Redux 3.3_CW Exercises

ex01: configure redux store

challenge

Create a Redux store using Redux Toolkit. Configure reducers for both students and school data.

solution

```
import { configureStore } from '@reduxjs/toolkit'
import { studentsSlice } from '../features/students/studentsSlice'
import { schoolSlice } from '../features/school/schoolSlice'

export default configureStore({
  reducer: {
    students: studentsSlice.reducer,
    school: schoolSlice.reducer,
  },
})
```

COPY

ex02.1: student view - fetching students

challenge

1. Set up the Redux store, including the `studentsSlice` with `fetchStudents` async thunk.
   - Create a Redux store using `createSlice` and `createAsyncThunk`.
   - Define an async thunk action named `fetchStudents` within the `studentsSlice` to fetch student data from an API.
   - Define the initial state for the Redux store, including:
     - An empty array `students` to store student data.
     - A `status` field set to `"idle"` to indicate the initial state.
     - An `error` field initially set to `null`.
2. Create a component (e.g., `StudentView`) to display a list of students.
   - Implement a `useEffect` hook to fetch students when the `StudentView` component mounts. Fetch students only if the `status` in the Redux store is `"idle"` to avoid redundant requests.
   - Create UI elements to handle loading and error states:

    - When fetching students, display a loading message (e.g., "Loading...").
    - If an error occurs while fetching, display an error message along with the error details (e.g., `"Error: {error}"`).

3. Display Student List:

    - Use the `StudentList` component (which will be created separately) to display the list of students.
    - Pass the `students` array from the Redux store as a prop to the `StudentList` component.

solution

```
import { createSlice } from '@reduxjs/toolkit'

import { createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios'

export const fetchStudents = createAsyncThunk(
  'students/fetchStudents',
  async () => {
    const response = await axios.get(
      'https://reduxtoolkit-example-student-management.tanaypratap.repl.co/students',
    )
    console.log(response.data)
    return response.data
  },
)

const initialState = {
  students: [],
  status: 'idle',
  error: null,
}

export const studentsSlice = createSlice({
  name: 'students',
  initialState,
  reducers: {},
  extraReducers: {
    [fetchStudents.pending]: (state) => {
      state.status = 'loading'
    },
    [fetchStudents.fulfilled]: (state, action) => {
      state.status = 'success'
      state.students = action.payload
    },
    [fetchStudents.rejected]: (state, action) => {
      state.status = 'error'
      console.log(action.error.message)
      state.error = action.error.message
    },
  },
})
import React, { useEffect } from 'react'
import StudentList from '../features/students/StudentList'
import { Link } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import { fetchStudents } from '../features/students/studentsSlice'
```

```
const StudentView = () => {
  const dispatch = useDispatch()
  const students = useSelector((state) => state.students.students)
  const status = useSelector((state) => state.students.status)
  const error = useSelector((state) => state.students.error)

  useEffect(() => {
    if (status === 'idle') {
      dispatch(fetchStudents())
    }
  }, [status, dispatch])

  return (
    <div>
      <h1>Student View </h1>

      {status === 'loading' && <p>Loading...</p>}
      {error && <p>Error: {error}</p>}

      <StudentList students={students} />

      <h3>
        <Link to={`/students/add`}>Add student</Link>
      </h3>
    </div>
  )
}

export default StudentView

import React from 'react'
import { Link } from 'react-router-dom'

const StudentList = ({ students }) => {
  return (
    <div>
      <h2>Student List</h2>
      <ul>
        {students.map((student) => (
          <li key={student._id}>
            <Link to={`/students/${student._id}`}>
              {student.name} (Age: {student.age}, Grade: {student.grade})
            </Link>
          </li>
        ))}
      </ul>
    </div>
  )
}

export default StudentList
```

COPY

ex02.2: student view - adding students

challenge

1. Create a form component (e.g., `StudentForm`) for adding new students.
    - Inside the `StudentForm` component, use the `useState` hook to manage the following form input fields:
        - Name
        - Age
        - Grade
        - Gender (as radio buttons)
        - Attendance (if editing an existing student)
        - Marks (if editing an existing student)
2. Use the `useDispatch` hook to dispatch the `addStudentAsync` action when the form is submitted.
    - Implement a `handleSubmit` function that does the following:
        - Creates an object `newStudent` with the values of the form fields.
        - Dispatches the `addStudentAsync` action with `newStudent` as an argument if it's a new student.
        - Dispatches the `updateStudentAsync` action if it's an existing student (updating). [LATER]
3. Implement validation and error handling for the form.
4. After adding a student, update the Redux store with the new student data.

