Problem 3 – Course Planner

Working with Remote Data

For the solution of some of the following tasks, you will need to use an up-to-date version of the **local REST service** provided in the lesson's resources archive. You can read the documentation here.

Environment Specifics

Please be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

The following actions are **NOT** supported:

- `.forEach()` with **NodeList** (returned by `querySelector()` and `querySelectorAll()`)
- `.forEach()` with **HTMLCollection** (returned by `getElementsByClassName()` and `element.children`)
- using the **spread-operator** (`...`) to convert a **NodeList** into an array
- `append()` (use only `appendChild()`)
- `prepend()`
- `replaceWith()`
- `replaceAll()`
- `closest()`
- `replaceChildren()`

If you want to perform these operations, you may use `Array.from()` to first convert the collection into an array.

Requirements

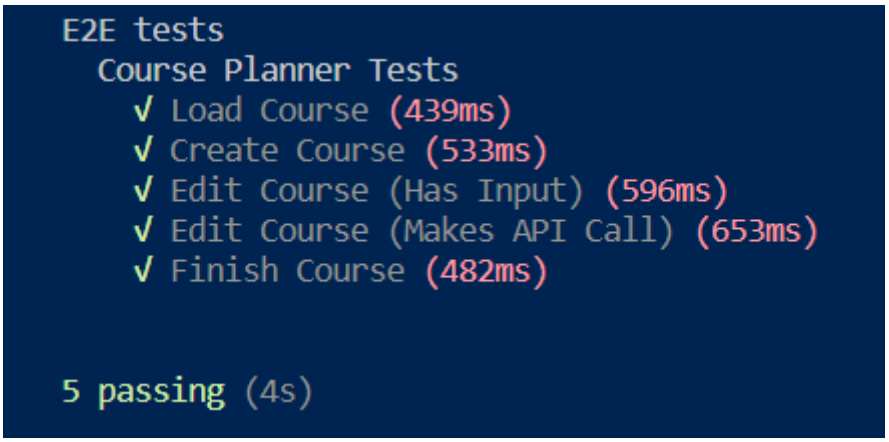
Write a JS program that can load, create, remove and edit a list of courses. You will be given an HTML template to which you must bind the needed functionality.

First, you need to install all dependencies using the **`npm install`** command

Then, you can start the front-end application with the **`npm start`** command

You also must start the ***server.js*** file in the ***server*** folder using the **`node server.js`** command in another console **(BOTH THE CLIENT AND THE SERVER MUST RUN AT THE SAME TIME)**.

