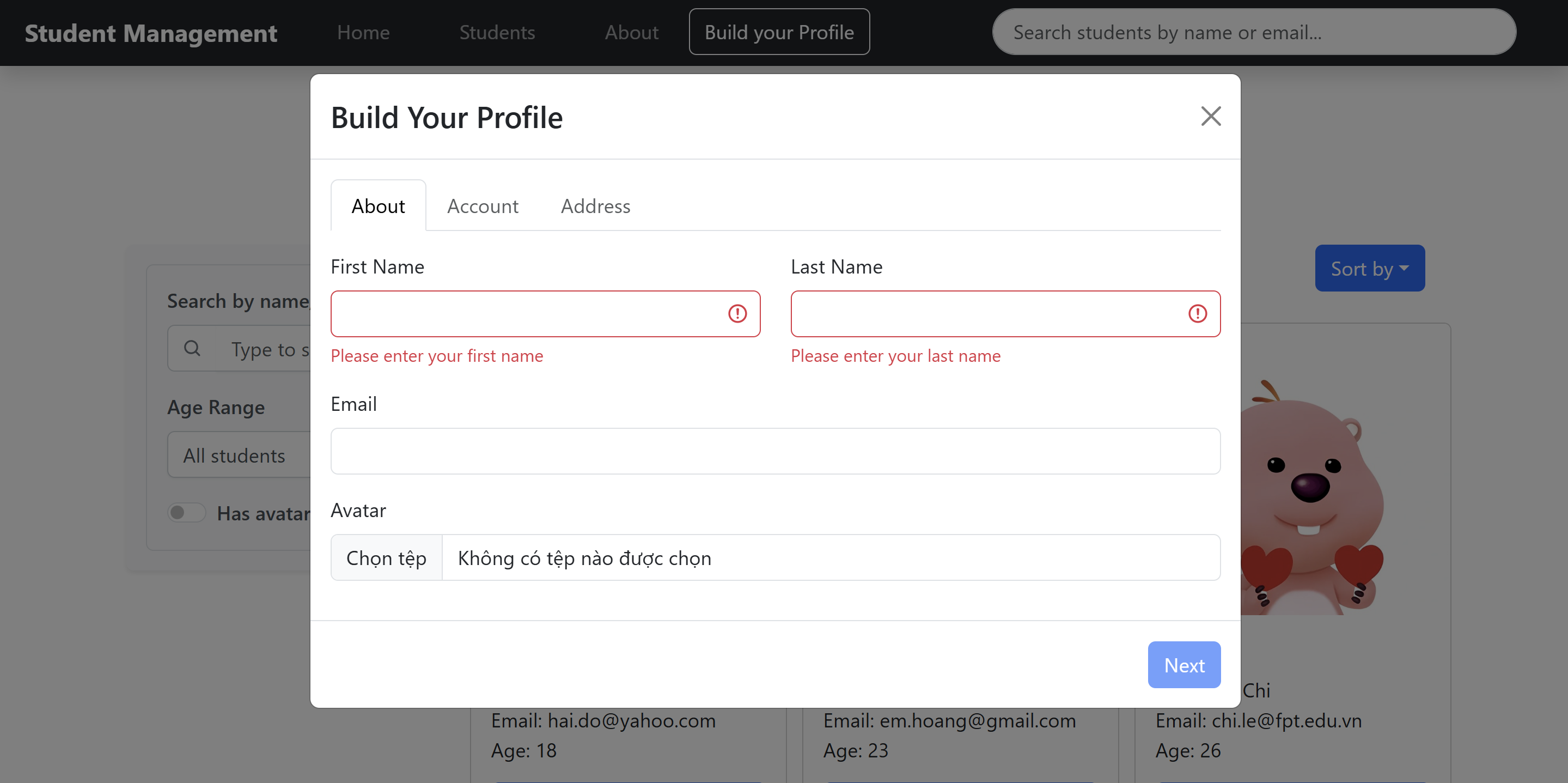
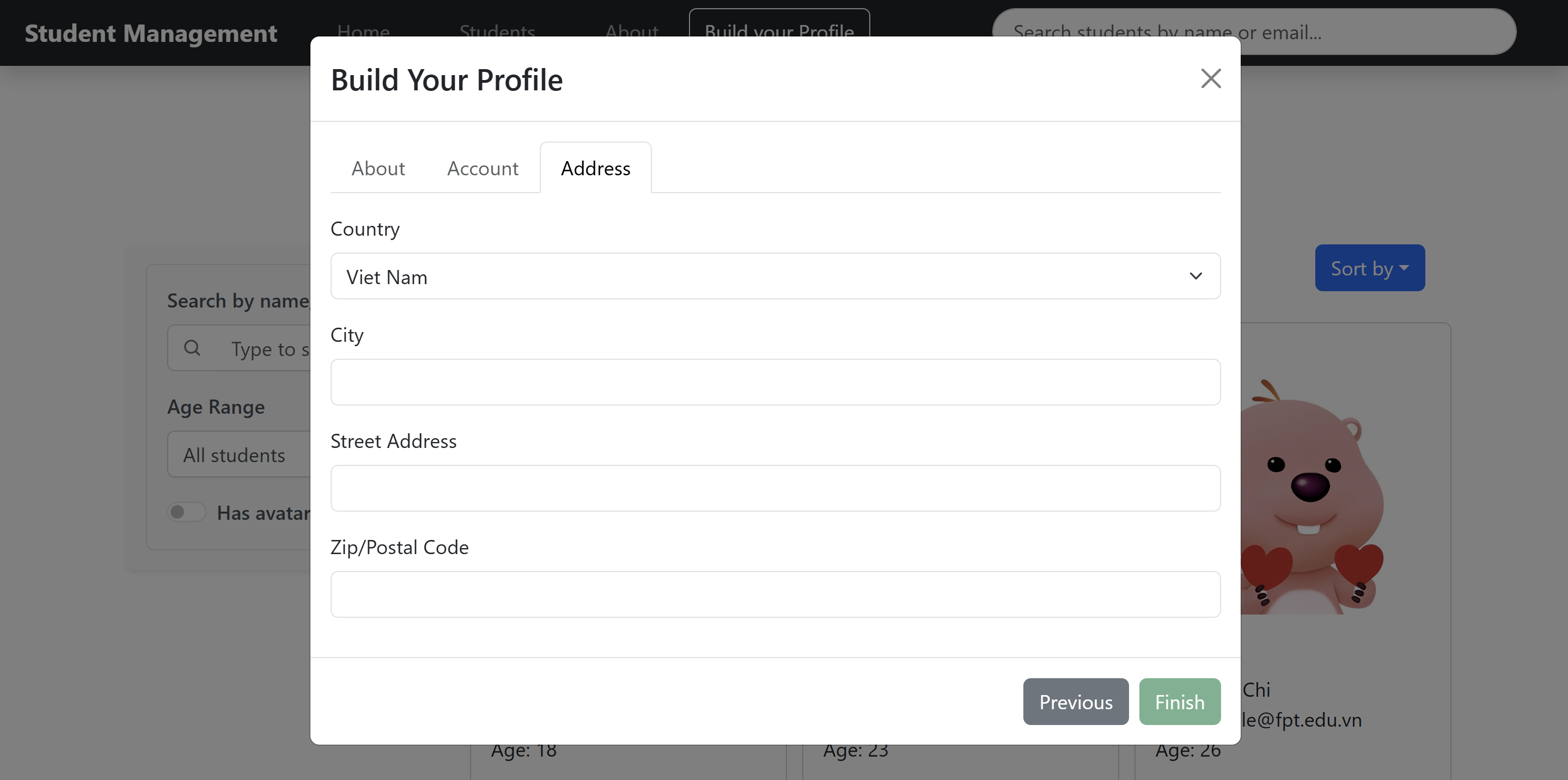
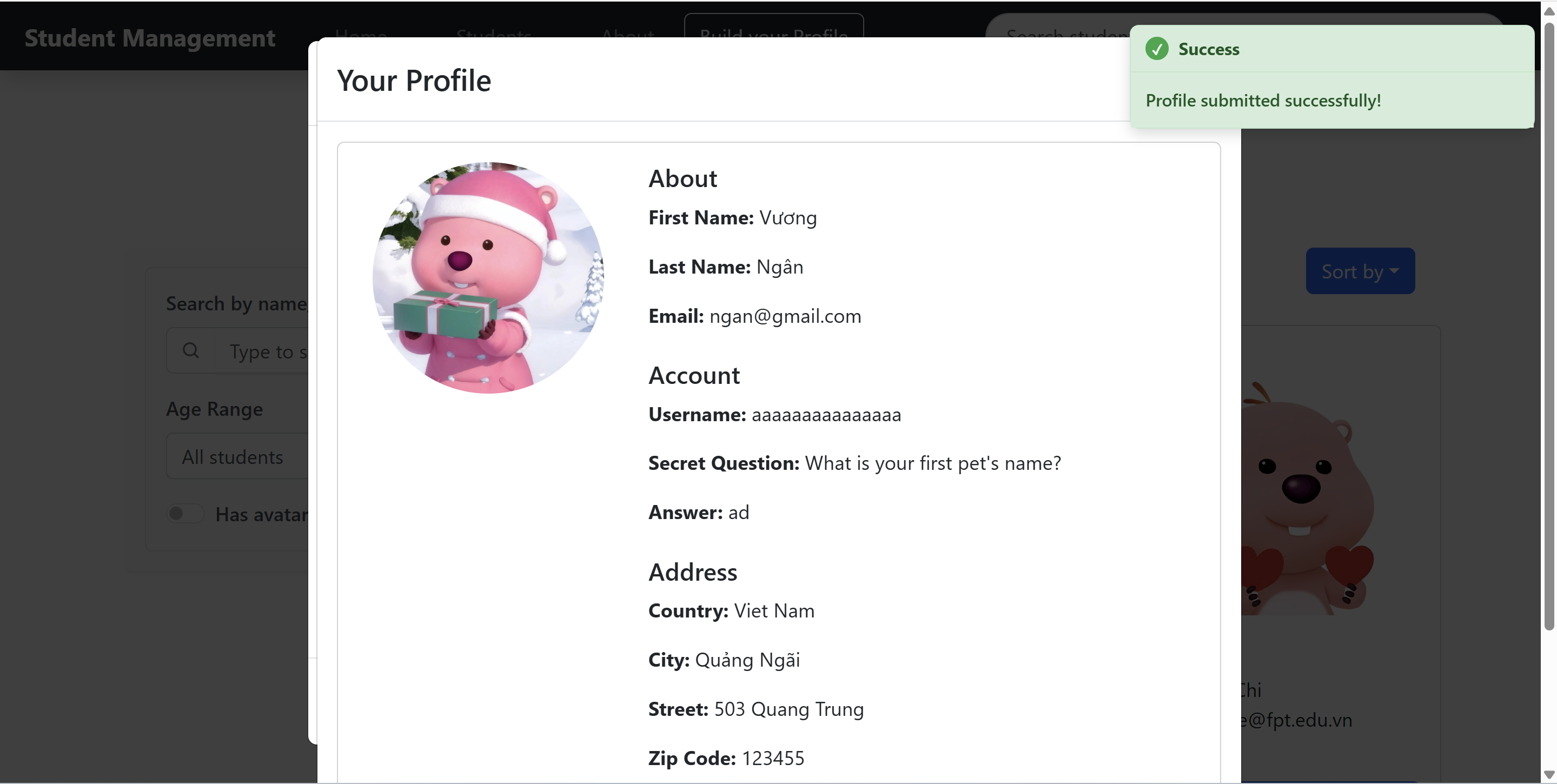
# Lab 3

