CS545 Web Application Architecture
Final Exam

Dr. Muhyieddin Al-Taranweh

Id:	Name:

The exam takes 180 minutes.
Please read the exam policy before you start the exam.
There is no tolerance policy for academic dishonesty on exams. **You will be asked to leave the exam room immediately without a warning** if you do the following things which means you'll get an **NC**.

1. You are caught cheating or trying to cheat.
2. All mobile phones should be turned off and stored with your coat or backpack.
3. You are not expected to make any changes on the backend.

You need to develop a part of React project for the Student Management Application. Assume having an application with a Restful API for the Student ManagementAapplication.

Based on the projects from the following repositories:

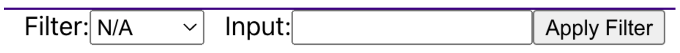
https://github.com/muhyidean/SecondExam-Frontend.git
https://github.com/muhyidean/SecondExam-Backend.git

You must build a Single-Page-Application by using **React Router (V6).** You need to build the following components:
1. Student 2. Students 3. StudentDetails 4. AddStudent 5. SelectedStudent 6. Dashboard 7.Header 8.Course 9. App

1. Student: This component will display two fields of the student object. (*Name* and *ID).* It should be clickable that is associated with a link to display the student details.

Clicking on the **Student** component will route to the **StudentDetails** (separate page) component that displays all fields of the **Student** object.

2. Students: You need to create this component to render **Student** components and have a filtering option *(Dropdown list, text field, button)*



3. StudentDetails: This component will be in a separate page. It should be a child component in **Students** and it will display full information about a student (*Id, Name, Gpa, Courses*) and contain the following:

i. *Delete* *-> this will delete the selected student and update the displayed students*

ii. *Select / Unselect* *-> this will add the current student to a stateful array that will maintain a list of selected students during execution. (No database calls to implement this)*

iii. *Back* *-> this will navigate back to the displayed students*

4. AddStudent: This component will be in a separate page and takes the input from the user and sends the data to the server, then it should navigate to the **Students** component when submitted.

5. SelectedStudents: This component will be in a separate page and it should read the students that were selected and render them in a list showing the (Id, Name) and an option to '**Unselect**' (*You can also implement this in two components*).

6. Dashboard: This component has the **Header** component and contains the required routes for all the components (**Students, StudentDetails, AddStudent, SelectedStudents**).

iv. When displaying selected students list, you might consider using **React Context**.

v. The default route ('/') should render the **Students** component

7. Header: This component will be holding three links that will go the following pages:

vi. Students
vii. Add Student
viii. Selected Students

8. Course: This component will be used to render every course in the student's courses. If the student has courses, they should be printed out in the **StudentDetails** component. If there are no courses, you should display "`Term status: inactive`". Refer to the images for this part.

9. App: This component contains the **Dashboard** component.

BackEnd Endpoints

- **GET** request to '**localhost:8080/api/v1/students**' returns all the Students with their courses in the following JSON object:

```
[
    {
        "id": 111,
        "name": "Zaineh",
        "gpa": 3.6,
        "courseList": [
            {
                "id": 201,
                "name": "Procedural Programming",
                "program": "BSC"
            },
            {
                "id": 221,
                "name": "Data Structures",
                "program": "BSC"
            },
            {
                "id": 545,
                "name": "Web Application Architecture",
                "program": "MSC"
            }
        ]
    },
{
        "id": 112,
        "name": "Yasmeen",
           …
           …
           …
```

- **GET** request to '**localhost:8080/api/v1/students /{id}**' returns all information for one specific student with the following JSON object:

```
    {
        "id": 111,
        "name": "Zaineh",
        "gpa": 3.6,
        "courseList": [
            {
                "id": 201,
                "name": "Procedural Programming",
                "program": "BSC"
            },
            {
                "id": 221,
                "name": "Data Structures",
                "program": "BSC"
            },
            {
                "id": 545,
                "name": "Web Application Architecture",
                "program": "MSC"
            }
        ]
    },
```

- **POST** request to '**localhost:8080/api/v1/students**' persists a Student object.
  Request Body ->

```
{
"name":"john",
"gpa":3.4.      }
```