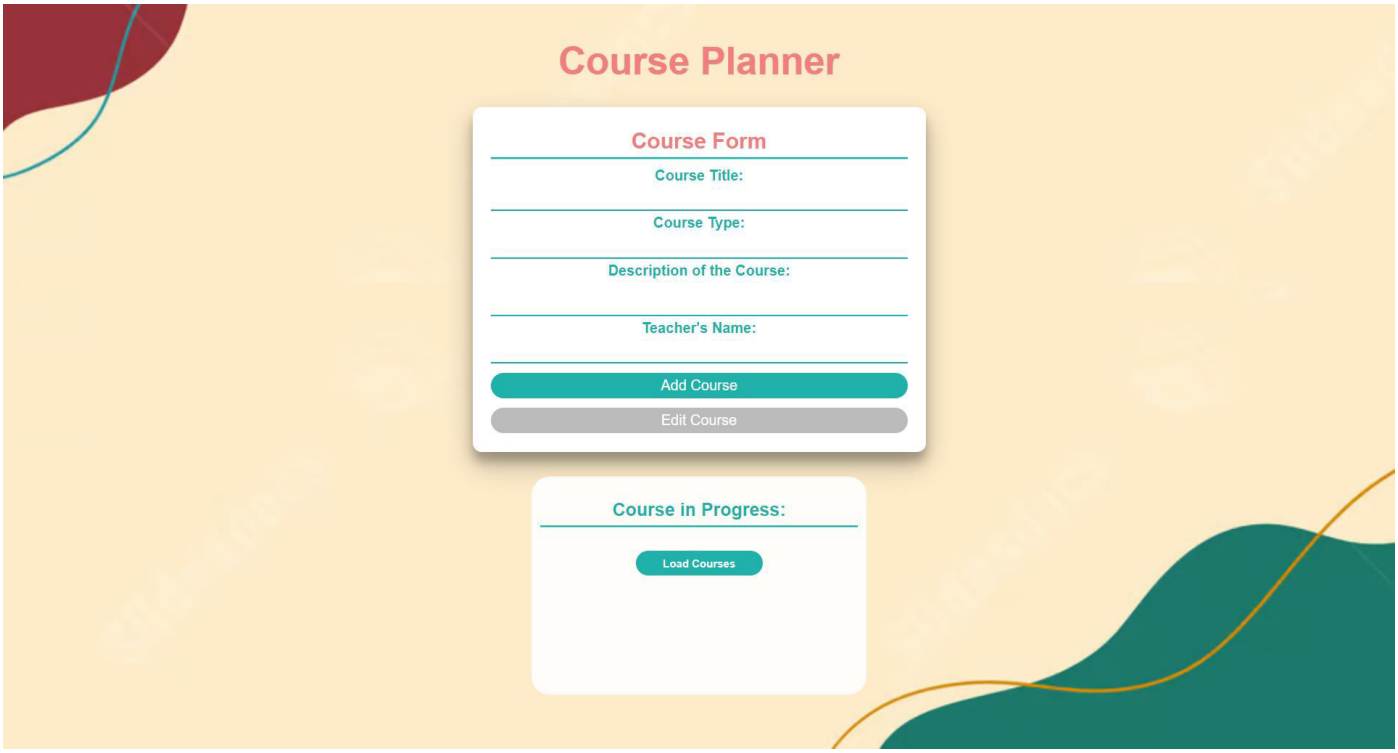
At any point, you can open up another console and run **`npm test`** to test the **current state** of your application. It's preferable for **all of your tests to pass locally** before you submit to the Judge platform, like this:



Endpoints

- http://localhost:3030/jsonstore/tasks/
- http://localhost:3030/jsonstore/tasks/:id

