Photo Viewer

Project Link: [github.com/voltamperewatt/Photo-Viewer](https://github.com/voltamperewatt/Photo-Viewer)

This application is a full-stack application that I built from scratch using the MERN stack. I integrated an AWS S3 bucket to store the images.

# Screenshots

Login screen, Gallery, Upload Image, Gallery rendered again after upload, grid and list view, resizing on a larger screen,

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a person

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a computer

Description automatically generated

1AWS S3 Bucket

A screenshot of a computer

Description automatically generated

2MongoDB database

# What is the MERN stack?

The MERN stack is a web development stack that used JavaScript on the frontend and backend. The elements of the stack are

the **MongoDB** database, the **Express.js** web application framework for the backend, the **React.js** web framework for the frontend, and the **Node.js** runtime.

# How does this application work?

The application has login functionality. When you log in with your credentials, the gallery is loaded. The gallery only loads the images that are linked to a specific username.

A record of the usernames, passwords, and image URLs are stored in MongoDb collections.

# How is the database structured?

Table 1: Users collection

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| username | String | Usernames of the users registered |
| password | String | passwords of the users stored in plain text |

Table 2: Images Collection

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| url | String | Download URL of file on AWS S3 bucket |
| uploader | String | The user who uploaded the file. Value extracted from username field in document of Users collection |

# Application Architecture

A close up of a logo

Description automatically generated

# Things to work on

* Encrypt passwords before storing
* Set up AWS configuration and tighten up access

# Notes

* Explanation of Async - <https://www.stanleyulili.com/javascript/asynchronous-programming-with-callbacks-in-javascript/>
* Async/await - <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await>
* Promises - <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise>
* Callback formats for Mongoose - <https://mongoosejs.com/docs/queries.html>
* To run backend server, use nodemon server. To run frontend server, user npm start