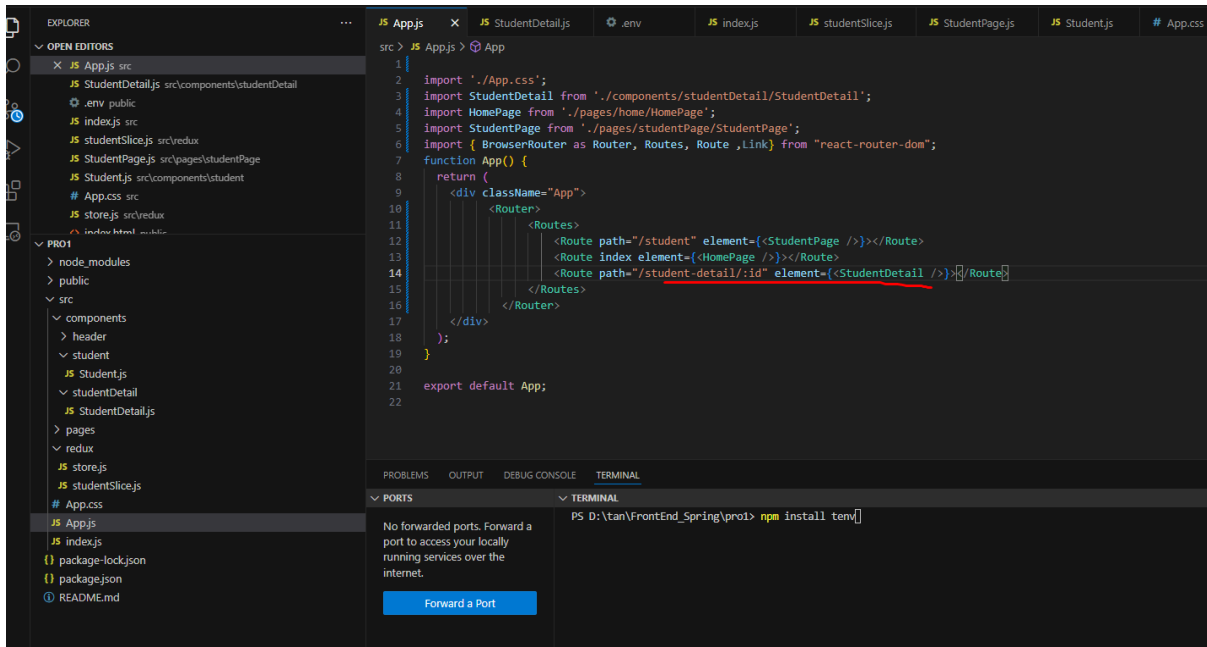


```
254     Files.copy(file.getInputStream(),destination,StandardCopyOption.REPLACE_EXISTING);
255     return uniqueFileName;
256 }
257
258 @GetMapping("/images/{imageName}") new *
259 public ResponseEntity<?> viewImage(@PathVariable String imageName) {
260     try {
261         java.nio.file.Path imagePath = Paths.get("upload/"+imageName);
262         UrlResource resource = new UrlResource(imagePath.toUri());
263
264         if (resource.exists()) {
265             return ResponseEntity.ok()
266                 .contentType(MediaType.IMAGE_JPEG)
267                 .body(resource);
268         } else {
269             // logger.info(imageName + " not found");
270             return ResponseEntity.ok()
271                 .contentType(MediaType.IMAGE_JPEG)
272                 .body(new UrlResource(Paths.get("uploads/notfound.jpeg").toUri()));
273             //return ResponseEntity.notFound().build();
274         }
275     } catch (Exception e) {
276         //logger.error("Error occurred while retrieving image: " + e.getMessage());
277         return ResponseEntity.notFound().build();
278     }
279 }
280 }
```