## ProfileWizard.js









import React, { useState, useReducer, useMemo, useCallback } from 'react';

import { Modal, Form, Button, Tab, Tabs, Card, Toast, InputGroup, Image, Col, Row } from 'react-bootstrap';

import { FaEye, FaEyeSlash } from 'react-icons/fa';

import PropTypes from 'prop-types';

const formReducer = (state, action) => {

    switch (action.type) {

        case 'UPDATE\_FIELD':

            return {

                ...state,

                [action.section]: {

                    ...state[action.section],

                    [action.field]: action.value

                }

            };

        case 'UPDATE\_AVATAR':

            return {

                ...state,

                about: {

                    ...state.about,

                    avatar: action.value

                }

            };

        case 'RESET\_FORM':

            return {

                about: {

                    firstName: '',

                    lastName: '',

                    email: '',

                    avatar: null

                },

                account: {

                    username: '',

                    password: '',

                    confirmPassword: '',

                    secretQuestion: secretQuestions[0],

                    secretAnswer: '',

                    showPassword: false,

                    showConfirmPassword: false

                },

                address: {

                    country: countries[0],

                    city: '',

                    street: '',

                    zipCode: ''

                }

            };

        default:

            return state;

    }

};

const secretQuestions = [

    "What is your first pet's name?",

    "What is your mother's maiden name?",

    "In which city were you born?",

    "Who was your favorite teacher?"

];

const countries = ["Viet Nam", "Korea", "Italy", "Japan", "USA"];

const ProfileWizard = ({ show, onHide }) => {

    const [state, dispatch] = useReducer(formReducer, {

        about: {

            firstName: '',

            lastName: '',

            email: '',

            avatar: null

        },

        account: {

            username: '',

            password: '',

            confirmPassword: '',

            secretQuestion: secretQuestions[0],

            secretAnswer: '',

            showPassword: false,

            showConfirmPassword: false

        },

        address: {

            country: countries[0],

            city: '',

            street: '',

            zipCode: ''

        }

    });

    const [touched, setTouched] = useState({

        about: {

            firstName: false,

            lastName: false,

            email: false

        },

        account: {

            username: false,

            password: false,

            confirmPassword: false,

            secretAnswer: false

        },

        address: {

            city: false,

            street: false,

            zipCode: false

        }

    });

    const [activeTab, setActiveTab] = useState('about');

    const [showSuccess, setShowSuccess] = useState(false);

    const [showProfile, setShowProfile] = useState(false);

    const [attemptedNext, setAttemptedNext] = useState(false);

    // Validation functions

    const isAboutValid = useMemo(() => (

        state.about.firstName.trim() &&

        state.about.lastName.trim() &&

        /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(state.about.email)

    ), [state.about]);

    const isAccountValid = useMemo(() => {

        const hasUpperCase = /[A-Z]/.test(state.account.password);

        const hasNumber = /\d/.test(state.account.password);

        const hasSpecial = /[!@#$%^&\*]/.test(state.account.password);

        return (

            state.account.username.length >= 6 &&

            state.account.password.length >= 8 &&

            hasUpperCase &&

            hasNumber &&

            hasSpecial &&

            state.account.password === state.account.confirmPassword &&

            state.account.secretAnswer.trim()

        );

    }, [state.account]);

    const isAddressValid = useMemo(() => (

        state.address.street.trim() &&

        state.address.city.trim() &&

        state.address.zipCode.trim()

    ), [state.address]);

    const updateField = useCallback((section, field, value) => {

        dispatch({ type: 'UPDATE\_FIELD', section, field, value });

    }, []);

    const handleFileChange = useCallback((e) => {

        const file = e.target.files[0];

        if (file) {

            const reader = new FileReader();

            reader.onloadend = () => {

                dispatch({ type: 'UPDATE\_AVATAR', value: reader.result });

            };

            reader.readAsDataURL(file);

        }

    }, []);

    const togglePasswordVisibility = useCallback((field) => {

        updateField('account', `show${field}`, !state.account[`show${field}`]);

    }, [state.account, updateField]);

    const handleBlur = useCallback((section, field) => {

        setTouched(prev => ({

            ...prev,

            [section]: {

                ...prev[section],

                [field]: true

            }

        }));

    }, []);

    const nextStep = useCallback(() => {

        setAttemptedNext(true);

        if (activeTab === 'about' && !isAboutValid) return;

        if (activeTab === 'account' && !isAccountValid) return;

        if (activeTab === 'about') setActiveTab('account');

        else if (activeTab === 'account') setActiveTab('address');

    }, [activeTab, isAboutValid, isAccountValid]);

    const prevStep = useCallback(() => {

        if (activeTab === 'account') setActiveTab('about');

        else if (activeTab === 'address') setActiveTab('account');

    }, [activeTab]);

    const handleSubmit = useCallback(() => {

        if (isAddressValid) {

            setShowProfile(true);

            setShowSuccess(true);

            setTimeout(() => {

                setShowSuccess(false);

            }, 3000);

