CODELABS ACADEMY

File Upload With Expressjs

In this assignment, you will implement profile update with file upload functionality in both your Express backend app and React app, utilizing Supabase as a backend-as-a-service.

1. Supabase setup

- Before you begin, ensure to create a new project in Supabase.
- Create a **public** bucket named `avatars`. The bucket should be public to allow anyone with the image URL to access it, as we are not using the Supabase authentication service in our backend.
- Install the Supabase JavaScript client in your Express application.
- Copy the project URL and the API key, and set them as environment variables in your Express app.
- Install multer library. Multer is a Node.js middleware used to handle multipart/form-data, the data transfer format for file upload.

2. Profile Update

- Create a route in the coder profile management module in your express application that allows the coder to **update** his profile. The coder can update his `avatar`, `first_name`, `last_name,` and `about` fields.
- Create an upload middleware using Multer. Configure it to use in-memory storage, which is suitable for handling file uploads without saving them to disk immediately, and add it to the previous route.

You can make use of this code snippet for the in-memory storage middleware:

JavaScript

```
export const uploadMiddleware = multer({

  storage: multer.memoryStorage(),

});
```

- Create the controller for that route.
- Develop a utility function responsible for uploading the file object (accessible via request.file) to the Supabase `avatars` bucket. Ensure that this function retrieves the public URL of the uploaded image upon a successful upload.
- Create the service called by the controller and pass the request data to it along with the file object.
- The service should invoke the upload function and get the public URL.
- The service should update the coder's profile information based on the received data. If the public URL is not empty, update the `avatar` field. Also, update the `first_name`, `last_name`, and `about` fields if they are present in the request data.

Note: If you get row-level security issues, make sure to have the right policies applied to your bucket or **disable this security feature just for the sake of development**! using the following SQL query.

```
Unset
ALTER TABLE storage.objects DISABLE ROW LEVEL SECURITY
```

3. Frontend Integration

Make sure that when the submit button of the profile page is clicked to send a `multipart/form-data` request to this endpoint.