- **PUT** request to '**localhost:8080/api/v1/students /{id}**' updates a Student object.
- **DELETE** request to '**localhost:8080/api/v1/students /{id}**' removes a Student object.
- **There are two options for filtering when adding request parameters:**
  - **By GPA less than**: This will return all the students that have a GPA lower than the given value.
    GET 'localhost:8080/api/v1/students?filter=gpa&input=3.5'
  - **By program level**: This will return all the students that at least take one course within a given program (MSC or BSC).
    GET 'localhost:8080/api/v1/students?filter=program&input=MSC'

Help from your professor:

* When you read the values from the filtering options. You can add the request parameters to the call using the following: *Even if there are no parameters, it will work.*

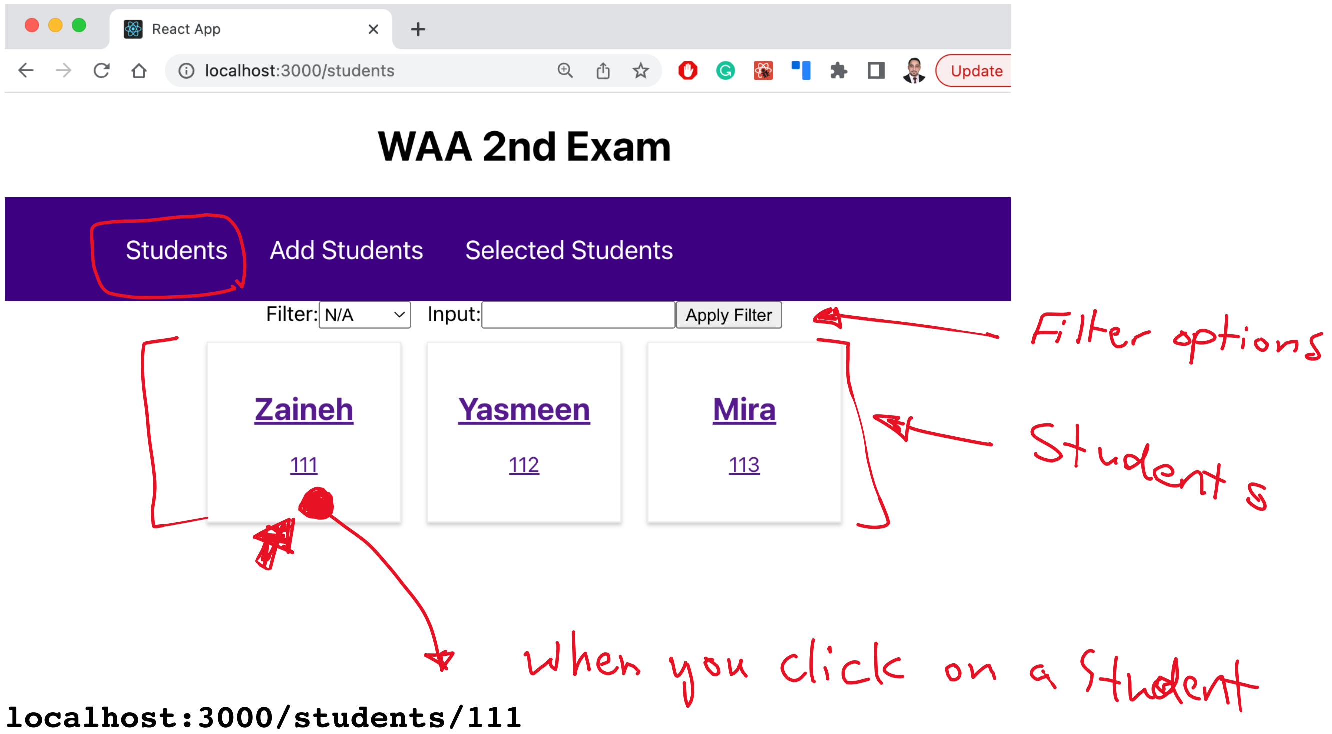
```
  axios.get('http://localhost:8080/api/v1/students', {
            params: {
                filter: // [PLACE DROPDOWN DATA HERE…]
                input: // [PLACE INPUT DATA HERE…]
            }
        })
```

* You can use this for the dropdown. (Don't forget to create a useRef() to link it.)