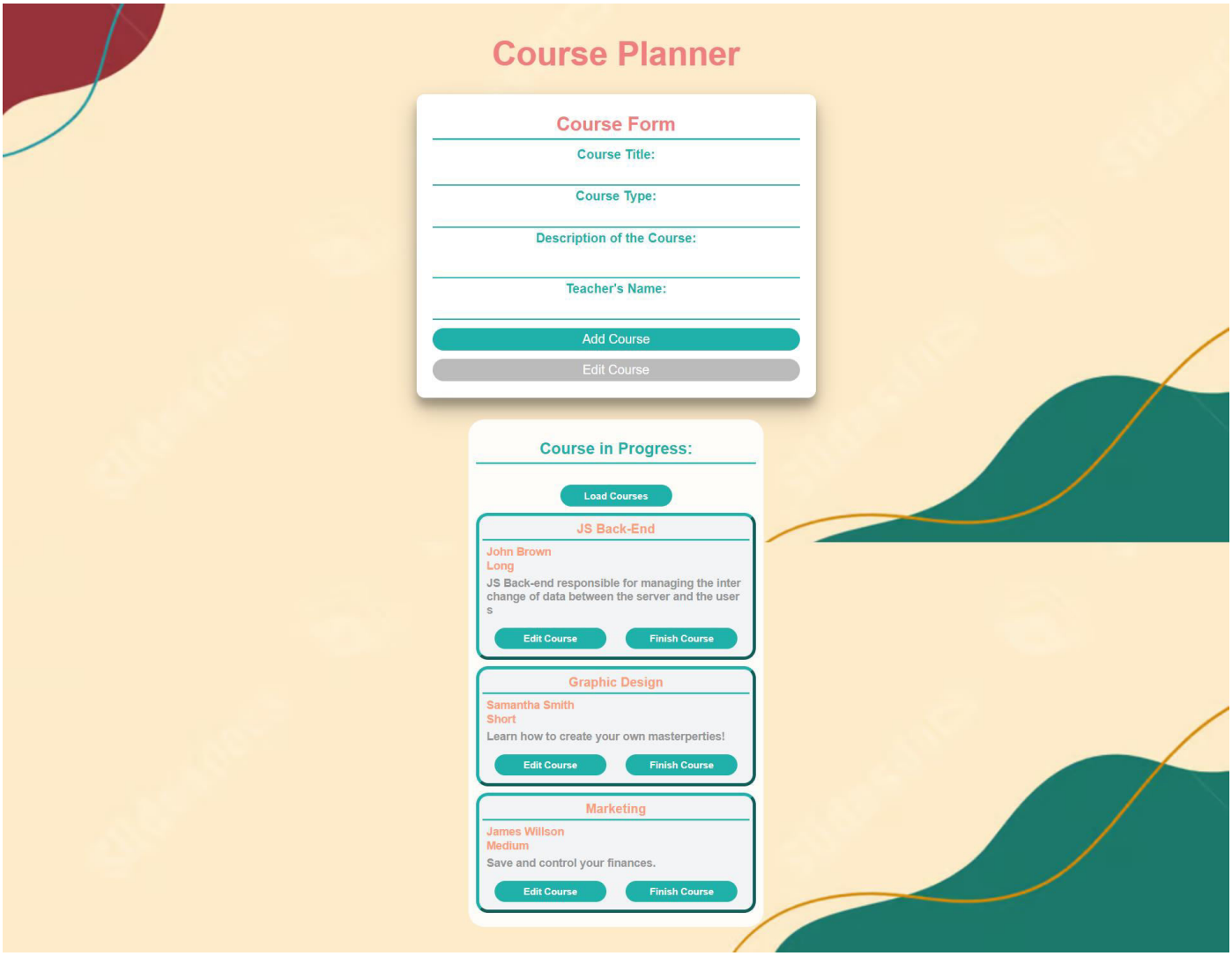
Load Courses



Clicking the **[Load Courses]** button should send a **GET** request to the server to fetch **all courses** from your local database. You must add each task to the **`<div>`** with **`id="list"`. [Edit Course]** button should be deactivated.