solution

```
import React, { useState } from 'react'
import { useDispatch } from 'react-redux'
import {
  addStudent,
  addStudentAsync,
  updateStudent,
  updateStudentAsync,
} from './studentsSlice'
import { useLocation } from 'react-router-dom'

const StudentForm = () => {
  let { state } = useLocation()

  const student = state ? state : null

  const [name, setName] = useState(student ? student.name : '')
  const [age, setAge] = useState(student ? student.age : '')
  const [grade, setGrade] = useState(student ? student.grade : '')
  const [attendance, setAttendance] = useState(
    student ? student.attendance : '',
  )
  const [marks, setMarks] = useState(student ? student.marks : '')
  const [gender, setGender] = useState(student ? student.gender : 'Male')
  const dispatch = useDispatch()

  const handleSubmit = () => {
    const newStudent = {
      name,
      age,
      grade,
      gender,
      attendance,
      marks,
    }

    if (student) {
      dispatch(
        updateStudentAsync({ id: student._id, updatedStudent: newStudent }),
      )
    } else {
      dispatch(addStudentAsync(newStudent))
    }
  }

  return (
    <div>
      <h2>{student ? 'Edit Student' : 'Add Student'}</h2>
      <input
        type='text'
        placeholder='Name'
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <input
        type='number'
        placeholder='Age'
        value={age}
        onChange={(e) => setAge(e.target.value)}
      />
      <input
        type='text'
        placeholder='Grade'
        value={grade}
        onChange={(e) => setGrade(e.target.value)}
      />
      <div>
        <label>
          Gender:
          <input
            type='radio'
            name='gender'
            value='Male'
            checked={gender === 'Male'}
            onChange={() => setGender('Male')}
          /> Male
        </label>
        <label>
          <input
            type='radio'
            name='gender'
            value='Female'
            checked={gender === 'Female'}
            onChange={() => setGender('Female')}
          />{' '}
          Female
        </label>
      </div>
      {student && (
        <>
          <input
            type='text'
            placeholder='Attendance'
            value={attendance}
            onChange={(e) => setAttendance(e.target.value)}
          />
          <input
            type='text'
            placeholder='Marks'
            value={marks}
            onChange={(e) => setMarks(e.target.value)}
          />
        </>
      )}
      <button onClick={handleSubmit}>{student ? 'Update' : 'Add'}</button>
    </div>
  )
}

export default StudentForm

import { createSlice } from '@reduxjs/toolkit'

import { createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios'

export const addStudentAsync = createAsyncThunk(
  'students/addStudentAsync',
  async (newStudent) => {
    const response = await axios.post(
      'https://reduxtoolkit-example-student-management.tanaypratap.repl.co/students',
      newStudent,
    )
    return response.data
  },
)

const initialState = {
  students: [],
  status: 'idle',
  error: null,
}

export const studentsSlice = createSlice({
  name: 'students',
  initialState,
  reducers: {},
  extraReducers: {
    [addStudentAsync.pending]: (state) => {
      state.status = 'loading'
    },
    [addStudentAsync.fulfilled]: (state, action) => {
      state.status = 'success'
      state.students.push(action.payload)
    },
    [addStudentAsync.rejected]: (state, action) => {
      state.status = 'error'
      state.error = action.error.message
    },
  },
})
```

COPY

ex02.3: student view - updating students

challenge

1. Create StudentDetail Component:
   - Create a new component named `StudentDetail.js`.
2. UseParams Hook:
   - In the `StudentDetail` component, use the `useParams` hook from `react-router-dom` to extract the `id` of the student being viewed or edited.
3. Retrieve Student Data:
   - Utilize the `useSelector` hook to retrieve the student's data from the Redux store. You can find the student with the matching `id` in the array of students.
4. Edit Link:
   - In the `StudentDetail` component, create a link/button labeled "Edit Details." This link should navigate to the `StudentForm` component to edit the student's information. Pass the student's data as state in the link.
5. Edit StudentForm:
   - Modify the `StudentForm` component to handle both adding and editing students.
   - Use the `useLocation` hook from `react-router-dom` to access the state passed via the link when editing.
   - Pre-fill the form fields with the existing student's data if you're editing an existing student.
6. Update Student Data:
   - When editing a student, dispatch the `updateStudentAsync` action with the `id` of the student being edited and the updated student data as arguments.
7. Redux Store Update:
   - In the `studentsSlice`, handle the `updateStudentAsync` action by updating the Redux store with the edited student's data.

