Homework #4

Points: 15

Deadlines:

Groups 1, 2, 3, 4 and **5**: **15 December**, 2024, 23:59

Groups 6 and **7**: **17 December**, 2024, 23:59

The main aim of this homework is to assess your capabilities concerning the use of **Node.js, PostgreSQL database, JWT, and Vue.js** to create the **front-end** and **back-end** of a **simple "secure" App.**

Your App will be similar to what we did in **week 12 and week 13**, with **several new tasks**. In short, the **App** will offer a **front-end** that allows **authenticated users** to see the homepage, which will fetch and present all already added posts from a **table in the database**. A logged-in user can **add new posts, and update, and delete existing ones**. The **front-end** will depend on the **back-end (Node.js App)** which, in turn, will depend on a **database**. The **back-end and database** are expected to handle the various requests coming from the **front-end** and allow the App to work properly.

Detailed information about the App:

The Front-end (Vue.js) App should offer the following "pages"

a. The **home page** (details will follow about its content) should be protected, i.e., only authenticated users can reach/access it.
b. A **contact us "page"** (**not protected**) that contains just basic contact us information. **Note:** you can simply change the "About page" (created by Vue.js) to the "Contacts page".
c. **A signup** "page" that allows a user to register by providing her email and password. The **signup** "page" should look close enough to Figure 1a (**2 points**).
d. **A login** "page" that allows a registered user to login by providing her email and password. The **login** "page" should contain a button that, when pressed, should redirect the user to the **signup** "page". The **login** "page" should look close enough to Figure 1b (**2 points**).

Note: The points for (**c** and **d**) are assigned based on the provided functionalities. For example, you get two points for "**c**" if your signup "page" sends the credentials (email and password) of the user to the server, and the server checks if such a user exists. If not, it will insert the credentials in the database. Then, create a JWT and return it to the client.

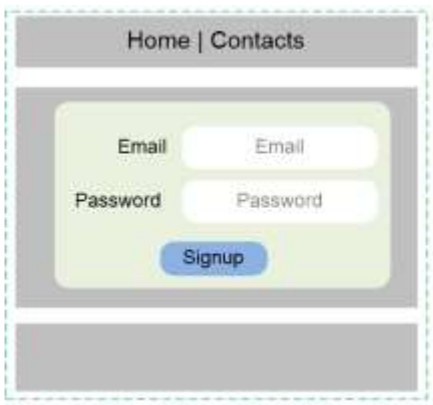

Figure 1a

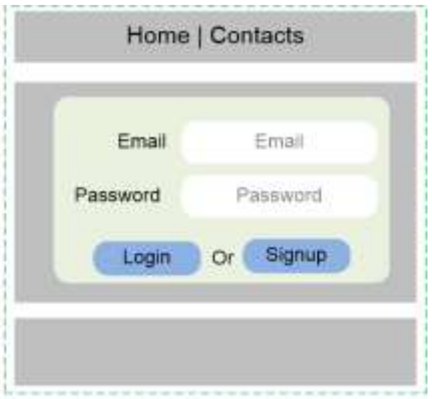

Figure 1b

The **home "page"** should:

a. Automatically **fetches and presents** all posts from the **database** (**1 point**);
b. Each listed post should be **clickable** and when clicked, it should redirect you to "**a post**" page (details about the content of this page will follow) - (**0.5 points**).
c. includes a "**logout**" button that, when clicked, will **logout the user and redirect her to the login page** (**1 point**).
d. includes an "**add post**" button that, when clicked, **will redirect the user to the add post "page"** (details about the content of this page will follow) (**0.5 points**).
e. includes a "**delete all**" button, that when clicked will delete all the posts from the **database** (**2 points**).