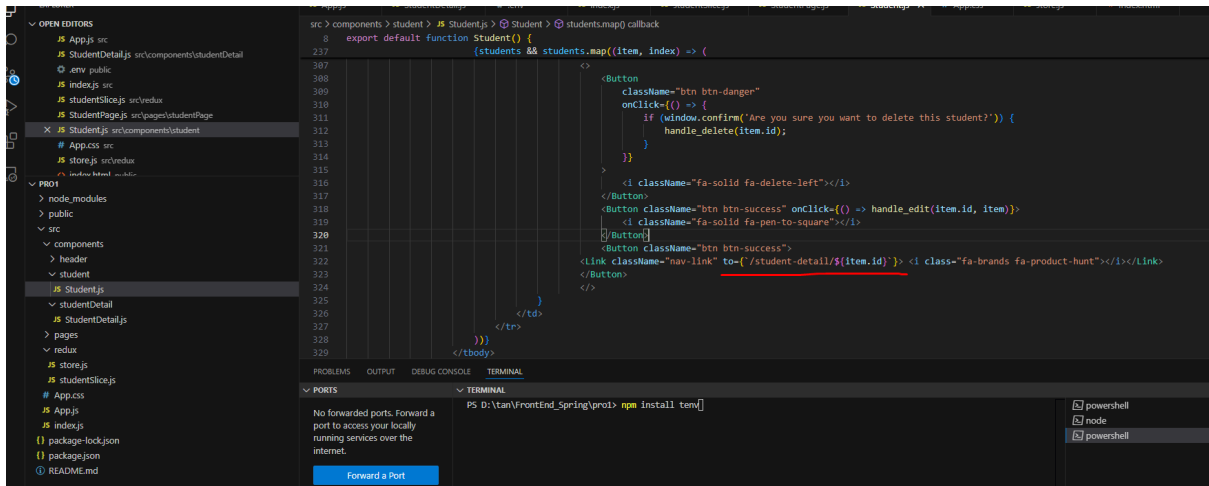
App.js



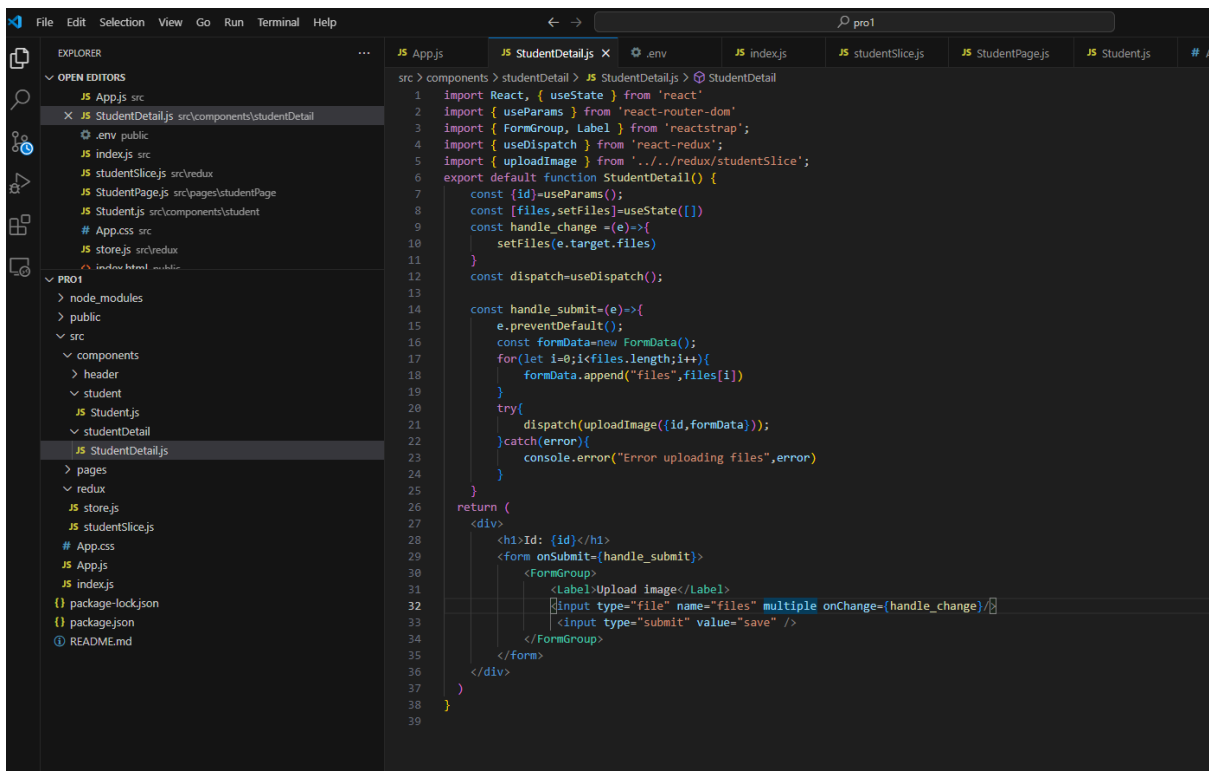
The screenshot shows the VS Code interface with the `App.js` file open. The Explorer sidebar on the left shows the project structure, including `src`, `components`, `pages`, `redux`, and `store`. The `App.js` file is selected in the Explorer. The main editor shows the `App.js` code, which imports `App.css`, `StudentDetail`, `HomePage`, `StudentPage`, and `BrowserRouter` from `react-router-dom`. The `App` function returns a `Router` component with three routes: `/student` (rendering `StudentPage`), `/` (rendering `HomePage`), and `/student-detail/:id` (rendering `StudentDetail`). The `App` component is exported as the default export.

```
1 import './App.css';
2 import StudentDetail from './components/studentDetail/StudentDetail';
3 import HomePage from './pages/home/HomePage';
4 import StudentPage from './pages/studentPage/StudentPage';
5 import { BrowserRouter as Router, Routes, Route ,Link} from "react-router-dom";
6 function App() {
7     return (
8         <div className="App">
9             <Router>
10                 <Routes>
11                     <Route path="/student" element={<StudentPage />}></Route>
12                     <Route index element={<HomePage />}></Route>
13                     <Route path="/student-detail/:id" element={<StudentDetail />}></Route>
14                 </Routes>
15             </Router>
16         </div>
17     );
18 }
19 export default App;
```