solution

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'

import { Link, useParams } from 'react-router-dom'
import { deleteStudentAsync } from './studentsSlice'
```

```
const StudentDetail = () => {
  const { id } = useParams()
  const dispatch = useDispatch()
  const student = useSelector((state) =>
    state.students.students.find((s) => s._id === id),
  )

  if (!student) {
    return <div>Student not found.</div>
  }

  const handleDelete = (id) => {
    dispatch(deleteStudentAsync(id))
  }

  return (
    <div>
      <h2>Student Detail</h2>
      <p>Name: {student.name}</p>
      <p>Age: {student.age}</p>
      <p>Grade: {student.grade}</p>
      <p>Attendance: {student.attendance}</p>
      <p>Marks: {student.marks}</p>
      <Link to={`/students/edit/${student.id}`} state={student}>
        Edit Details
      </Link>
      <button onClick={() => handleDelete(student._id)}>Delete</button>
    </div>
  )
}

export default StudentDetail

import { createSlice } from '@reduxjs/toolkit'

import { createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios'

export const updateStudentAsync = createAsyncThunk(
  'students/updateStudentAsync',
  async ({ id, updatedStudent }) => {
    console.log(id, updatedStudent)
    const response = await axios.put(
      `https://reduxtoolkit-example-student-management.tanaypratap.repl.co/students/${id}`,
      updatedStudent,
    )
    return response.data
  },
)

const initialState = {
  students: [],
  status: 'idle',
  error: null,
}

export const studentsSlice = createSlice({
  name: 'students',
  initialState,
  reducers: {},
  extraReducers: {
    [updateStudentAsync.pending]: (state) => {
      state.status = 'loading'
    },
    [updateStudentAsync.fulfilled]: (state, action) => {
      state.status = 'success'
      const updatedStudent = action.payload
      const index = state.students.findIndex((s) => s.id === updatedStudent.id)
      if (index !== -1) {
        state.students[index] = updatedStudent
      }
    },
    [updateStudentAsync.rejected]: (state, action) => {
      state.status = 'error'
      state.error = action.error.message
    },
  },
})

import React, { useState } from 'react'
import { useDispatch } from 'react-redux'
import {
  addStudent,
  addStudentAsync,
  updateStudent,
  updateStudentAsync,
} from './studentsSlice'
import { useLocation } from 'react-router-dom'

const StudentForm = () => {
  let { state } = useLocation()

  const student = state ? state : null

  const [name, setName] = useState(student ? student.name : '')
  const [age, setAge] = useState(student ? student.age : '')
  const [grade, setGrade] = useState(student ? student.grade : '')
  const [attendance, setAttendance] = useState(
    student ? student.attendance : '',
  )
  const [marks, setMarks] = useState(student ? student.marks : '')
  const [gender, setGender] = useState(student ? student.gender : 'Male')
  const dispatch = useDispatch()

  const handleSubmit = () => {
    const newStudent = {
      name,
      age,
      grade,
      gender,
      attendance,
      marks,
    }

    if (student) {
      dispatch(
        updateStudentAsync({ id: student._id, updatedStudent: newStudent }),
      )
    } else {
      dispatch(addStudentAsync(newStudent))
    }
```

