71       params: { 'xepLoai': xepLoai },
72     });
73     return response.data;
74   } catch (error) {
75     return thunkAPI.rejectWithValue(error.response.data);
76   }
77 });
78
79 export const search = createAsyncThunk('student/search', async ({xepLoai,ten,thanhPho,startYear,endYear}, thunkAPI) => {
80   const url = `${BASE_URL}/student/search?xepLoai=${xepLoai}&ten=${ten}&thanhPho=${thanhPho}&startYear=${startYear}&endYear=${endYear}`;
81   try {
82     const response = await axios.get(url);
83     return response.data;
84   } catch (error) {
85     return thunkAPI.rejectWithValue(error.response.data);
86   }
87 });
88
89 const studentSlice = createSlice({
90   name: 'student',
91   initialState: {
92     status: 'idle',
93     error: null,
94     students: null,
95     totalPages: 10,
96     message: '',
97   },
98   reducers: {
99     resetStatusAndMessage: (state) => {
```

```
src > redux > JS studentSlice.js > [0] studentSlice > extraReducers > addCase() callback
87 const studentSlice = createSlice({
103   extraReducers: (builder) => {
104     //
137     .addCase(editStudent.rejected, (state, action) => {
138       state.status=action.payload.status
139       state.message=action.payload.message
140       state.error=action.payload.data
141     })
142     .addCase(searchStudents.fulfilled, (state, action) => {
143       state.students = action.payload.data;
144       state.status=action.payload.status
145     })
146     .addCase(searchStudents.rejected, (state, action) => {
147       state.status=action.payload.status
148       state.message=action.payload.message
149       state.error = action.payload.data;
150     })
151     .addCase(searchStudentsByYear.fulfilled, (state, action) => {
152       state.students = action.payload.data;
153       state.status=action.payload.status
154     })
155     .addCase(searchStudentsByYear.rejected, (state, action) => {
156       state.status=action.payload.status
157       state.message=action.payload.message
158       state.error = action.payload.data;
159     })
160     .addCase(searchStudentsXepLoai.fulfilled, (state, action) => {
161       state.students = action.payload.data;
162       state.status=action.payload.status
163     })
164     .addCase(searchStudentsXepLoai.rejected, (state, action) => {
165       state.status=action.payload.status
166       state.message=action.payload.message
167       state.error = action.payload.data;
168     })
169     .addCase(search.fulfilled, (state, action) => {
170       state.students = action.payload.data;
171       state.status=action.payload.status
172     })
173     .addCase(search.rejected, (state, action) => {
174       state.status=action.payload.status
175       state.message=action.payload.message
176       state.error = action.payload.data;
177     })
178   },
179 };
180
181
182 export const { resetStatusAndMessage } = studentSlice.actions;
```

giao diện



```
69 const handle_save = (id) => {
74   student: {
77     xepLoai: convertToValue(ESudent.xepLoai) // Sử dụng giá trị phù hợp với enum
78   }
79 }
80 };
81 setStudentEdit({ isEdit: false, id: "" })
82
83
84 const convertDateToYYYYMMDD = (date) => {
85   const [day, month, year] = date.split('-');
86   return `${year}-${month}-${day}`;
87 };
88
89 const convertDateToDDMMYYYY = (date) => {
90   const [year, month, day] = date.split('-');
91   return `${day}-${month}-${year}`;
92 };
93
94 const [searchTerm, setSearchTerm] = useState('');
95 const [searchApi, setSearchApi] = useState('');
96 const [xepLoai, setXepLoai] = useState('Giỏi');
97 const filteredStudents = (students || []).filter(student =>
98   student.ten.toLowerCase().includes(searchTerm.toLowerCase())
99 );
100
101 const handle_search_api = () => {
102   dispatch(searchStudents(searchApi));
103 }
104
105 const [startYear, setStartYear] = useState(2000);
106 const [endYear, setEndYear] = useState(2001);
107 const handleSearchByYear = () => {
108   if (startYear && endYear) {
109     dispatch(searchStudentsByYear({ startYear, endYear }));
110   }
111 };
112
113 const [studentSearch, setStudentSearch] = useState({ "xepLoai": "Giỏi", "ten": "", "thanhPho": "", "startYear": 2000, "endYear": 2003 })
114 // const handle_Search_All = () => {
115 //   dispatch(search(studentSearch))
116 // }
117
118 useEffect(() => {
119   dispatch(search(studentSearch));
120 }, [studentSearch]);
121
122 return (
123   <div className="products">
```

```
<div className="searchAll">
  <div>
    <Input
      id="xepLoai"
      name="xepLoai"
      type="select"
      value={studentSearch.xepLoai}
      onChange={(e) => {
        setStudentSearch({...studentSearch, "xepLoai": convertToValue(e.target.value)})
      }}
    />
    <option>Giỏi</option>
    <option>Khá</option>
    <option>Trung bình</option>
    <option>Yếu</option>
  />
</div>
<div className="my-3 d-flex">
  <Input
    type="number"
    value={studentSearch.startYear}
    onChange={(e) => {
      setStudentSearch({...studentSearch, "startYear": e.target.value})
    }}
    className="mr-2"
  />
  <Input
    type="number"
    value={studentSearch.endYear}
    onChange={(e) => {
      setStudentSearch({...studentSearch, "endYear": e.target.value})
    }}
    className="mr-2"
  />
  <Button >Search</Button>
</div>
<Input type="text" placeholder="Search API" className="my-3"
value={studentSearch.ten}
onChange={(e) => {
  setStudentSearch({...studentSearch, "ten": e.target.value})
}} />
</div>
```