        }

    }, [isAddressValid]);

    const handleCloseProfile = useCallback(() => {

        setShowProfile(false);

        onHide();

        dispatch({ type: 'RESET\_FORM' });

        setActiveTab('about');

        setAttemptedNext(false);

        setTouched({

            about: {

                firstName: false,

                lastName: false,

                email: false

            },

            account: {

                username: false,

                password: false,

                confirmPassword: false,

                secretAnswer: false

            },

            address: {

                city: false,

                street: false,

                zipCode: false

            }

        });

    }, [onHide]);

    return (

        <>

            <Modal show={show} onHide={onHide} size="lg" centered>

                <Modal.Header closeButton>

                    <Modal.Title>Build Your Profile</Modal.Title>

                </Modal.Header>

                <Modal.Body>

                    <Tabs activeKey={activeTab} onSelect={setActiveTab} className="mb-3">

                        <Tab eventKey="about" title="About" disabled={activeTab !== 'about'}>

                            <Form>

                                <Row>

                                    <Col md={6}>

                                        <Form.Group className="mb-3">

                                            <Form.Label>First Name \*</Form.Label>

                                            <Form.Control

                                                type="text"

                                                value={state.about.firstName}

                                                onChange={(e) => updateField('about', 'firstName', e.target.value)}

                                                onBlur={() => handleBlur('about', 'firstName')}

                                                required

                                                isInvalid={(touched.about.firstName || attemptedNext) && !state.about.firstName.trim()}

                                            />

                                            <Form.Control.Feedback type="invalid">

                                                Please enter your first name

                                            </Form.Control.Feedback>

                                        </Form.Group>

                                    </Col>

                                    <Col md={6}>

                                        <Form.Group className="mb-3">

                                            <Form.Label>Last Name \*</Form.Label>

                                            <Form.Control

                                                type="text"

                                                value={state.about.lastName}

                                                onChange={(e) => updateField('about', 'lastName', e.target.value)}

                                                onBlur={() => handleBlur('about', 'lastName')}

                                                required

                                                isInvalid={(touched.about.lastName || attemptedNext) && !state.about.lastName.trim()}

                                            />

                                            <Form.Control.Feedback type="invalid">

                                                Please enter your last name

                                            </Form.Control.Feedback>

                                        </Form.Group>

                                    </Col>

                                </Row>

                                <Form.Group className="mb-3">

                                    <Form.Label>Email \*</Form.Label>

                                    <Form.Control

                                        type="email"

                                        value={state.about.email}

                                        onChange={(e) => updateField('about', 'email', e.target.value)}

                                        onBlur={() => handleBlur('about', 'email')}

                                        required

                                        isInvalid={(touched.about.email || attemptedNext) &&

                                            (!state.about.email || !/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(state.about.email))}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        {!state.about.email

                                            ? 'Please enter your email'

                                            : 'Please enter a valid email address'}

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Avatar</Form.Label>

                                    <Form.Control

                                        type="file"

                                        accept="image/\*"

                                        onChange={handleFileChange}

                                    />

                                    {state.about.avatar && (

                                        <Image

                                            src={state.about.avatar}

                                            alt="Avatar Preview"

                                            thumbnail

                                            className="mt-2"

                                            style={{ maxWidth: '150px' }}

                                        />

                                    )}

                                </Form.Group>

                            </Form>

                        </Tab>

                        <Tab eventKey="account" title="Account" disabled={activeTab !== 'account'}>

                            <Form>

                                <Form.Group className="mb-3">

                                    <Form.Label>Username (min 6 characters) \*</Form.Label>

                                    <Form.Control

                                        type="text"

                                        value={state.account.username}

                                        onChange={(e) => updateField('account', 'username', e.target.value)}

                                        onBlur={() => handleBlur('account', 'username')}

                                        required

                                        minLength={6}

                                        isInvalid={(touched.account.username || attemptedNext) &&

                                            (!state.account.username || state.account.username.length < 6)}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        {!state.account.username

                                            ? 'Please enter your username'

                                            : 'Username must be at least 6 characters'}

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Password \*</Form.Label>

                                    <InputGroup>

                                        <Form.Control

                                            type={state.account.showPassword ? "text" : "password"}

                                            value={state.account.password}

                                            onChange={(e) => updateField('account', 'password', e.target.value)}

                                            onBlur={() => handleBlur('account', 'password')}

                                            required

                                            minLength={8}

                                            isInvalid={(touched.account.password || attemptedNext) &&

                                                (!state.account.password ||

                                                !/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/.test(state.account.password))}

                                        />

                                        <Button

                                            variant="outline-secondary"

                                            onClick={() => togglePasswordVisibility('Password')}

                                        >

                                            {state.account.showPassword ? <FaEyeSlash /> : <FaEye />}

                                        </Button>

                                    </InputGroup>

                                    <Form.Text>

                                        Must contain: uppercase, lowercase, number, and special character

                                    </Form.Text>

                                    <Form.Control.Feedback type="invalid">

                                        {!state.account.password

                                            ? 'Please enter your password'

                                            : 'Password must contain at least one uppercase letter, one lowercase letter, one number, and one special character'}

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Confirm Password \*</Form.Label>

                                    <InputGroup>

                                        <Form.Control

                                            type={state.account.showConfirmPassword ? "text" : "password"}

                                            value={state.account.confirmPassword}

                                            onChange={(e) => updateField('account', 'confirmPassword', e.target.value)}

                                            onBlur={() => handleBlur('account', 'confirmPassword')}

                                            required

                                            isInvalid={(touched.account.confirmPassword || attemptedNext) &&

                                                (!state.account.confirmPassword ||

                                                state.account.password !== state.account.confirmPassword)}

                                        />

                                        <Button

                                            variant="outline-secondary"

                                            onClick={() => togglePasswordVisibility('ConfirmPassword')}

                                        >

                                            {state.account.showConfirmPassword ? <FaEyeSlash /> : <FaEye />}

                                        </Button>

                                    </InputGroup>

                                    <Form.Control.Feedback type="invalid">

                                        {!state.account.confirmPassword

                                            ? 'Please confirm your password'

                                            : 'Passwords do not match'}

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Secret Question \*</Form.Label>

                                    <Form.Select

                                        value={state.account.secretQuestion}

                                        onChange={(e) => updateField('account', 'secretQuestion', e.target.value)}

                                        required

                                    >

                                        {secretQuestions.map((question, index) => (

                                            <option key={index} value={question}>{question}</option>

                                        ))}

                                    </Form.Select>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Answer \*</Form.Label>

                                    <Form.Control

                                        type="text"

                                        value={state.account.secretAnswer}

                                        onChange={(e) => updateField('account', 'secretAnswer', e.target.value)}

                                        onBlur={() => handleBlur('account', 'secretAnswer')}

                                        required

                                        isInvalid={(touched.account.secretAnswer || attemptedNext) && !state.account.secretAnswer.trim()}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        Please enter an answer for your secret question

                                    </Form.Control.Feedback>

                                </Form.Group>

                            </Form>

                        </Tab>

                        <Tab eventKey="address" title="Address" disabled={activeTab !== 'address'}>

                            <Form>

                                <Form.Group className="mb-3">

                                    <Form.Label>Country</Form.Label>

                                    <Form.Select

                                        value={state.address.country}

                                        onChange={(e) => updateField('address', 'country', e.target.value)}

                                        required

                                    >

                                        {countries.map((country, index) => (

                                            <option key={index} value={country}>{country}</option>

                                        ))}

                                    </Form.Select>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>City \*</Form.Label>

                                    <Form.Control

                                        type="text"

                                        value={state.address.city}

                                        onChange={(e) => updateField('address', 'city', e.target.value)}

                                        onBlur={() => handleBlur('address', 'city')}

                                        required

                                        isInvalid={(touched.address.city || attemptedNext) && !state.address.city.trim()}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        Please enter your city

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Street Address \*</Form.Label>

                                    <Form.Control

                                        type="text"

                                        value={state.address.street}

                                        onChange={(e) => updateField('address', 'street', e.target.value)}

                                        onBlur={() => handleBlur('address', 'street')}

                                        required

                                        isInvalid={(touched.address.street || attemptedNext) && !state.address.street.trim()}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        Please enter your street address

                                    </Form.Control.Feedback>

                                </Form.Group>

                                <Form.Group className="mb-3">

                                    <Form.Label>Zip/Postal Code \*</Form.Label>

                                    <Form.Control

                                        type="text"