Student.js



StudentDetail



SliceStudent

```
85 }
86 });
87
88 export const uploadImage = createAsyncThunk('student/uploadImage', async ({id,formData}, thunkAPI) => {
89   const url = `${BASE_URL}/student/uploads/${id}`;
90   try {
91     const response = await axios.post(url,formData,{
92       headers:{
93         'Content-Type':'multipart/form-data'
94       }
95     });
96     return response.data;
97   } catch (error) {
98     return thunkAPI.rejectWithValue(error.response.data);
99   }
100 });
101
102 const studentSlice = createSlice({
103   name: 'student',
104   initialState: {
105     status: 'idle',
106     error: null,
107     students:null,
108     totalPages:10,
109     message:"",
110   },
111   reducers: {
112     resetStatusAndMessage: (state) => {
113       state.status = null;
114       state.message = "";
115       state.error=null;
116     },
117   },
118   extraReducers: (builder) => {
119     builder
120       .addCase(getAll.fulfilled, (state, action) => {
121         state.students = action.payload.data.studentResonseList
```

SÁNG ĐI LÀM

CHIỀU VỀ



Chức năng hiển thị hình ảnh
reducer

```
src > redux > JS studentSlicejs > @ studentSlice > @ extraReducers
180 export const uploadImage = createAsyncThunk( student/uploadImage , async ({id,formData}, thunkAPI) => {
181 });
182 export const getAllStudentDetail = createAsyncThunk('student/getAllStudentDetail', async (id,thunkAPI) => {
183   const url = `${BASE_URL}/student/getAllImage/${id}`;
184   try {
185     const response = await axios.get(url);
186     return response.data; // Trả về dữ liệu từ phản hồi
187   } catch (error) {
188     return thunkAPI.rejectWithValue(error.response.data); // Trả về lỗi nếu có
189   }
190 });
191
192 const studentSlice = createSlice({
193   name: 'student',
194   initialState: {
195     status: 'idle',
196     error: null,
197     students:null,
198     studentDetails:null,
199     totalPages:0,
200     message:"",
201   },
202   reducers: {
203     resetStatusAndMessage: (state) => {
204       state.status = null;
205       state.message = "";
206       state.error=null;
207     },
208   },
209   extraReducers: (builder) => {
210     builder
211       .addCase(getAllStudentDetail.fulfilled, (state, action) => {
212         state.status=action.payload.status
213         state.message=action.payload.message
214         state.studentDetails = action.payload.data;
215       })
216       .addCase(getAllStudentDetail.rejected, (state, action) => {
217         state.status=action.payload.status
218         state.message=action.payload.message
219         state.error=action.payload.data
220       })
221   }
222 })
223
224 .addCase(getAll.fulfilled, (state, action) => {
```

