File Upload

Introduction to Uploading files

One of the very important parts of creating a web application or website, in general, is files. You have your files that run the code and show something on the browser. Also, when the user is interacting with the website, he/she would also want to upload some files. For example - Uploading your profile picture or posting an image on Facebook or any other social networking sites.

In this module, we will be creating a page to upload our profile picture on the user's page using a library.

***EXTRA*:**
You can check out various websites which allow the user to upload images in a post, in the profile picture, in comments, etc, and make a list of it and try to find some difference if they have any.

Uploading Files: How Does It Work?

- We have our server that has been initialized on a file system or an operating system.
- Let's say on our profile page, we choose a picture and we send it to the server in the form of a file { **File is a sequence or a stream of bytes** }.
- The server is connected to the database which is not saving the file. When the file is sent to the server we store our file inside the folder { **avatar** } and reference it inside the database.
- The hash of every user in our database { **MongoDB** }, we have a key called an **avatar** and that key has a value which is the path to the file.

- The separate folder is one of the local storage options that we use for smaller-scale projects.
- In the real world, we store the file somewhere else in a bucket { **A folder on the cloud is known as a bucket** }. We store the file in the bucket and give the database the path of the file.
- We can store more details in the database other than the path that is the file type, name of the file, size of the file, etc.

Installing Multer + Documentation

- Usually, when we are submitting the form we send some text data, some number data, etc but to send a file we have one input type called a **{ file }** which is the attribute value.
- We use another attribute called encryption type, and we say that this form has multipart data { It can have the text and file data }.
- In the terminal install multer using the command { **npm install multer** }.
- Whenever we define the variable for multer, the setting, or the configuration of multer for uploading the avatar of the user, we will also be defining the destination of the file it will be stored at and the file name.
- In the file { **user_profile.ejs** }, we will put multipart inside the form using the attribute enctype. We will define another input field with a type **file** where we will be uploading the profile picture.

- For setting up multer we will go to the models folder and inside the file { **user.js** } we will be doing the following things.
    1. Import the multer library and the path of where the file will be stored.
    2. We will define the path that is { **/uploads/users/avatars** }.
    3. We will create a new folder **{ uploads }**, inside the uploads folder we will create one more folder **{ users },** inside the users folder we will create one more folder **{ avatars }.**

EXTRA:
You can check out the link below to understand more about multer -

https://www.npmjs.com/package/multer

Configuring Multer for User Avatar

- We have defined the folder in a variable, but not in the schema of the database.
- We will define the field inside the schema with the datatype string.
- We have to link the avatar path, multer, and the avatar field making sure that whenever we are saving the file it gets saved inside the folder using multer and the path gets saved inside the field.
- We will be using disk storage as we are storing the files and folders inside the same machine.
- **multer.diskStorage** is a function that will take an object which has further two keys, one of them defines the destination and the other defines the filename.

- The reason for the file name key is because of the following reason - If you and I are going to be saving a file with the same name, one of the files will be overriding the other one so to prevent that we are going to append the current time in { milliseconds } since 1st Jan 1970 midnight which is called as **EPOC time** using **Date. now()** function.

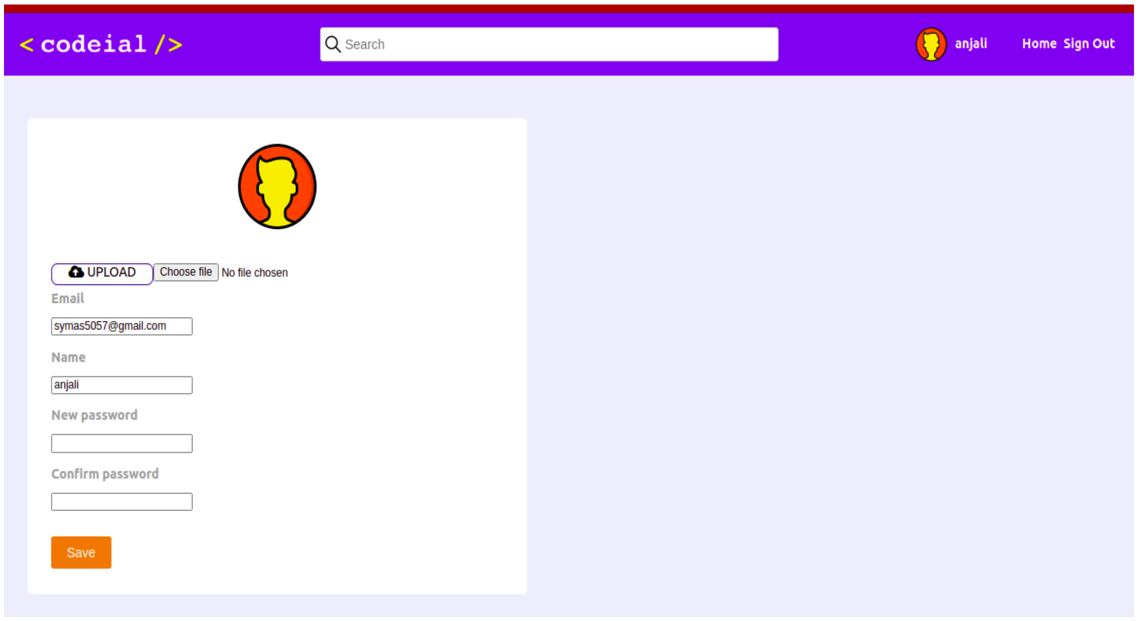
Saving File from Params

- In the filename key of the **multer.diskStorage** function, there exists the field name in which we are storing the file.
- Every file that we upload for the field for every user will be stored as avatar- { the value of **Date. now()** function } .
- We need to define static functions - { Static functions are the ones which can be called overall on the whole class }.
- .single is the property that says that only one file can be uploaded for the field name **avatar.**
- We have to define the avatar path because we need this avatar path { **/uploads/users/avatars** } to be available publicly for the user model.
- We have to create a check while uploading the file to check whether the user is updating the file or uploading the file.

Showing the Avatar, Connecting the Dots

We have to show the image that we have uploaded on the profile page.

- We have to declare an img tag and the src for this image will be **users.avatar** as this is the path for our image**.**
- We need to make this path available to the browser when the browser asks for it.
- Inside the { **index.js }** file, make the uploads path available to the browser using **app.use** method



Email

symas5057@gmail.com

Name

anjali

New password

Confirm password

Edge Case:: Replacing an Avatar

Whenever we are uploading an avatar while one was already uploaded earlier, the previous one either gets replaced, deleted or archived somewhere else.

- Inside the file **{ users_controllers.js }** { if there is any file in the request then we have to upload it } we have to check if the user already has an avatar associated with him /her.
- If it is present, we have to remove that avatar and upload a new one.
- For deleting the previous avatar, we need the FS { file system } and the path module.

Summarizing File Uploads

- You can limit the size of the file that the user can upload, you can limit the type of file that the user can upload using multer.
- Input type file keeps the first level of security, it doesn't give you the path.

***EXTRA*:**
Your Assignment is to -

- To display the preview of the image selected and remove that preview if we don't like that image.
- Beautify the page.