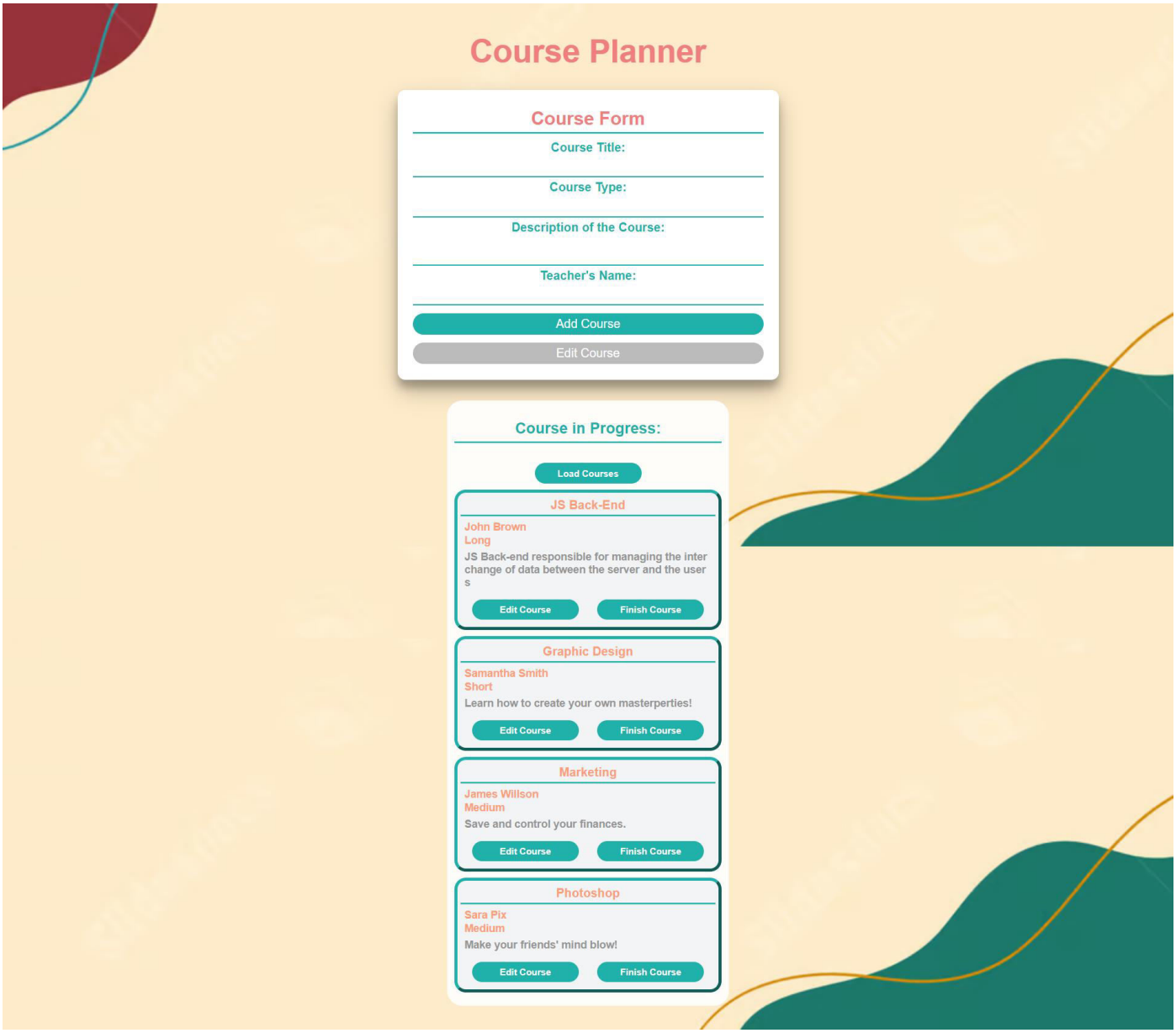
Each course has the following **HTML structure**:

```
<div class="container">
  <h2>JS Back-End</h2>
  <h3>John Brown</h3>
  <h3>Long</h3>
  <h4>
    "JS Back-end responsible for managing the interchange of data between the server and the users"
  </h4>
  <button class="edit-btn">Edit Course</button>
  <button class="finish-btn">Finish Course</button>
</div>
```