                                        value={state.address.zipCode}

                                        onChange={(e) => updateField('address', 'zipCode', e.target.value)}

                                        onBlur={() => handleBlur('address', 'zipCode')}

                                        required

                                        isInvalid={(touched.address.zipCode || attemptedNext) && !state.address.zipCode.trim()}

                                    />

                                    <Form.Control.Feedback type="invalid">

                                        Please enter your zip/postal code

                                    </Form.Control.Feedback>

                                </Form.Group>

                            </Form>

                        </Tab>

                    </Tabs>

                </Modal.Body>

                <Modal.Footer>

                    {activeTab !== 'about' && (

                        <Button variant="secondary" onClick={prevStep}>

                            Previous

                        </Button>

                    )}

                    {activeTab !== 'address' ? (

                        <Button

                            variant="primary"

                            onClick={nextStep}

                            disabled={

                                (activeTab === 'about' && !isAboutValid && !attemptedNext) ||

                                (activeTab === 'account' && !isAccountValid && !attemptedNext)

                            }

                        >

                            Next

                        </Button>

                    ) : (

                        <Button

                            variant="success"

                            onClick={handleSubmit}

                            disabled={!isAddressValid}

                        >

                            Finish

                        </Button>

                    )}

                </Modal.Footer>

            </Modal>

            <Modal show={showProfile} onHide={handleCloseProfile} size="lg" centered>

                <Modal.Header closeButton>

                    <Modal.Title>Your Profile</Modal.Title>

                </Modal.Header>

                <Modal.Body>

                    <Card>

                        <Card.Body>

                            <div className="row">

                                <div className="col-md-4 text-center">

                                    {state.about.avatar && (

                                        <Image

                                            src={state.about.avatar}

                                            roundedCircle

                                            fluid

                                            style={{ width: '200px', height: '200px', objectFit: 'cover' }}

                                        />

                                    )}

                                </div>

                                <div className="col-md-8">

                                    <h5>About</h5>

                                    <p><strong>First Name:</strong> {state.about.firstName}</p>

                                    <p><strong>Last Name:</strong> {state.about.lastName}</p>

                                    <p><strong>Email:</strong> {state.about.email}</p>

                                    <h5 className="mt-4">Account</h5>

                                    <p><strong>Username:</strong> {state.account.username}</p>

                                    <p><strong>Secret Question:</strong> {state.account.secretQuestion}</p>

                                    <p><strong>Answer:</strong> {state.account.secretAnswer}</p>

                                    <h5 className="mt-4">Address</h5>

                                    <p><strong>Country:</strong> {state.address.country}</p>

                                    <p><strong>City:</strong> {state.address.city}</p>

                                    <p><strong>Street:</strong> {state.address.street}</p>

                                    <p><strong>Zip Code:</strong> {state.address.zipCode}</p>

                                </div>

                            </div>

                        </Card.Body>

                    </Card>

                </Modal.Body>

                <Modal.Footer>

                    <Button variant="primary" onClick={handleCloseProfile}>

                        Close

                    </Button>

                </Modal.Footer>

            </Modal>

            <Toast

                show={showSuccess}

                onClose={() => setShowSuccess(false)}

                delay={5000}

                autohide

                style={{

                    position: 'fixed',

                    top: '20px',

                    right: '20px',

                    zIndex: 9999,

                    backgroundColor: '#d4edda',

                    border: '1px solid #c3e6cb',

                    borderRadius: '8px',

                    boxShadow: '0 4px 12px rgba(0,0,0,0.15)',

                    minWidth: '300px'

                }}

            >

                <Toast.Header

                    closeButton={false}

                    style={{

                        backgroundColor: '#d4edda',

                        borderBottom: '1px solid #c3e6cb',

                        color: '#155724'

                    }}

                >

                    <div className="d-flex align-items-center">

                        <div

                            className="me-2"

                            style={{

                                width: '20px',

                                height: '20px',

                                backgroundColor: '#28a745',

                                borderRadius: '50%',

                                display: 'flex',

                                alignItems: 'center',

                                justifyContent: 'center',

                                color: 'white',

                                fontSize: '12px',

                                fontWeight: 'bold'

                            }}

                        >

                            ✓

                        </div>

                        <strong className="me-auto" style={{ color: '#155724' }}>Success</strong>

                    </div>

                </Toast.Header>

                <Toast.Body style={{

                    backgroundColor: '#d4edda',

                    color: '#155724',

                    fontWeight: '500'

                }}>

                    Profile submitted successfully!

                </Toast.Body>

            </Toast>

        </>

    );

};

ProfileWizard.propTypes = {

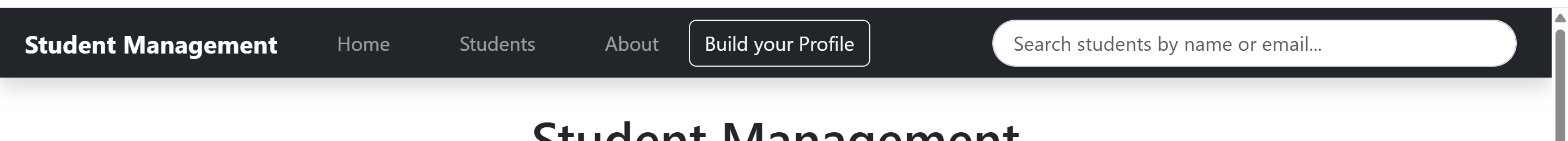
    show: PropTypes.bool.isRequired,

    onHide: PropTypes.func.isRequired

};

export default ProfileWizard;

## Nvbar.js



import React, { useState } from 'react'; // Thêm useState

import { Navbar as BootstrapNavbar, Nav, Form, FormControl, Container, Button } from 'react-bootstrap';

function Navbar({ onQuickSearch, onShowProfile }) {

  const [searchTerm, setSearchTerm] = useState('');

  const handleSearch = (e) => {

    setSearchTerm(e.target.value);

    onQuickSearch(e.target.value);

  };

  return (

    <BootstrapNavbar bg="dark" variant="dark" expand="lg" sticky="top" className="shadow">

      <Container fluid>

        <BootstrapNavbar.Brand href="#home" className="d-flex align-items-center">

          <i className="bi bi-people-fill me-2" style={{ fontSize: '1.5rem' }}></i>

          <span className="fw-bold">Student Management</span>

        </BootstrapNavbar.Brand>

        <BootstrapNavbar.Toggle aria-controls="navbarScroll" className="border-0" />

        <BootstrapNavbar.Collapse id="navbarScroll">

          <Nav className="me-auto my-2 my-lg-0" style={{ maxHeight: '100px' }} navbarScroll>

            <Nav.Link href="#home" className="mx-2 px-3 py-2 rounded-3">

              <i className="bi bi-house-door me-2"></i>Home

            </Nav.Link>

            <Nav.Link href="#students" className="mx-2 px-3 py-2 rounded-3">

              <i className="bi bi-people me-2"></i>Students

            </Nav.Link>

            <Nav.Link href="#about" className="mx-2 px-3 py-2 rounded-3">

              <i className="bi bi-info-circle me-2"></i>About

            </Nav.Link>

          </Nav>

          <Button

            variant="outline-light"

            className="me-2"

            onClick={onShowProfile}

          >

            Build your Profile

          </Button>

          <Form className="d-flex flex-grow-1 mx-3">

            <FormControl

              type="search"

              placeholder="Search students by name or email..."

              aria-label="Search"

              value={searchTerm}

              onChange={handleSearch}

              className="rounded-pill px-3"

              style={{

                marginLeft: 'auto',

                maxWidth: '450px',

                width: '100%'

              }}

            />

          </Form>

        </BootstrapNavbar.Collapse>

      </Container>

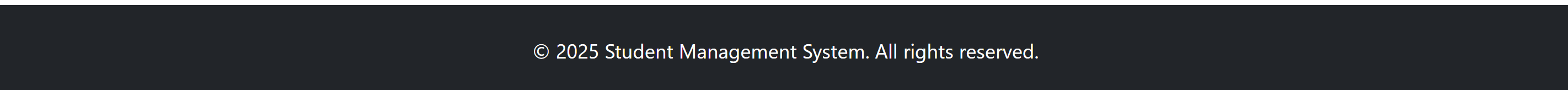
    </BootstrapNavbar>

  );

}

export default Navbar;

## Footer.js



import React from 'react';

import { Container, Navbar } from 'react-bootstrap';

function Footer() {

  return (

    <Navbar bg="dark" variant="dark" className="mt-auto py-3">

      <Container>

        <Navbar.Text className="text-center w-100 text-white">

          © {new Date().getFullYear()} Student Management System. All rights reserved.