Detail student

```
src > components > studentDetail > JS StudentDetailjs > @ StudentDetail > @ handle_submit
1
2 import React, { useEffect, useState } from 'react'
3 import { useParams } from 'react-router-dom'
4 import { Alert, Button, FormGroup, Label, Table } from 'reactstrap';
5 import { useDispatch, useSelector } from 'react-redux';
6 import { getAllStudentDetail, resetStatusAndMessage, uploadImage } from '../redux/studentSlice';
7 import axios from 'axios';
8 export default function StudentDetail() {
9   const {id}=useParams();
10   const [files,setFiles]=useState([])
11   const { studentDetails ,message, error,status } = useSelector((state) => state.student);
12   const [images, setImages] = useState([]);
13   const handle_change =(e)=>{
14     setFiles(e.target.files)
15   }
16   const [showMessage, setShowMessage] = useState(false); //
17   const dispatch=useDispatch();
18
19   const handle_submit=(e)=>{
20     e.preventDefault();
21     const formData=new FormData();
22     for(let i=0;i<files.length;i++){
23       formData.append("files",files[i])
24     }
25     try{
26       dispatch(uploadImage({id,formData}));
27     }catch(error){
28       console.error("Error uploading files",error)
29     }
30   }
31   const fetchImage = async (imageUrl) => {
32     try {
33       const response = await axios.get('http://localhost:8080/api/v1/admin/student/images/${imageUrl}', {
34         responseType: 'blob' // Đảm bảo phản hồi trả về là Blob
35       });
36       const imageObjectURL = URL.createObjectURL(response.data);
37       setImages(prev => ({ ...prev, [imageUrl]: imageObjectURL }));
38     } catch (error) {
39       console.error("Error fetching image", error);
40     }
41   };
42   useEffect(() => {
43     if (studentDetails) {
44       studentDetails.forEach(item => {
45         fetchImage(item.imageUrl);
```

```
Terminal Help  ← →  pro1

JS App.js  JS StudentDetails.js  {} package.json  .env  JS index.js  JS studentSlice.js  JS StudentPage.js  JS Student.js

src > components > studentDetail > JS StudentDetails.js > StudentDetail > handle_submit
8  export default function StudentDetail() {
19  const handle_submit=(e)=>{
21    const formData=new FormData();
22    for(let i=0;i<files.length;i++){
23      formData.append("files",files[i])
24    }
25    try{
26      dispatch(uploadImage({id,formData}));
27    }catch(error){
28      console.error("Error uploading files",error)
29    }
30  }
31  const fetchImage = async (imageUrl) => {
32    try {
33      const response = await axios.get(`http://localhost:8080/api/v1/admin/student/images/${imageUrl}`, {
34        responseType: 'blob' // Đảm bảo phản hồi trả về là Blob
35      });
36      const imageObjectURL = URL.createObjectURL(response.data);
37      setImages(prev => ({ ...prev, [imageUrl]: imageObjectURL }));
38    } catch (error) {
39      console.error("Error fetching image", error);
40    }
41  };
42  useEffect(() => {
43    if (studentDetails) {
44      studentDetails.forEach(item => {
45        fetchImage(item.imageUrl);
46      });
47    }
48  }, [studentDetails,dispatch]);
49  useEffect(() => {
50    dispatch(getAllStudentDetail(id));
51  }, [dispatch, id]);
52
53  useEffect(() => {
54    if (status && message) {
55      setShowMessage(true);
56
57      const timer = setTimeout(() => {
58        setShowMessage(false);
59        dispatch(resetStatusAndMessage()); // Reset status and message
60      }, 2000);
61
62      return () => clearTimeout(timer);
63    }
64  }, [status, message, dispatch]);
65
```

```

8   export default function StudentDetail() {
68     {showMessage && (
69       <Alert color={status === 200 ? "success" : "danger"}>
70         {message}
71       </Alert>
72     )}
73     {error && (
74       <Alert color="danger">
75         <ul>
76           {error.map((error, index) => (
77             <li key={index}>{error}</li>
78           ))}
79         </ul>
80       </Alert>
81     )}
82     <form onSubmit={handle_submit}>
83       <FormGroup>
84         <Label>Upload image</Label>
85         <input type="file" name="files" multiple onChange={handle_change}/>
86         <input type="submit" value="save" />
87       </FormGroup>
88     </form>
89     <Table hover>
90       <thead>
91         <tr>
92           <th>#</th>
93           <th>ID</th>
94           <th>Image</th>
95           <th>Delete</th>
96         </tr>
97       </thead>
98       <tbody>
99         {studentDetails && studentDetails.map((item, index) => (
100           <tr key={index}>
101             <th scope="row">{index + 1}</th>
102             <td>{item.id}</td>
103             <td>
104               <img src={images[item.imageUrl]} alt="Product" style={{ width: '100px', height: '100px', objectFit: 'cover' }} />
105             </td>
106             <td>
107               <Button className="btn btn-danger">
108                 <i className="fa-solid fa-delete-left"></i>
109               </Button>
110             </td>
111           </tr>
112         ))}