```jsx
  }

  return (
    <div>
      <h2>{student ? 'Edit Student' : 'Add Student'}</h2>
      <input
        type='text'
        placeholder='Name'
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <input
        type='number'
        placeholder='Age'
        value={age}
        onChange={(e) => setAge(e.target.value)}
      />
      <input
        type='text'
        placeholder='Grade'
        value={grade}
        onChange={(e) => setGrade(e.target.value)}
      />
      <div>
        <label>
          Gender:
          <input
            type='radio'
            name='gender'
            value='Male'
            checked={gender === 'Male'}
            onChange={() => setGender('Male')}
          /> Male
        </label>
        <label>
          <input
            type='radio'
            name='gender'
            value='Female'
            checked={gender === 'Female'}
            onChange={() => setGender('Female')}
          />{' '}
          Female
        </label>
      </div>
      {student && (
        <>
          <input
            type='text'
            placeholder='Attendance'
            value={attendance}
            onChange={(e) => setAttendance(e.target.value)}
          />
          <input
            type='text'
            placeholder='Marks'
            value={marks}
            onChange={(e) => setMarks(e.target.value)}
          />
        </>
      )}
      <button onClick={handleSubmit}>{student ? 'Update' : 'Add'}</button>
    </div>
  )
}

export default StudentForm
```

COPY

ex02.4: student view - deleting students

challenge:

1. Create Delete Button:
   - In the `StudentDetail` component, create a "Delete" button.
2. Use useDispatch Hook:
   - Inside the `StudentDetail` component, use the `useDispatch` hook from `react-redux` to access the dispatch function.
3. Dispatch deleteStudentAsync:
   - Implement an event handler for the "Delete" button click.
   - Dispatch the `deleteStudentAsync` action with the `id` of the student being deleted as an argument.
4. Update Redux Store:
   - In the `studentsSlice`, handle the `deleteStudentAsync` action by removing the deleted student from the state. Use the `filter` method to filter out the student with the matching `id`.
5. Create Navigation Links:
   - In the `StudentList` component, map over the list of students and create navigation links to the `StudentDetail` component for each student.
   - These links should navigate to the `StudentDetail` component with the `id` of the respective student in the URL.

solution

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'

import { Link, useParams } from 'react-router-dom'
import { deleteStudentAsync } from './studentsSlice'

const StudentDetail = () => {
  const { id } = useParams()
  const dispatch = useDispatch()
  const student = useSelector((state) =>
    state.students.students.find((s) => s._id === id),
  )

  if (!student) {
    return <div>Student not found.</div>
  }

  const handleDelete = (id) => {
    dispatch(deleteStudentAsync(id))
  }

  return (
    <div>
      <h2>Student Detail</h2>
      <p>Name: {student.name}</p>
      <p>Age: {student.age}</p>
      <p>Grade: {student.grade}</p>
      <p>Attendance: {student.attendance}</p>
      <p>Marks: {student.marks}</p>
      <Link to={`/students/edit/${student.id}`} state={student}>
        Edit Details
      </Link>
      <button onClick={() => handleDelete(student._id)}>Delete</button>
    </div>
  )
}

export default StudentDetail

import { createSlice } from '@reduxjs/toolkit'

import { createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios'

export const deleteStudentAsync = createAsyncThunk(
  'students/deleteStudentAsync',
  async (id) => {
    const response = await axios.delete(
      `https://reduxtoolkit-example-student-management.tanaypratap.repl.co/students/${id}`,
    )
    return response.data
  },
)

const initialState = {
  students: [],
  status: 'idle',
  error: null,
}

export const studentsSlice = createSlice({
  name: 'students',
  initialState,
  reducers: {},
  extraReducers: {
    [deleteStudentAsync.pending]: (state) => {
      state.status = 'loading'
    },
    [deleteStudentAsync.fulfilled]: (state, action) => {
      state.status = 'success'
      state.students = state.students.filter(
        (student) => student.id !== action.payload.id,
      )
    },
    [deleteStudentAsync.rejected]: (state, action) => {
      state.status = 'error'
      state.error = action.error.message
    },
  },
})
```

COPY

ex03: create class view component

challenge

1. Create Class View Component.
2. Create `setFilter` and `setSortBy` actions in your students slice and in the initialState add `filter: "All"` and `sortBy: "name"` properties.
3. Filter Students:
   - Implement a filtering mechanism for students based on gender. Create a variable `filteredStudents` that filters the students based on the selected `filter` value.
   - The filter options should include "All," "Boys," and "Girls."
4. Sort Students:
   - Implement sorting functionality for students based on the selected `sortBy` value. Create a variable `sortedStudents` that sorts the `filteredStudents` array accordingly.
   - The sorting options should include "Name," "Marks," and "Attendance."
5. Handle Filter Change:
   - Create an event handler function, such as `handleFilterChange`, that dispatches the `setFilter` action when the filter dropdown selection changes.
6. Handle Sort Change:
   - Create an event handler function, such as `handleSortChange`, that dispatches the `setSortBy` action when the sort dropdown selection changes.
7. Render UI:
   - Render the following elements in your `ClassView` component:
     - An `<h1>` element with the text "Class View."
     - A dropdown for filtering students by gender with options "All," "Boys," and "Girls." Bind this dropdown to the `filter` value and use the `handleFilterChange` event handler.
     - A dropdown for sorting students with options "Name," "Marks," and "Attendance." Bind this dropdown to the `sortBy` value and use the `handleSortChange` event handler.
     - A list of students rendered within a `<ul>` element. Map through the `sortedStudents` array and display student information, including name, gender, marks, and attendance,