        </Navbar.Text>

      </Container>

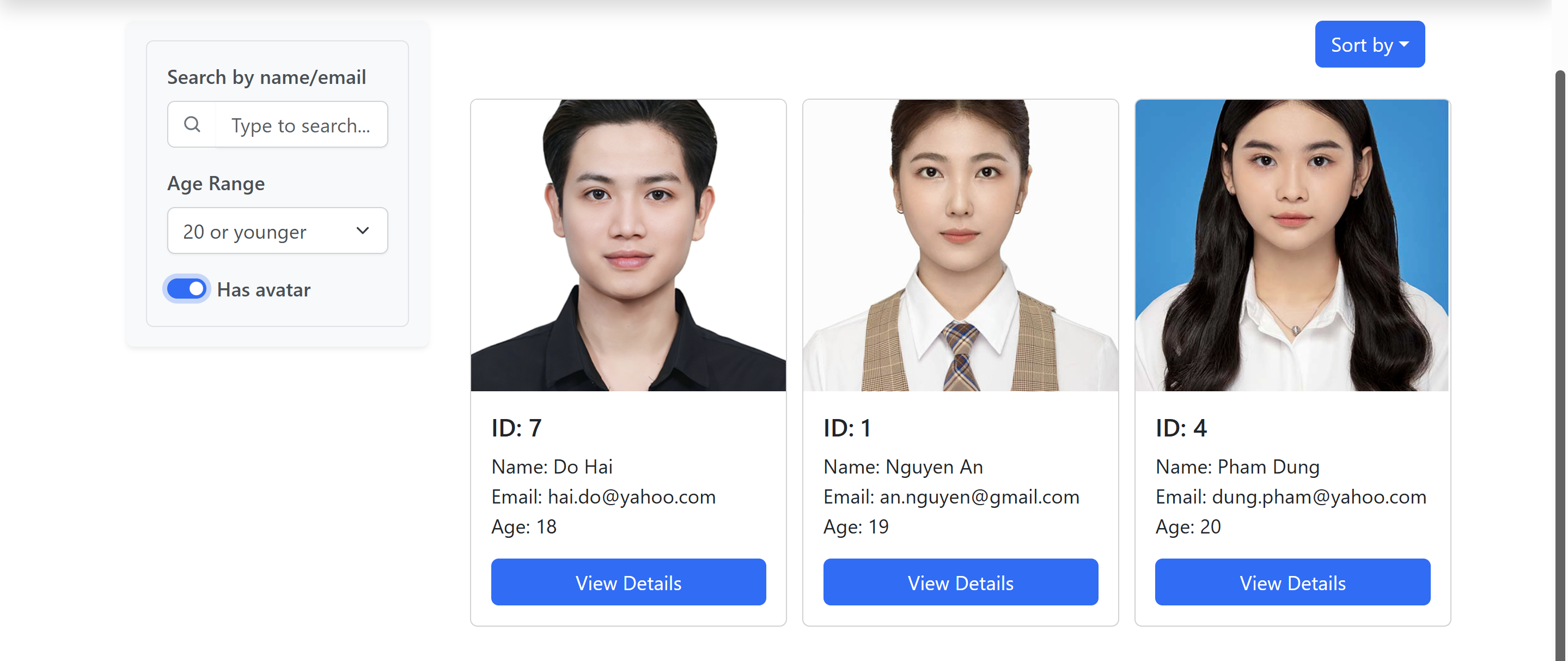
    </Navbar>

  );

}

export default Footer;

## Filters.js



import React, { useState } from 'react';

import { Form, InputGroup } from 'react-bootstrap';

function Filters({ onFilter }) {

  const [searchTerm, setSearchTerm] = useState('');

  const [ageRange, setAgeRange] = useState('all');

  const [hasAvatar, setHasAvatar] = useState(false);

  const handleFilter = () => {

    onFilter({ searchTerm, ageRange, hasAvatar });

  };

  return (

    <Form className="p-3 border rounded" style={{

      maxWidth: '400px',

      backgroundColor: '#f8f9fa',

      borderColor: '#dee2e6'

    }}>

      {/\* Search Field \*/}

      <Form.Group controlId="formSearch" className="mb-3">

        <Form.Label className="fw-semibold" style={{ color: '#495057' }}>

          Search by name/email

        </Form.Label>

        <InputGroup>

          <InputGroup.Text style={{

            backgroundColor: '#fff',

            borderColor: '#ced4da'

          }}>

            <svg width="16" height="16" viewBox="0 0 16 16" fill="#6c757d">

              <path fillRule="evenodd" d="M11.5 7a4.5 4.5 0 11-9 0 4.5 4.5 0 019 0zm-.82 4.74a6 6 0 111.06-1.06l2.79 2.79a.75.75 0 11-1.06 1.06l-2.79-2.79z" clipRule="evenodd"/>

            </svg>

          </InputGroup.Text>

          <Form.Control

            type="text"

            placeholder="Type to search..."

            value={searchTerm}

            onChange={(e) => {

              setSearchTerm(e.target.value);

              handleFilter();

            }}

            style={{

              borderLeft: 'none',

              boxShadow: '0 1px 2px rgba(0,0,0,0.05)',

              borderColor: '#ced4da'

            }}

          />

        </InputGroup>

      </Form.Group>

      {/\* Age Range Select \*/}

      <Form.Group controlId="formAgeRange" className="mb-3">

        <Form.Label className="fw-semibold" style={{ color: '#495057' }}>

          Age Range

        </Form.Label>

        <Form.Select

          value={ageRange}

          onChange={(e) => {

            setAgeRange(e.target.value);

            handleFilter();

          }}

          style={{

            boxShadow: '0 1px 2px rgba(0,0,0,0.05)',

            borderColor: '#ced4da',

            color: '#495057'

          }}

        >

          <option value="all">All students</option>

          <option value="<=20">20 or younger</option>

          <option value="21-25">21 to 25 years</option>

          <option value=">25">Over 25 years</option>

        </Form.Select>

      </Form.Group>

      {/\* Avatar Toggle \*/}

      <Form.Group controlId="formHasAvatar">

        <Form.Check

          type="switch"

          id="avatar-switch"

          label={

            <span className="fw-semibold" style={{ color: '#495057' }}>

              Has avatar

            </span>

          }

          checked={hasAvatar}

          onChange={(e) => {

            setHasAvatar(e.target.checked);

            handleFilter();

          }}

          style={{

            '--bs-form-switch-bg': 'url("data:image/svg+xml,%3csvg xmlns=%27http://www.w3.org/2000/svg%27 viewBox=%27-4 -4 8 8%27%3e%3ccircle r=%273%27 fill=%27%23fff%27/%3e%3c/svg%3e")',

            '--bs-form-switch-border-color': '#adb5bd',

            '--bs-form-switch-bg-color': '#6c757d'