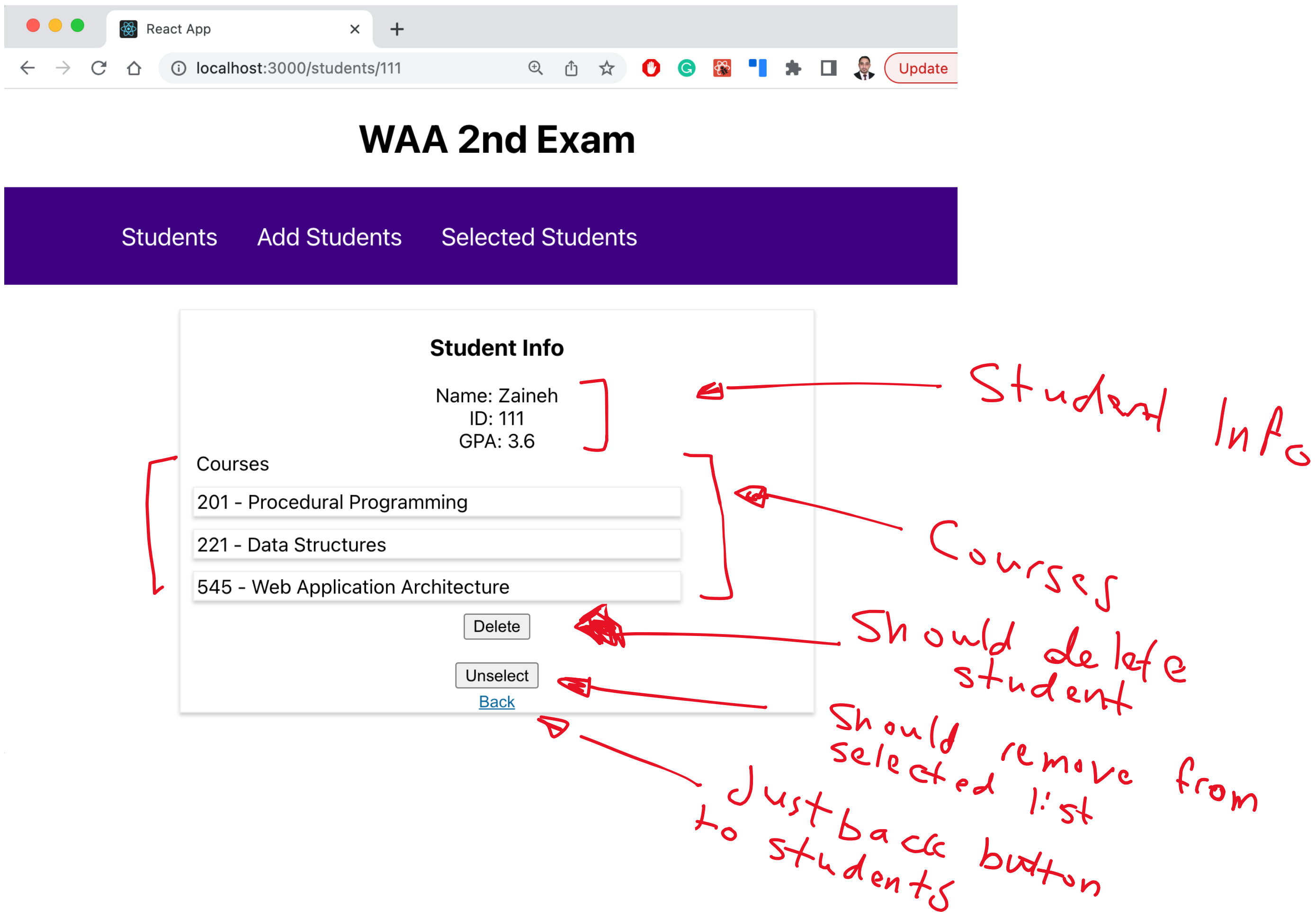
```
<select ref={dropdown}>
                <option value="0">N/A</option>
                <option value="gpa">&lt; gpa</option>
                <option value="program">program</option>
            </select>
```

For more clarification, refer to the following screenshots. (You are NOT obliged to implement the same design!)