in `<li>` elements.

solution

```
import React from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { setFilter, setSortBy } from '../features/students/studentsSlice'

const ClassView = () => {
  const students = useSelector((state) => state.students.students)
  const filter = useSelector((state) => state.students.filter)
  const sortBy = useSelector((state) => state.students.sortBy)
  const dispatch = useDispatch()

  const filteredStudents = students.filter((student) => {
    if (filter === 'All') return true
    return student.gender === filter
  })

  const sortedStudents = [...filteredStudents].sort((a, b) => {
    if (sortBy === 'name') return a.name.localeCompare(b.name)
    if (sortBy === 'marks') return b.marks - a.marks
    if (sortBy === 'attendance') return b.attendance - a.attendance
    return 0
  })

  const handleFilterChange = (e) => {
    dispatch(setFilter(e.target.value))
  }

  const handleSortChange = (e) => {
    dispatch(setSortBy(e.target.value))
  }

  return (
    <div>
      <h1>Class View</h1>
      <div>
        <label htmlFor='filter'>Filter by Gender:</label>
        <select id='filter' onChange={handleFilterChange} value={filter}>
          <option value='All'>All</option>
          <option value='Male'>Boys</option>
          <option value='Female'>Girls</option>
        </select>
      </div>
      <div>
        <label htmlFor='sortBy'>Sort by:</label>
        <select id='sortBy' onChange={handleSortChange} value={sortBy}>
          <option value='name'>Name</option>
          <option value='marks'>Marks</option>
          <option value='attendance'>Attendance</option>
        </select>
      </div>
      <div>
        <ul>
          {sortedStudents.map((student) => (
            <li key={student.id}>
              {student.name} - {student.gender} - Marks: {student.marks} -
              Attendance: {student.attendance}
          </li>
        ))}
      </ul>
    </div>
  </div>
 )
}

export default ClassView

import { createSlice } from "@reduxjs/toolkit";

import { createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";

const initialState = {
  students: [],
  status: "idle",
  error: null,
  filter: "All",
  sortBy: "name"
};

export const studentsSlice = createSlice({
  name: "students",
  initialState,
  reducers: {
    setFilter: (state, action) => {
      state.filter = action.payload;
    },
    setSortBy: (state, action) => {
      state.sortBy = action.payload;
    }
  },
  extraReducers: {...}
});

export const { setFilter, setSortBy } = studentsSlice.actions;

export default studentsSlice.reducer;
```

COPY

ex04: create school view component

challenge

1. Create School View Component.
2. Calculate School Statistics:
    - Inside the `useEffect` hook, calculate the following school statistics based on the students' data:
        - Total number of students in the school.
        - Average attendance (calculated as the sum of all students' attendance divided by the total number of students).

    - Average marks (calculated as the sum of all students' marks divided by the total number of students).
    - The top-performing student (student with the highest marks).

3. Dispatch Actions:

    - Dispatch the `updateSchoolStats` action to update the school statistics in the Redux store. Pass an object containing the calculated statistics (totalStudents, averageAttendance, averageMarks, topStudent) as payload.

4. Set Top Student:

    - Dispatch the `setTopStudent` action to set the top-performing student in the Redux store.