          }}

        />

      </Form.Group>

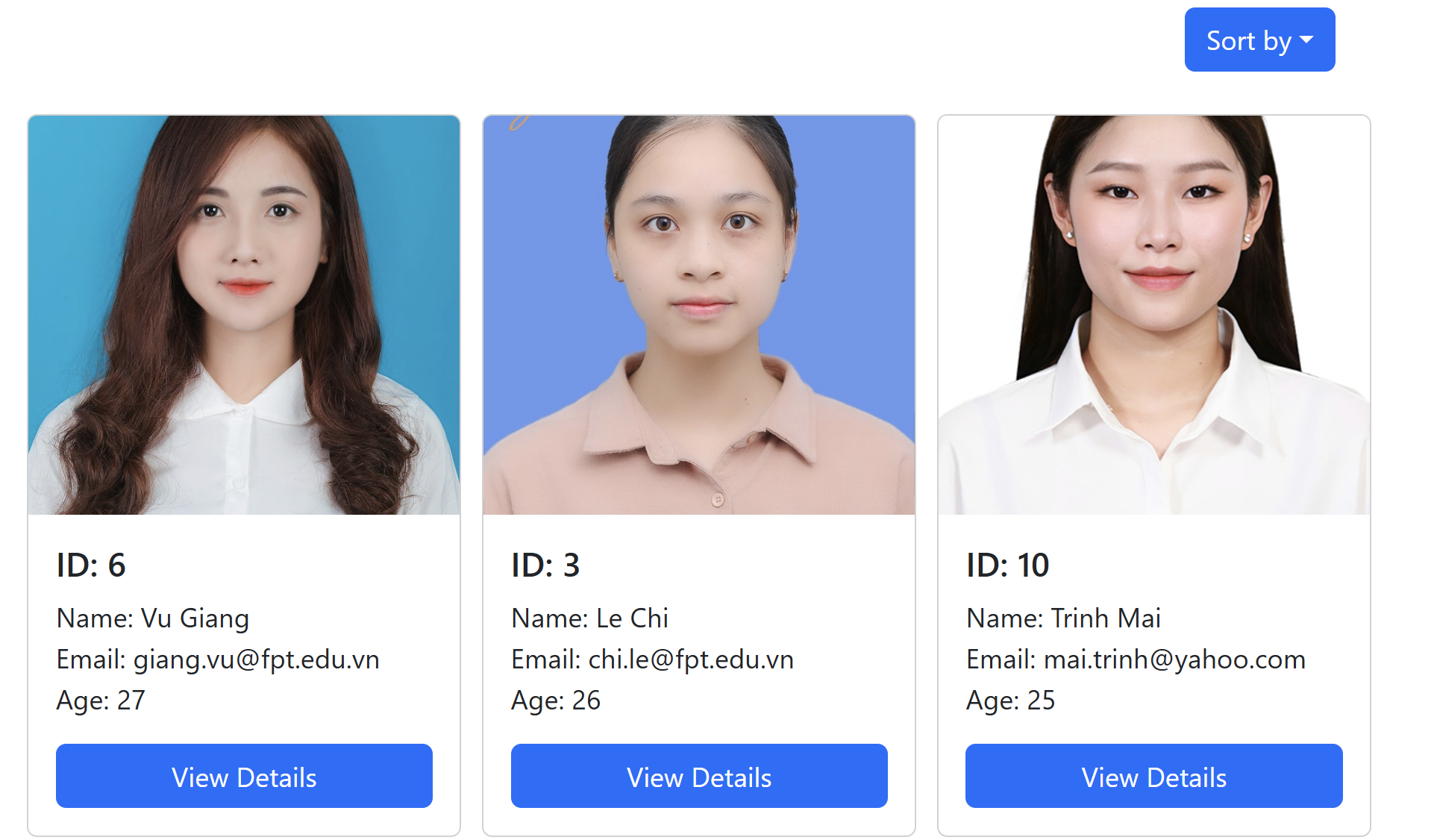
    </Form>

  );

}

export default Filters;

## SortDropdown.js



import React, { useState } from 'react';

import { Dropdown } from 'react-bootstrap';

function SortDropdown({ onSort }) {

  const [sortOption, setSortOption] = useState('nameAsc');

  const handleSort = (option) => {

    setSortOption(option);

    onSort(option);

  };

  return (

    <Dropdown className="mb-3 text-end">

      <Dropdown.Toggle variant="primary" id="dropdown-basic">

        Sort by

      </Dropdown.Toggle>

      <Dropdown.Menu>

        <Dropdown.Item onClick={() => handleSort('ageAsc')}>Age ↑</Dropdown.Item>

        <Dropdown.Item onClick={() => handleSort('ageDesc')}>Age ↓</Dropdown.Item>

        <Dropdown.Item onClick={() => handleSort('nameAsc')}>Name A-Z</Dropdown.Item>

        <Dropdown.Item onClick={() => handleSort('nameDesc')}>Name Z-A</Dropdown.Item>

      </Dropdown.Menu>

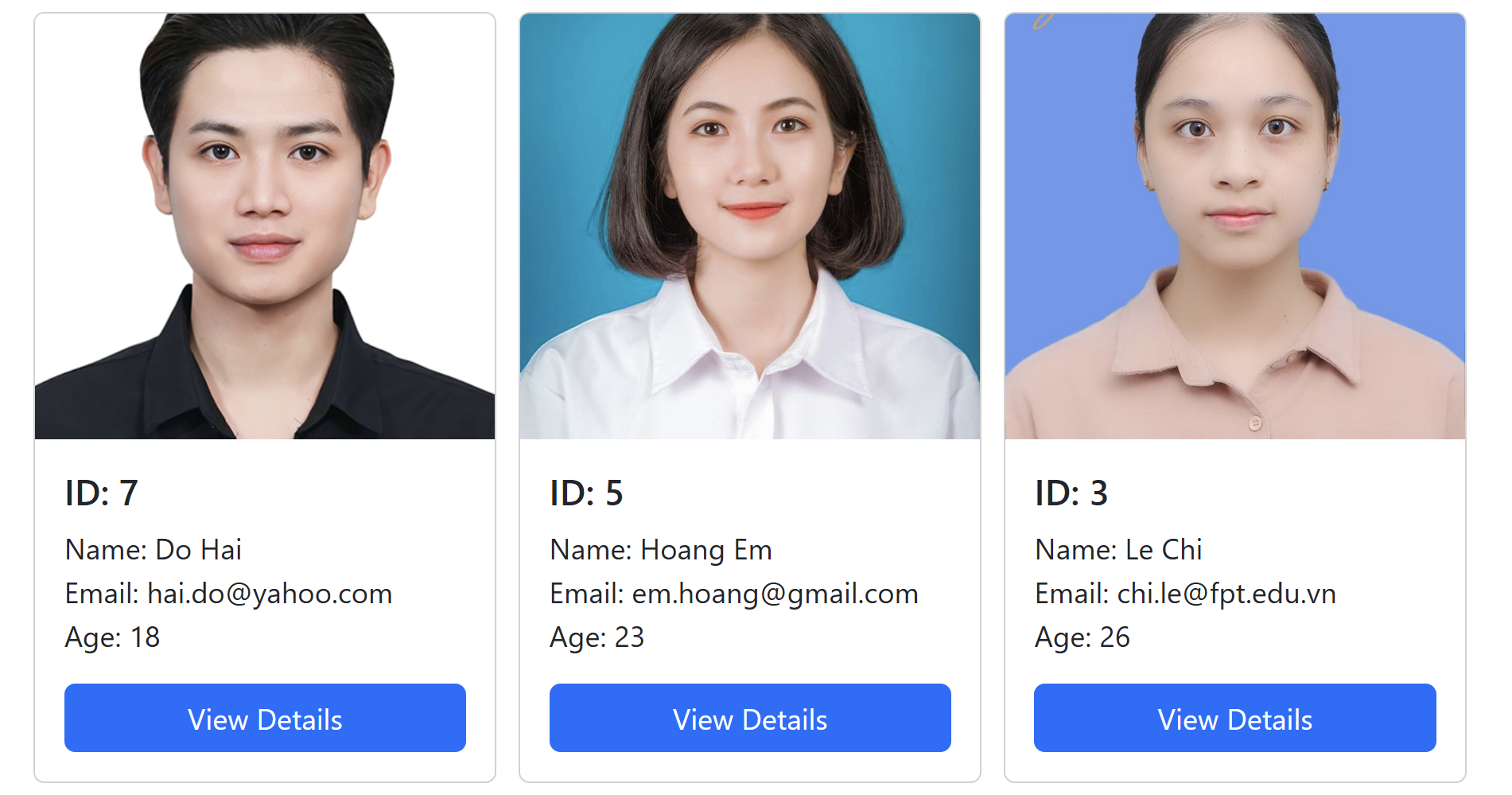
    </Dropdown>

  );

}

export default SortDropdown;

## StudentCard.js



import React from 'react';

import { Card } from 'react-bootstrap';

import PropTypes from 'prop-types';

function StudentCard({ student, onViewDetails }) {

  return (

    <Card style={{ width: '16rem', height: '97%', margin: '10px', display: 'flex', flexDirection: 'column', }}>

      <Card.Img variant="top" src={student.avatar || 'https://via.placeholder.com/150'} style={{ width: '100%', height: '250px', objectFit: 'cover' }} />

      <Card.Body style={{ flex: '1', display: 'flex', flexDirection: 'column', justifyContent: 'space-between' }}>

        <div>

          <Card.Title>ID: {student.id}</Card.Title>

          <Card.Text>

            Name: {student.name}<br />

            Email: {student.email}<br />

            Age: {student.age}

          </Card.Text>

        </div>

        <button onClick={() => onViewDetails(student)} style={{ width: '100%', marginTop: '10px' }} className="btn btn-primary">