The **home page** should look close enough to Figure 2

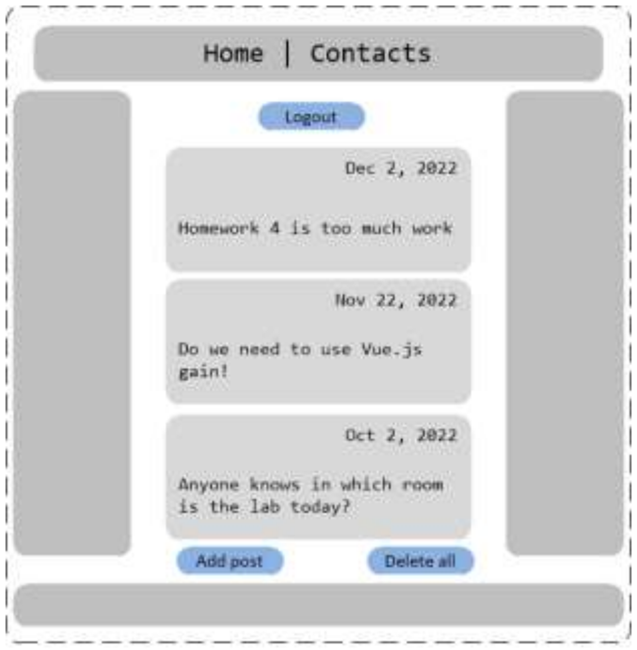
Home | Contacts
Logout
Dec 2, 2022
Homework 4 is too much work
Nov 22, 2022
Do we need to use Vue.js gain!
Oct 2, 2022
Anyone knows in which room is the lab today?
Add post
Delete all

Figure 2

The **add post "page"** should allow only a logged-in user to add a post, and it should look close enough to Figure 3 (**2 points**).

Note: you are adding only the body of the post through the **add post "page"**, but a post has a date too **(check Figure 2)**.

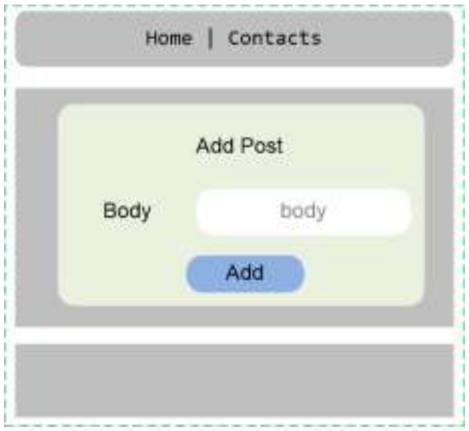

Figure 3

The **a post "page"** should fetch and present a specific post (the one clicked on in the homepage) from the database (**1 point**), and it should contain two buttons:

a. update, when pressed on, will update the post in the database (**1 point**); and
b. deletes, when pressed on, will delete the post from the database (**1 point**).

The **a post "page"** should look close enough to Figure 4.

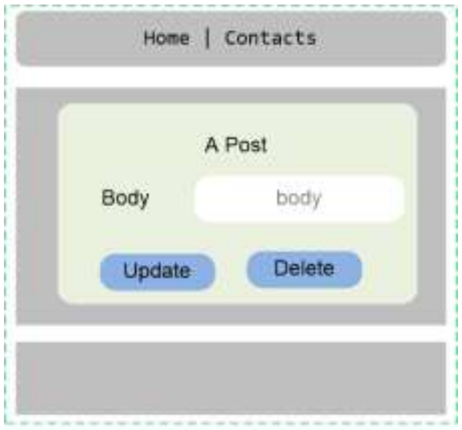

Figure 4

Note: if a user is not logged in, she should not be allowed to reach/access "**a post**" or "**add post" "pages"** (**1 point**).

Important note:

Back-end
Your back-end will be working as an end-point (we will be dealing only with JSON data format, no static files (e.g., images)). Your back-end should be able to handle all **required** CRUD operation requests coming from the front-end.

Security:
You can only use **JWT-related techniques** for the authentication in your App.

Database:
Any table required for your App should be created automatically when you run your back-end App just like we did in week 12 and week 13.

Rules for homework submission and discussion

1. Through Moodle, submit a text file (*.txt) that contains your **Team code**, Name(s), and a **valid and accessible link to the repository** that contains your **homework**. You can make your repo **private** but you need to add your teacher and me as collaborators. Still, you need to submit the link to it through Moodle.
   Note: if the **link** to your **repository** is **not accessible** or **valid** for any reason, you might not be allowed to discuss your homework or at least you will lose **5 points**.
2. You are **not allowed to modify** the content of your repo **after the deadline.**
3. You are **not allowed** to share the link to your repository with anyone **except your lab teacher and the course lecturer**.
4. You **must send** the link to your repository to your teacher via a **direct message** in **Slack, and include** the **team number**, and name(s) in the message.
5. **All team members should attend the discussion** of their homework; you **will not be allowed to discuss** if your team **is not complete**. If you already know that your team will not be complete because one or more of the members cannot attend due to another commitment, **contact me** as soon as possible and we can find a solution.
6. You have to submit your homework by the defined deadline, and **you will lose 0.5 point for each hour of delay.**

The previous rules will be strictly enforced and there will be no exceptions