`localhost:3000/students`



`localhost:3000/students/111`



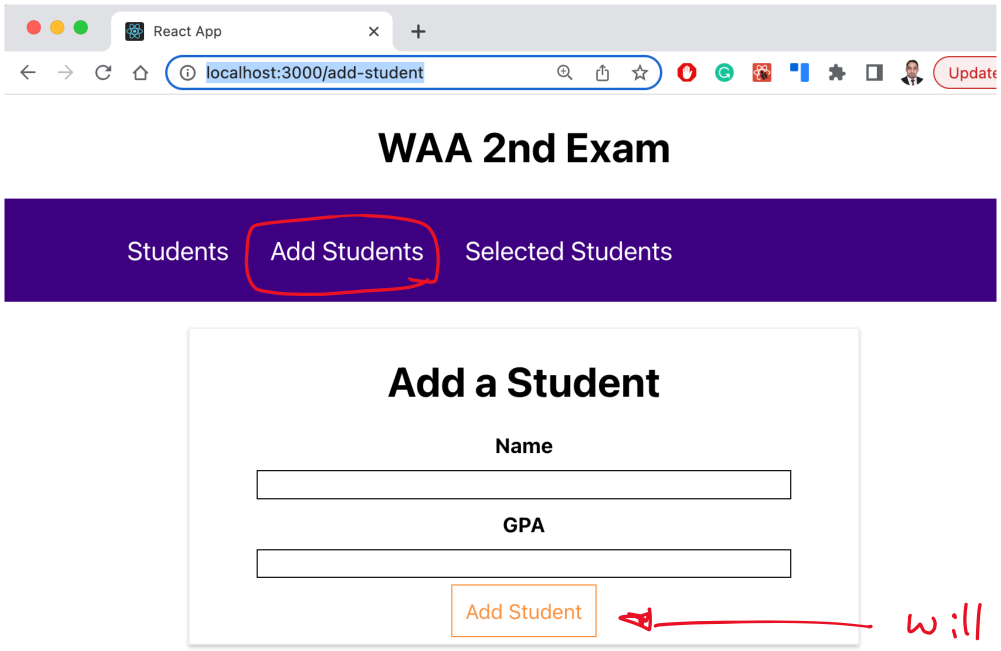
Student Info

Name: Zaineh
ID: 111
GPA: 3.6

Courses

201 - Procedural Programming
221 - Data Structures
545 - Web Application Architecture