Add a Course

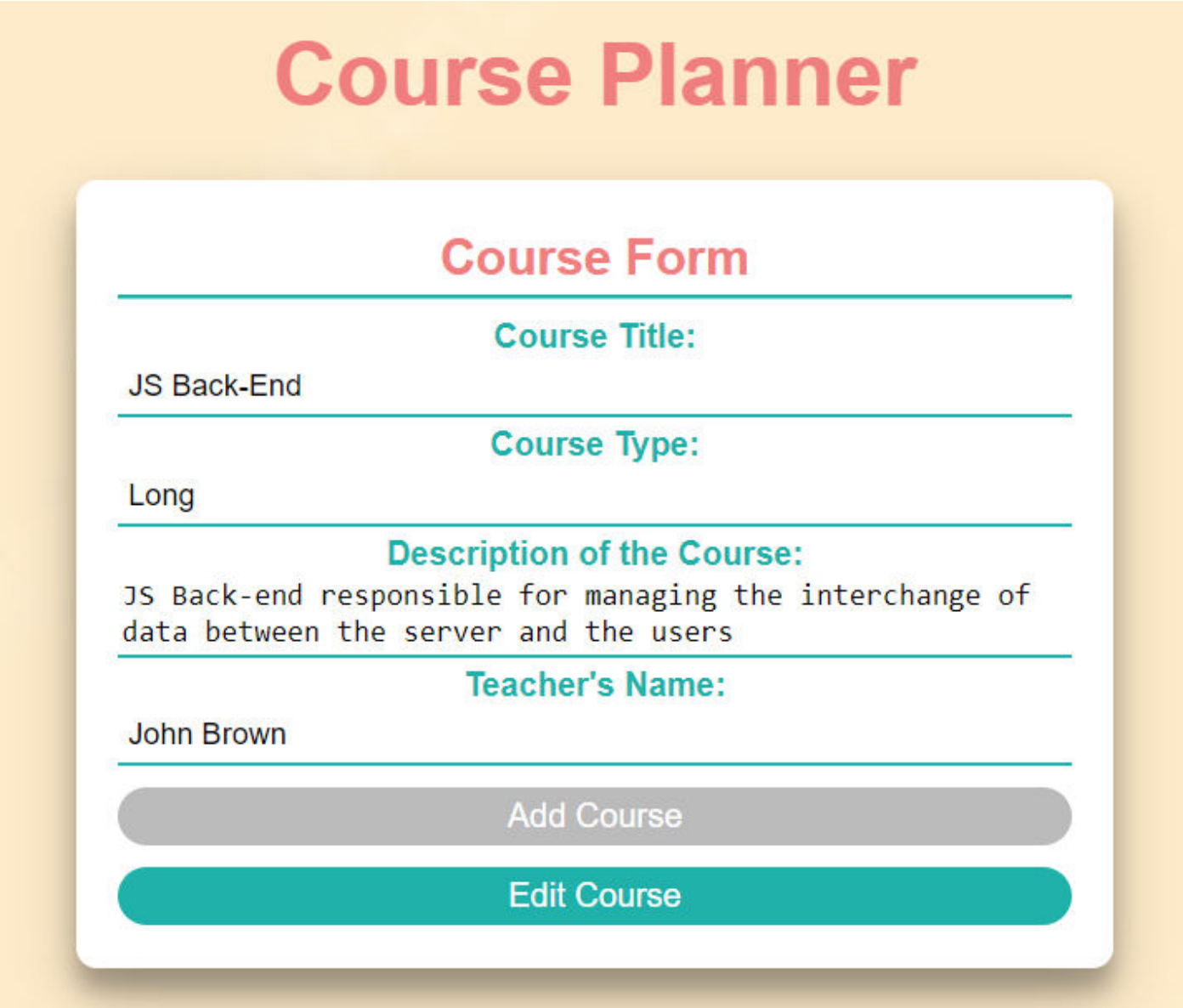
Clicking the **[Add Course]** button should send a **POST** request to the server, creating a new course with the **title**, **type** (“Long”, “Medium”, or “Short”), **description, , and the teacher's name** from the input values. After a successful creation, you should send another **GET** request to fetch all the courses, including the **newly added one** into the **Course Progress** column. You should also **clear all the input fields** after the creation!



Edit a Course

Clicking the **[Edit Course]** button on a record should remove the record from the DOM structure and the information about the task should be populated into the input fields above. The **[Edit Course]** button in the form should be activated and the **[Add Course]** one should be deactivated.

After clicking the [**Edit Course]** button in the form, you should send a **PUT** request to the server to **modify the title, type, description, and the teacher's name** of the changed item. After the successful request, you should **fetch the items again** and see that the changes have been made. After that, the **[Edit Course]** button should be deactivated and the **[Add Course]** one should be activated.