```

Giải thích 1:

- **Blob** là một loại đối tượng được sử dụng để đại diện cho dữ liệu nhị phân, chẳng hạn như hình ảnh, video hoặc các loại tệp khác. Từ viết tắt của **Binary Large Object**, **Blob** có thể chứa dữ liệu từ các nguồn như phản hồi từ một máy chủ web.
- **Blob** rất hữu ích khi bạn cần xử lý dữ liệu nhị phân từ các phản hồi HTTP.
- **Blob** được sử dụng để lưu trữ hình ảnh mà bạn nhận được từ máy chủ.

Giải thích 2:

- **Sử Dụng images Như Một Mảng:**
- `const [images,setImages]=useState([])`

=> nhưng cập nhật nó như một đối tượng
(setImages([...images, imageObjectURL])).
=> do **Chạy Hàm fetchImage Nhiều Lần**: Nếu
fetchImage được gọi nhiều lần, việc cập nhật trạng thái
bằng cách sử dụng setImages([...images,
imageObjectURL]) : do đây là hàm xử lý bất đồng bộ, có
thể không làm việc đúng cách vì bạn đang phụ thuộc vào giá
trị cũ của images để thêm một phần tử mới, và không phải
lúc nào cũng đồng bộ.

```
export default function StudentDetail() {
  const {id}=useParams();
  const [files,setFiles]=useState([])
  const { studentDetails ,message, error,status } = useSelector((state) => state.student);
  const [images, setImages] = useState({});
  const [arr,setArr]=useState()
  const handle_change =(e)=>{
    setFiles(e.target.files)
  }
  const [showMessage, setShowMessage] = useState(false); //
  const dispatch=useDispatch();

  const handle_submit=async(e)=>{
    e.preventDefault();
    const formData=new FormData();
    for(let i=0;i<files.length;i++){
      formData.append("files",files[i])
    }
    try{
      // dispatch(uploadImage({id,formData}));
      await dispatch(uploadImage({ id, formData })).unwrap();
      // Gọi getAllStudentDetail sau khi upload thành công
      dispatch(getAllStudentDetail(id));
    }catch(error){
      console.error("Error uploading files",error)
    }
  }

  const fetchImage = async (imageUrl) => {
    try {
      const response = await axios.get(`http://localhost:8080/api/v1/admin/student/images/${imageUrl}`, {
        responseType: 'blob' // Đảm bảo phản hồi trả về là Blob
      });
      const imageObjectURL = URL.createObjectURL(response.data);
    }
  }
}
```

Lộ trình spring boot: https://github.com/ngoctan1234/student_spring_boot_api.git

Lộ trình reactjs: https://github.com/ngoctan1234/student_Reactjs_API_spring_boot.git
npm i

Cách cài reactjs: npm i @reduxjs/toolkit react-redux axios
bootstrap react-paginate react-router-dom react-toastify
reactstrap