localhost:3000/add-student



WAA 2nd Exam

Add a Student

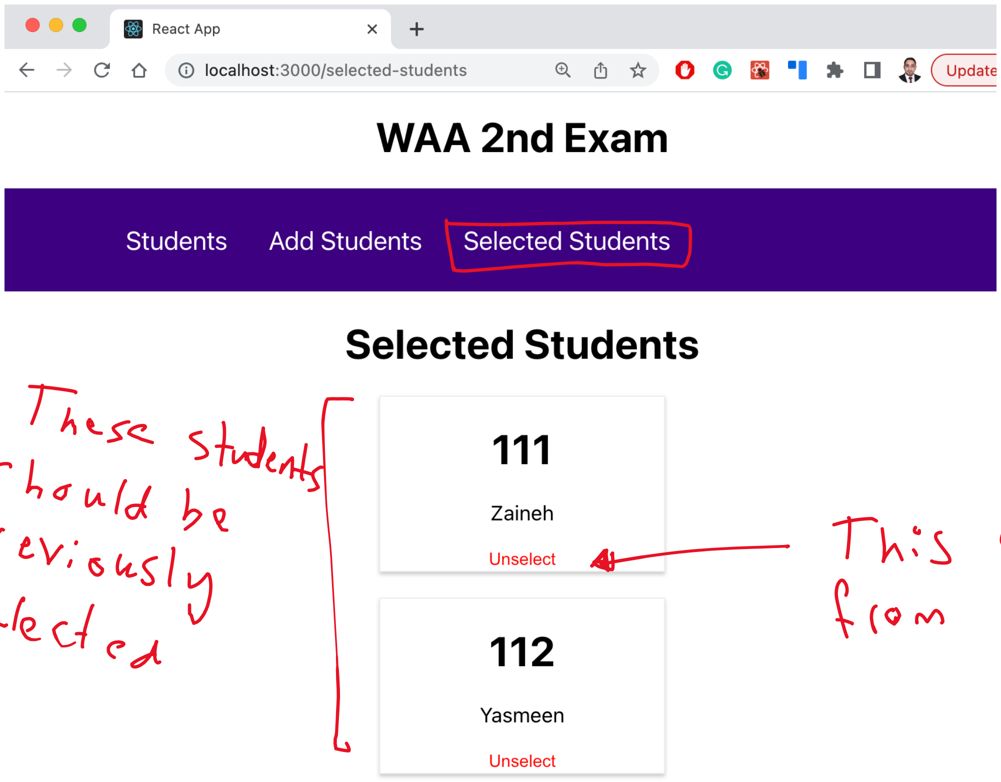
Name

GPA

Add Student

will make a POST taking the given data

localhost:3000/selected-students



WAA 2nd Exam

Selected Students

111

Zaineh

Unselect

112

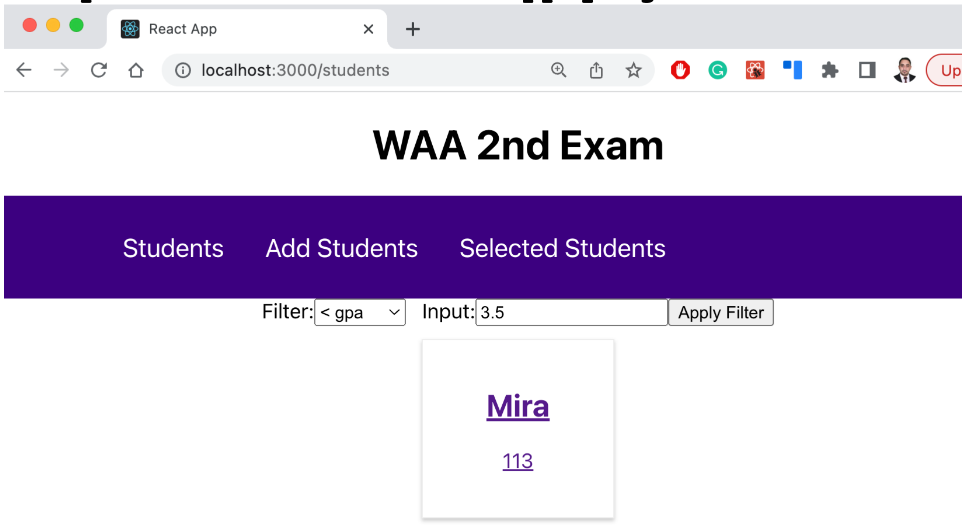
Yasmeen

Unselect

These students should be previously selected

This will remove from selected list

Example of students after applying filter:



React App

localhost:3000/students

WAA 2nd Exam

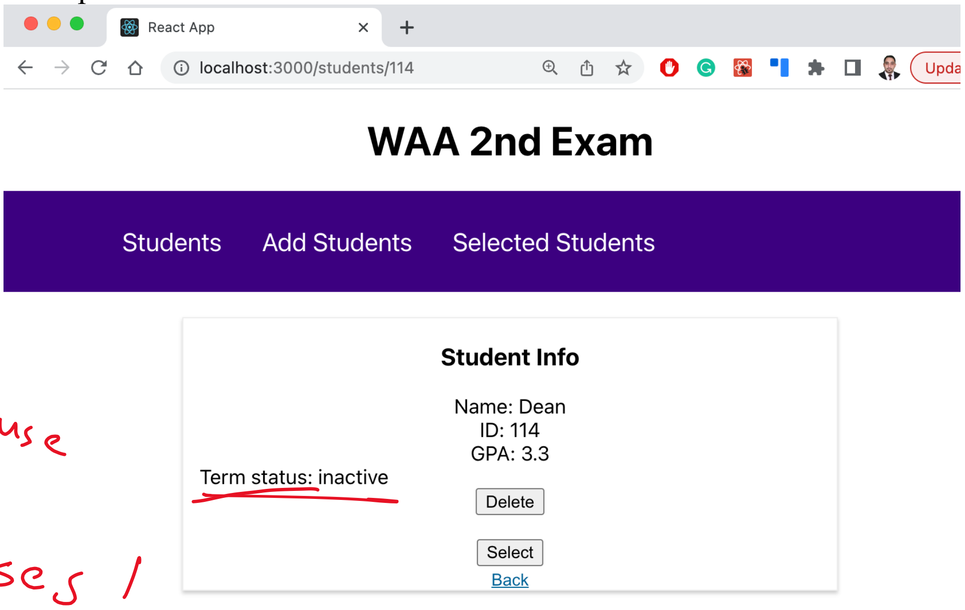
Students Add Students Selected Students

Filter: < gpa Input: 3.5 Apply Filter

Mira

113

Example of student details with no courses



React App

localhost:3000/students/114

WAA 2nd Exam

Students Add Students Selected Students

Student Info

Name: Dean
ID: 114
GPA: 3.3

Term status: inactive

Delete

Select

Back

Cause
No
courses!