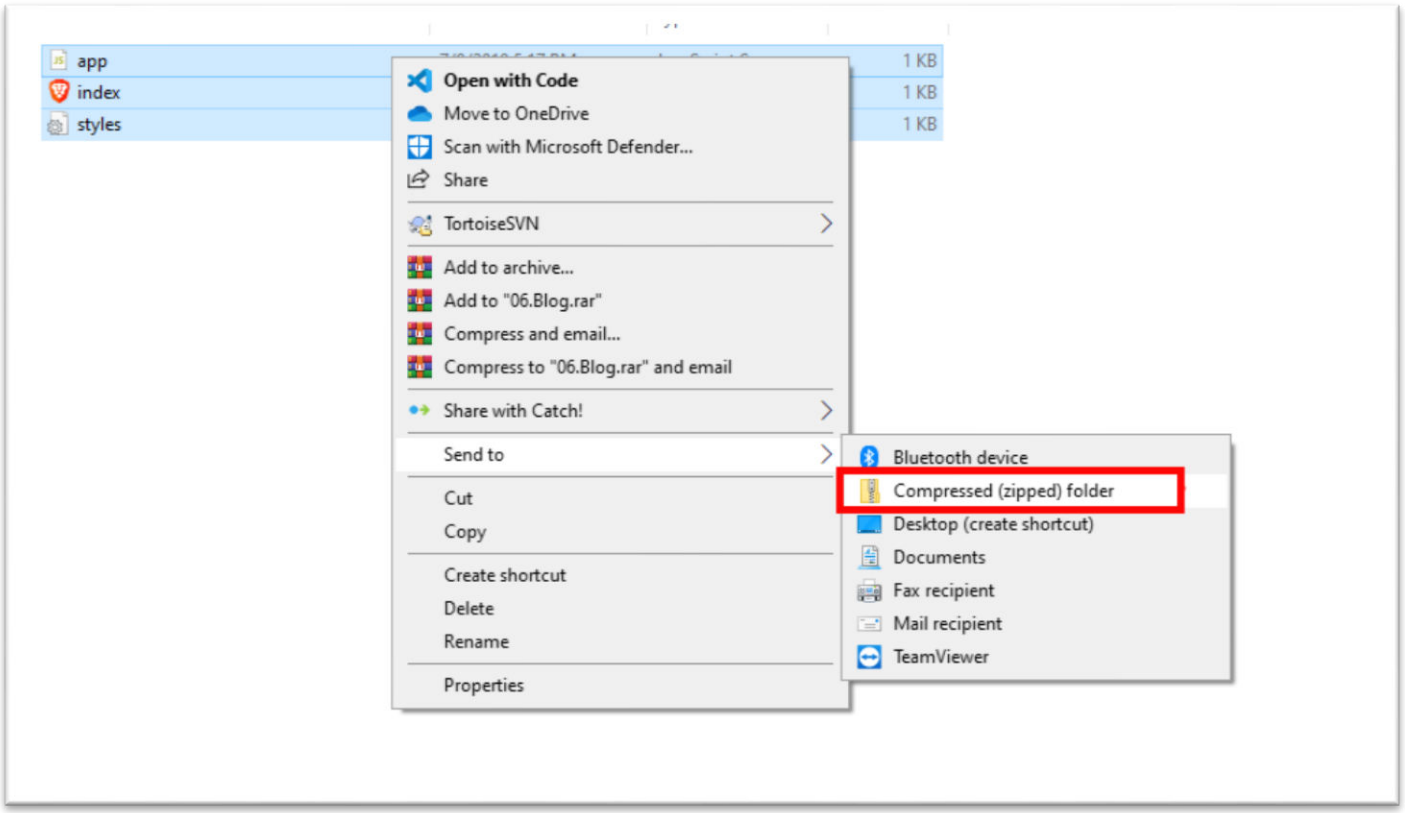


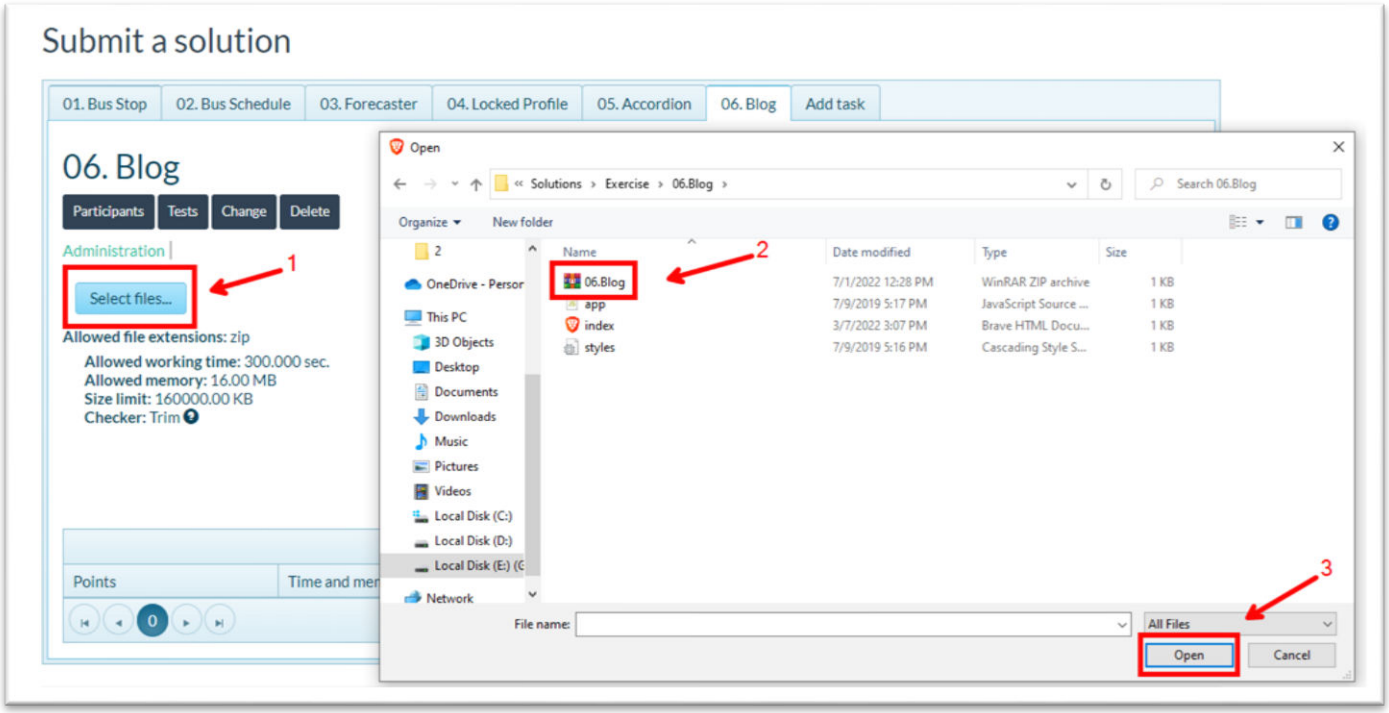
Finish a Course

Clicking the **[Finish Course]** button should send a **DELETE** request to the server and remove the item from your local database. After you've removed it successfully, **fetch** the items **again**.

Submitting Your Solution

Select the content of your working folder (the given resources). Exclude the *node_modules* & *tests* folders. Archive the rest into a **ZIP** file and upload the archive to Judge.


app
index
styles
1 KB
1 KB
1 KB
Open with Code
Move to OneDrive
Scan with Microsoft Defender...
Share
TortoiseSVN
Add to archive...
Add to "06.Blog.rar"
Compress and email...
Compress to "06.Blog.rar" and email
Share with Catch!
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Bluetooth device
Compressed (zipped) folder
Desktop (create shortcut)
Documents
Fax recipient
Mail recipient
TeamViewer


Submit a solution
01. Bus Stop
02. Bus Schedule
03. Forecaster
04. Locked Profile
05. Accordion
06. Blog
Add task
06. Blog
Participants
Tests
Change
Delete
Administration
Select files...
1
Allowed file extensions: zip
Allowed working time: 300.000 sec.
Allowed memory: 16.00 MB
Size limit: 160000.00 KB
Checker: Trim
Points
Time and mer
Open
Solutions › Exercise › 06.Blog
Search 06.Blog
Organize
New folder
2
OneDrive - Persor
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Network
Name
Date modified
Type
Size
06.Blog
app
index
styles
2
7/1/2022 12:28 PM
7/9/2019 5:17 PM
3/7/2022 3:07 PM
7/9/2019 5:16 PM
WinRAR ZIP archive
JavaScript Source ...
Brave HTML Docu...
Cascading Style S...
1 KB
File name:
All Files
Open
Cancel
3