          View Details

        </button>

      </Card.Body>

    </Card>

  );

}

StudentCard.propTypes = {

  student: PropTypes.shape({

    id: PropTypes.number.isRequired,

    name: PropTypes.string.isRequired,

    email: PropTypes.string.isRequired,

    age: PropTypes.number.isRequired,

    avatar: PropTypes.string,

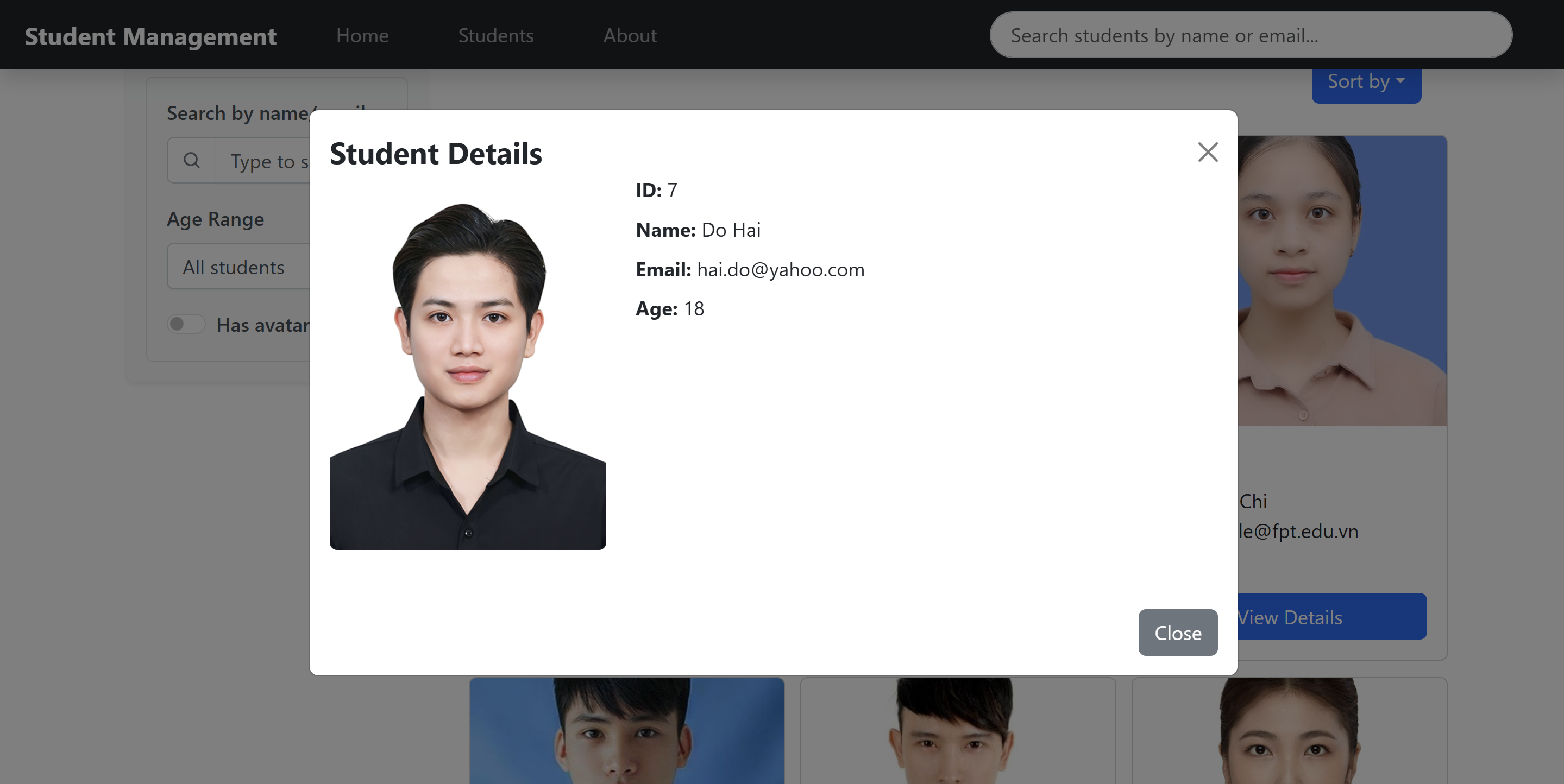
  }).isRequired,

  onViewDetails: PropTypes.func.isRequired,

};

export default StudentCard;

## StudentDetailModal.js



import React from 'react';

import { Modal, Button, Image } from 'react-bootstrap';

function StudentDetailModal({ show, onHide, student }) {

  return (

    <Modal show={show} onHide={onHide} centered size="lg">

      <Modal.Header closeButton className="border-0 pb-0">

        <Modal.Title className="fw-bold">Student Details</Modal.Title>

      </Modal.Header>

      <Modal.Body className="pt-0">

        <div className="row">

          <div className="col-md-4 text-center">

            <Image

              src={student.avatar || 'https://via.placeholder.com/300'}

              alt={student.name}

              fluid

              rounded

              className="mb-3"

            />

          </div>

          <div className="col-md-8">

            <ul className="list-unstyled">

              <li className="mb-2"><strong>ID:</strong> {student.id}</li>

              <li className="mb-2"><strong>Name:</strong> {student.name}</li>

              <li className="mb-2"><strong>Email:</strong> {student.email}</li>

              <li className="mb-2"><strong>Age:</strong> {student.age}</li>

            </ul>

          </div>

        </div>

      </Modal.Body>

      <Modal.Footer className="border-0">

        <Button variant="secondary" onClick={onHide}>

          Close

        </Button>

      </Modal.Footer>

    </Modal>

  );

}

export default StudentDetailModal;

## StudentGrid.js

import React from 'react';

import StudentCard from './StudentCard';