5. Render UI:

    - Render the following elements in your `SchoolView` component:

        - An `<h1>` element with the text "School View."
        - Display the total number of students.
        - Display the average attendance (rounded to two decimal places).
        - Display the average marks (rounded to two decimal places).
        - Display the name of the top-performing student or "-" if there is no top student.

solution

```
import { createSlice } from '@reduxjs/toolkit'

const initialState = {
  totalStudents: 0,
  averageAttendance: 0,
  averageMarks: 0,
  topStudent: null,
}

export const schoolSlice = createSlice({
  name: 'school',
  initialState,
  reducers: {
    updateSchoolStats: (state, action) => {
      const { totalStudents, averageAttendance, averageMarks, topStudent } =
        action.payload

      state.totalStudents = totalStudents
      state.averageAttendance = averageAttendance
      state.averageMarks = averageMarks
      state.topStudent = topStudent
    },
    setTopStudent: (state, action) => {
      state.topStudent = action.payload
    },
  },
})

export const { updateSchoolStats, setTopStudent } = schoolSlice.actions
export default schoolSlice.reducer

import React, { useEffect } from 'react'
import { useSelector, useDispatch } from 'react-redux'
import {
  setTopStudent,
  updateSchoolStats,
} from '../features/school/schoolSlice'

const SchoolView = () => {
  const schoolStats = useSelector((state) => state.school)
  const students = useSelector((state) => state.students.students)
  const dispatch = useDispatch()

  useEffect(() => {
    const totalStudents = students.length
    const totalAttendance = students.reduce(
      (sum, student) => sum + parseFloat(student.attendance),
      0,
    )
    const averageAttendance = totalAttendance / totalStudents
    const totalMarks = students.reduce(
      (sum, student) => sum + parseFloat(student.marks),
      0,
    )
    const averageMarks = totalMarks / totalStudents

    const topStudent = students.reduce((prev, current) => {
      return parseFloat(current.marks) > parseFloat(prev.marks) ? current : prev
    }, '')

    dispatch(
      updateSchoolStats({
        totalStudents,
        averageAttendance,
        averageMarks,
        topStudent,
      }),
    )

    dispatch(setTopStudent(topStudent))
  }, [students, dispatch])

  return (
    <div>
      <h1>School View</h1>
      <p>Total Students: {schoolStats.totalStudents}</p>
      <p>Average Attendance: {schoolStats.averageAttendance.toFixed(2)}</p>
      <p>Average Marks: {schoolStats.averageMarks.toFixed(2)}</p>
      <p>
        Top Student:{' '}
        {schoolStats.topStudent ? schoolStats.topStudent.name : '-'}
      </p>
    </div>
  )
}

export default SchoolView
```

COPY

homework

Extend the existing school management application by adding CRUD operations for teachers. i.e. to add a teacher, delete teacher, show a list of teacher and based on that display school-wide statistics and information in the SchoolView component.

ex5: integrate components in app

challenge

Integrate the Class View, Student View, and School View components into the App component. Set up routing using React Router.

solution

```
import ClassView from './components/ClassView'
import './styles.css'
import StudentView from './components/StudentView'
import SchoolView from './components/SchoolView'

import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom'
import './styles.css'
import StudentDetail from './features/students/StudentDetail'
import StudentForm from './features/students/StudentForm'

export default function App() {
  return (
    <div className='App'>
      <Router>
        <div>
          <div className='navbar'>
            <div className='logo'>Student Management System</div>
            <nav>
              <ul>
                <li>
                  <Link to='/'>Students</Link>
                </li>
                <li>
                  <Link to='/classes'>Classes</Link>
                </li>
                <li>
                  <Link to='/school'>School</Link>
                </li>
              </ul>
            </nav>
          </div>

          <Routes>
            <Route path='/school' element={<SchoolView />} />
            <Route path='/classes' element={<ClassView />} />
            <Route path='/' element={<StudentView />} />
            <Route path='/students/:id' element={<StudentDetail />} />
          <Route path='/students/add' element={<StudentForm />} />
          <Route path='/students/edit/:id' element={<StudentForm />} />
        </Routes>
      </div>
    </Router>
  </div>
  )
}

import { StrictMode } from 'react'
import ReactDOM from 'react-dom'
import store from './app/store'
import { Provider } from 'react-redux'
import App from './App'

console.log(store.getState())

const rootElement = document.getElementById('root')
ReactDOM.render(
  <StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </StrictMode>,
  rootElement,
)
```

COPY

entire solution

https://codesandbox.io/s/redux-toolkit-student-management-app-mvqdlk

backend solution

https://replit.com/@tanaypratap/reduxtoolkit-example-student-management