function StudentGrid({ students, onViewDetails }) {

  return (

    <div className="row">

      {students.map((student) => (

        <div key={student.id} className="col-12 col-md-6 col-lg-4">

          <StudentCard student={student} onViewDetails={onViewDetails} />

        </div>

      ))}

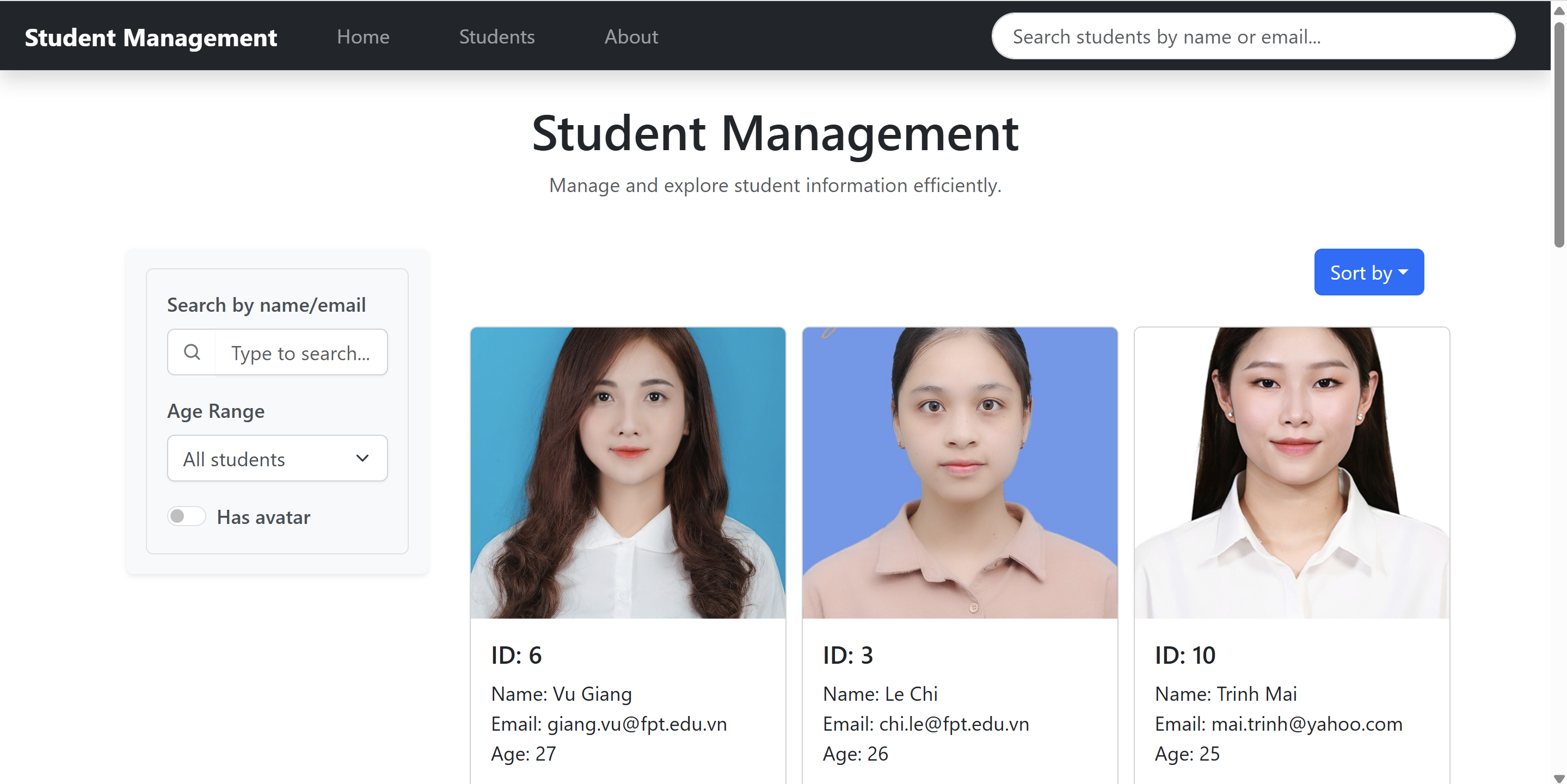
    </div>

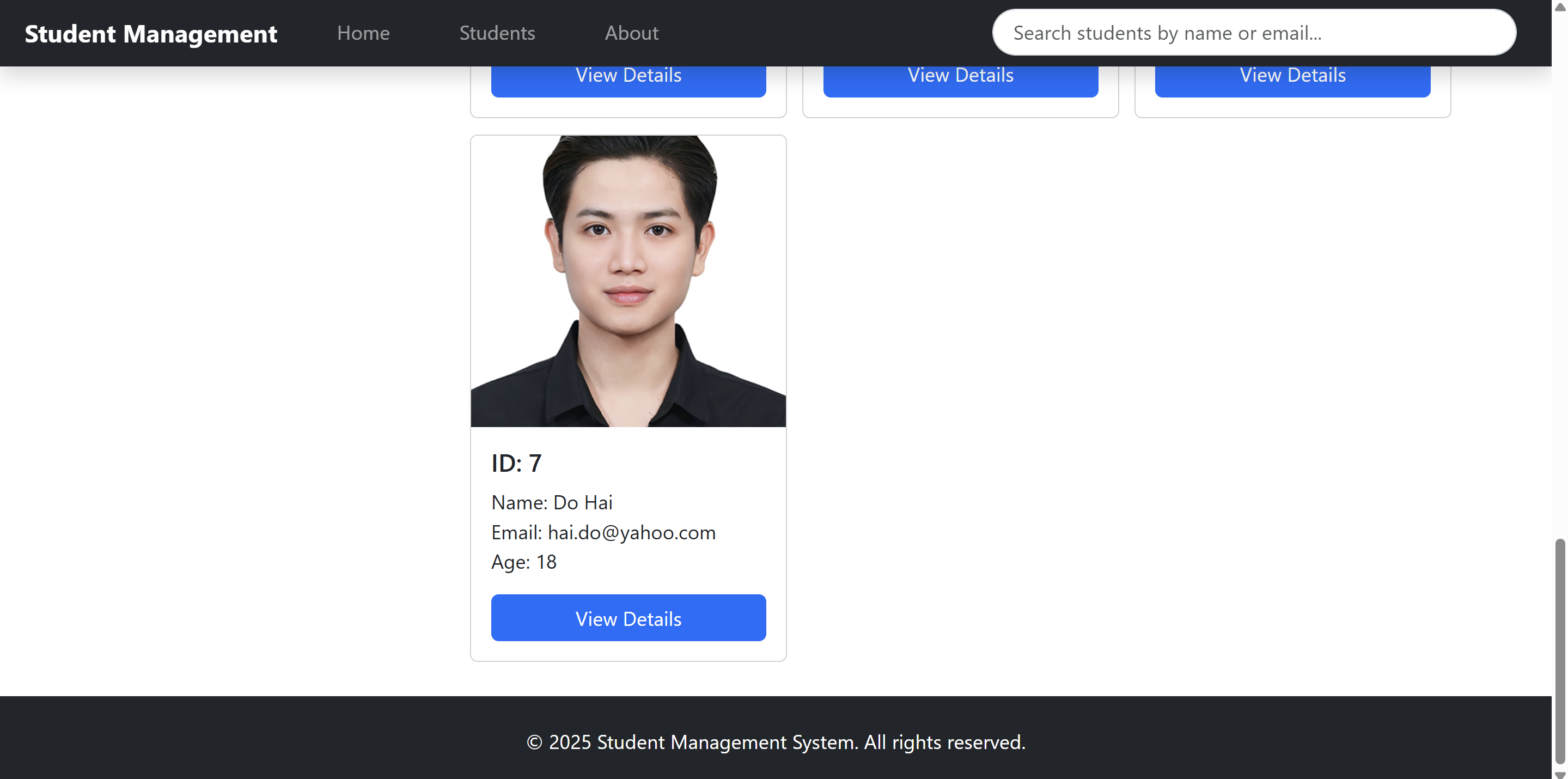
  );

}

export default StudentGrid;

## StudentsPage.js





import React, { useState, useMemo } from 'react';

import { Container, Row, Col } from 'react-bootstrap';

import Navbar from '../components/Navbar';

import Filters from '../components/Filters';

import SortDropdown from '../components/SortDropdown';

import StudentGrid from '../components/StudentGrid';

import StudentDetailModal from '../components/StudentDetailModal';

import { students as initialStudents } from '../data/students';

function StudentsPage() {

  const [searchTerm, setSearchTerm] = useState('');

  const [ageRange, setAgeRange] = useState('all');

  const [hasAvatar, setHasAvatar] = useState(false);

  const [sortOption, setSortOption] = useState('nameAsc');

  const [selectedStudent, setSelectedStudent] = useState(null);

  const filteredStudents = useMemo(() => {

    let result = [...initialStudents];

    if (searchTerm) {

      result = result.filter((s) =>

        s.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

        s.email.toLowerCase().includes(searchTerm.toLowerCase())

      );

    }

    if (hasAvatar) {

      result = result.filter((s) => s.avatar);

    }

    if (ageRange !== 'all') {

      if (ageRange === '<=20') result = result.filter((s) => s.age <= 20);

      else if (ageRange === '21-25') result = result.filter((s) => s.age >= 21 && s.age <= 25);

      else if (ageRange === '>25') result = result.filter((s) => s.age > 25);

    }

    return result.sort((a, b) => {

      if (sortOption === 'ageAsc') return a.age - b.age;

      if (sortOption === 'ageDesc') return b.age - a.age;

      if (sortOption === 'nameAsc') return a.name.localeCompare(b.name);

      if (sortOption === 'nameDesc') return b.name.localeCompare(a.name);

      return 0;

    });

  }, [searchTerm, ageRange, hasAvatar, sortOption]);

  return (

    <>

      <Navbar onQuickSearch={setSearchTerm} />

      <Container className="my-4">

        <Row className="mb-4">

          <Col>

            <h1 className="text-center">Student Management</h1>

            <p className="text-center text-muted">Manage and explore student information efficiently.</p>

          </Col>

        </Row>

        <Row>

          <Col md={3}>

            <div className="p-3 bg-light rounded shadow-sm">

              <Filters

                onFilter={({ searchTerm, ageRange, hasAvatar }) => {

                  setSearchTerm(searchTerm);

                  setAgeRange(ageRange);

                  setHasAvatar(hasAvatar);

                }}

              />

            </div>

          </Col>

          <Col md={9}>

            <div className="mb-3">

              <SortDropdown onSort={setSortOption} />

            </div>

            <StudentGrid students={filteredStudents} onViewDetails={setSelectedStudent} />

          </Col>

        </Row>

      </Container>

      <StudentDetailModal

        show={!!selectedStudent}

        onHide={() => setSelectedStudent(null)}

        student={selectedStudent || {}}

      />

    </>

  );

}

export default StudentsPage;