

DATABASE MANAGEMENT SYSTEMS LAB

Assignment - 5

Multimedia Management System

Team Name:

**Raghukul Bhushan Siyawar
Ramchandra ki Jai**

Team members

Nikhil Saraswat (20CS10039)
Vivek Jaiswal (20CS10077)
Shah Dhruv Rajendrabhai (20CS10088)
Amit Kumar (20CS30003)
Jay Kumar Thakur (20CS30024)

INTRODUCTION

In today's digital era, multimedia content has become ubiquitous, and it is imperative to have an efficient multimedia database to store and manage multimedia files. The proposed project aims to develop a cutting-edge multimedia database that can store and retrieve text/structured data/music/images/video files in an efficient and secure manner. Additionally, cloudinary API integration will ensure secure and reliable storage of multimedia files.

The primary goal of the project is to create a highly efficient multimedia database that enables the swift and secure storage and retrieval of multimedia files. Users can store text/structured data/music/images/video files in the database and execute fundamental queries with ease. The multimedia database will be designed to be user-friendly, secure, and highly scalable to cater to the diverse needs of businesses and individuals alike.

The frontend of the multimedia database will be designed with a focus on user experience, leveraging cutting-edge react and express techniques. The backend will use advanced technologies such as nodejs and MySQL to store multimedia files in a cloudinary API, ensuring secure and dependable storage.

Multimedia-Management-System-Frontend:

This is the frontend module of the Multimedia Management System Create a Multimedia Management System for storing and managing multimedia files

Technologies Used:

Multimedia Management System uses a number of open source projects to work properly:

- node.js - evented I/O for the frontend
- React - fast node.js webapp frontend framework
- Redux - state container for caching
- Redux-Thunk - thunk middleware for dispatching and get the state
- React-Router-DOM - some components and hooks like useParams, Link, History etc.
- Axios - making API Calls
- AntD - frontend framework for components like Table, Icons etc.

Multimedia-Management-System-Backend:

This is the backend module of the Multimedia Management System Create a Multimedia Management System for storing and managing multimedia files

Technologies Used:

Multimedia Management System uses a number of open source projects to work properly:

- node.js - evented I/O for the backend
- Express - fast node.js network app framework
- Cloudinary - cloud services to store images and videos

DATABASE SCHEMAS

a) User Schema:

```
CREATE TABLE Users (  
  id INTEGER PRIMARY KEY AUTO_INCREMENT,  
  name TEXT NOT NULL,  
  email VARCHAR(255) NOT NULL UNIQUE,  
  password TEXT NOT NULL  
);
```

- All the four users have common user table which are distinguished by **role** attribute.
- Here **id** attribute is the primary key. We can also make email as primary key but as we are storing json-web-token in the front-end storage so anyone can access it so we are storing only **id** in json-web-token to prevent access of personal details of user.
- Each user has a name which is non-nullable text field
- There is an email field which is a varchar(255) which is a non-nullable unique field.
- Password is a non-nullable text field. We have used text field because we are using encryption in this field that will ensure the security. Even if the database gets hacked, the password will be safe.

b) Multimedia Schema:

```
CREATE TABLE Multimedia (  
  id INTEGER PRIMARY KEY AUTO_INCREMENT,  
  public_id VARCHAR(255) NOT NULL UNIQUE,  
  title TEXT NOT NULL,  
  author TEXT,  
  description TEXT NOT NULL,  
  type ENUM('image', 'video', 'audio', 'document') NOT NULL,  
  link TEXT NOT NULL,  
  public INTEGER NOT NULL DEFAULT 0,  
  user_id INTEGER NOT NULL,  
  FOREIGN KEY (user_id) REFERENCES Users(id)  
);
```

The Multimedia table is a relational database table that stores information about multimedia content uploaded by users. The table contains the following columns:

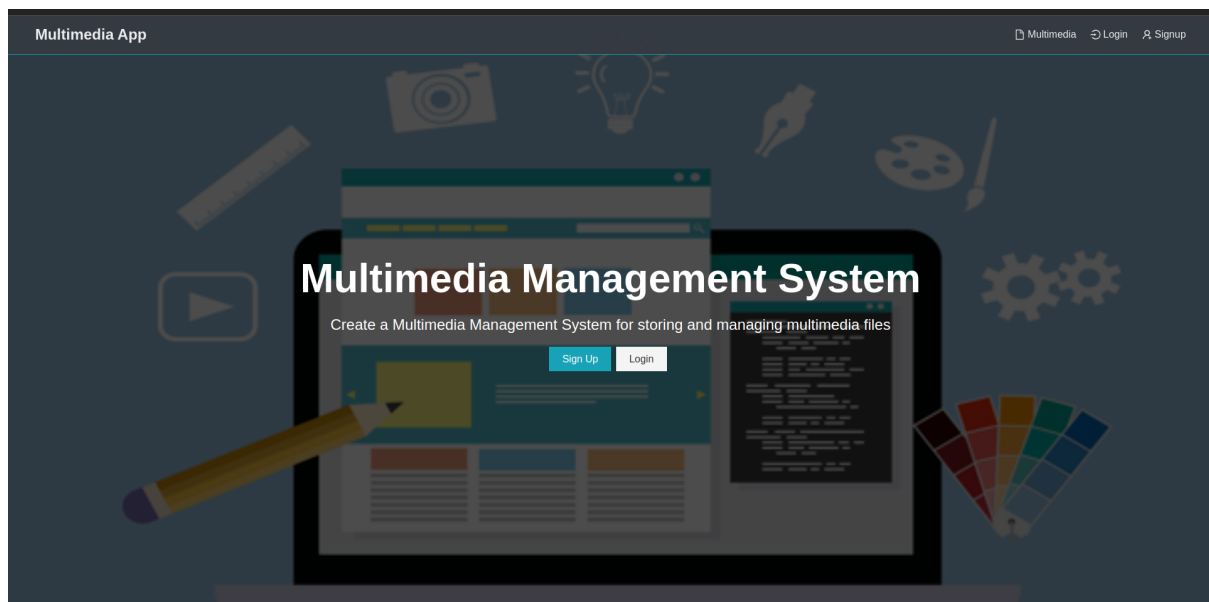
- id: A unique identifier for each multimedia content, with auto-incrementing integer values.
- public_id: A unique identifier that is publicly displayed for each multimedia content, stored as a string with a maximum length of 255 characters.
- title: The title of the multimedia content, stored as a text data type.
- author: The author of the multimedia content, stored as a text data type.
- description: A description of the multimedia content, stored as a text data type.
- type: The type of the multimedia content, which can be one of four values: 'image', 'video', 'audio', or 'document'.
- link: The URL link to the multimedia content, stored as a text data type.
- public: A boolean value indicating whether the multimedia content is public or not. If set to 1, the content is public; if set to 0, the content is private.

- **user_id:** A foreign key referencing the id column in the Users table, indicating the user who uploaded the multimedia content.

Overall, the Multimedia table provides a way to organize and store various types of multimedia content uploaded by users, with the ability to track ownership and privacy settings.

WEBPAGE USER INTERFACE:

HOME PAGE:




LOGIN:


CASE 1: Invalid Email Format:

Multimedia App

MultimediaLoginSignUp

Please include a valid email

 **Sign In**

 Sign In Your Account

Login


Don't have an account? [Sign Up](#)


CASE 2: Invalid Email or Password:

Multimedia App

MultimediaLoginSignUp

Invalid Credentials

 **Sign In**

 Sign In Your Account

Login

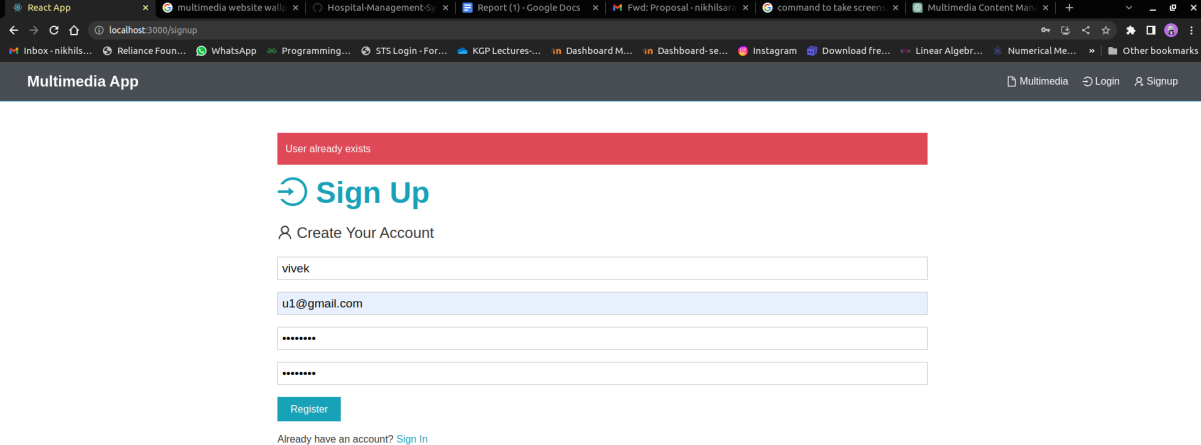
Don't have an account? [Sign Up](#)

SIGN UP :

Click on the signup button on navbar, it will redirect to '/signup':

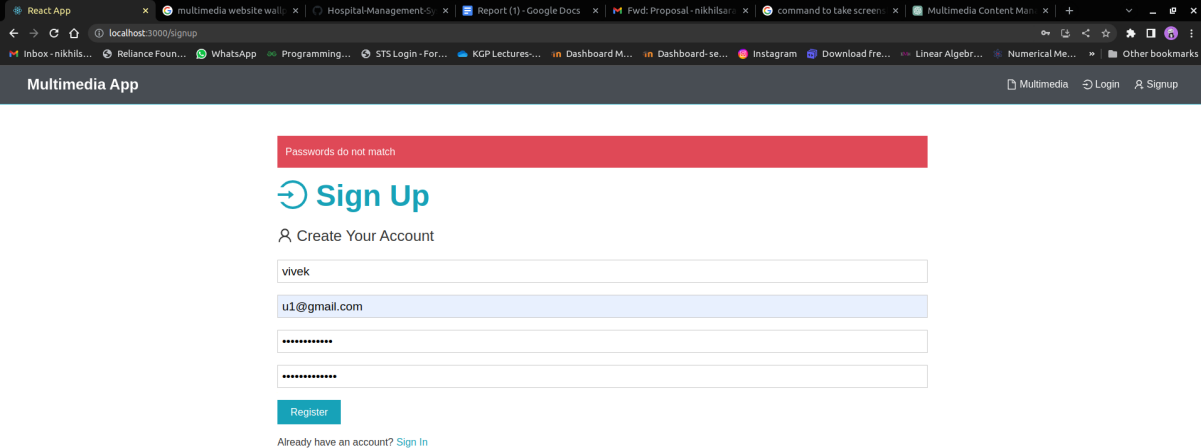
FILL UP CREDENTIALS:

CASE 1: Email Already Registered:



The screenshot shows a web browser window with the URL `localhost:3000/signup`. The page title is "Multimedia App". A red error message at the top states "User already exists". Below this is the "Sign Up" section with the heading "Create Your Account". The form contains four input fields: a text field with "vivek", an email field with "u1@gmail.com", and two password fields, both containing masked characters "*****". A blue "Register" button is at the bottom of the form. Below the button, a link says "Already have an account? Sign In".

CASE 2: Password doesn't match with confirm password:



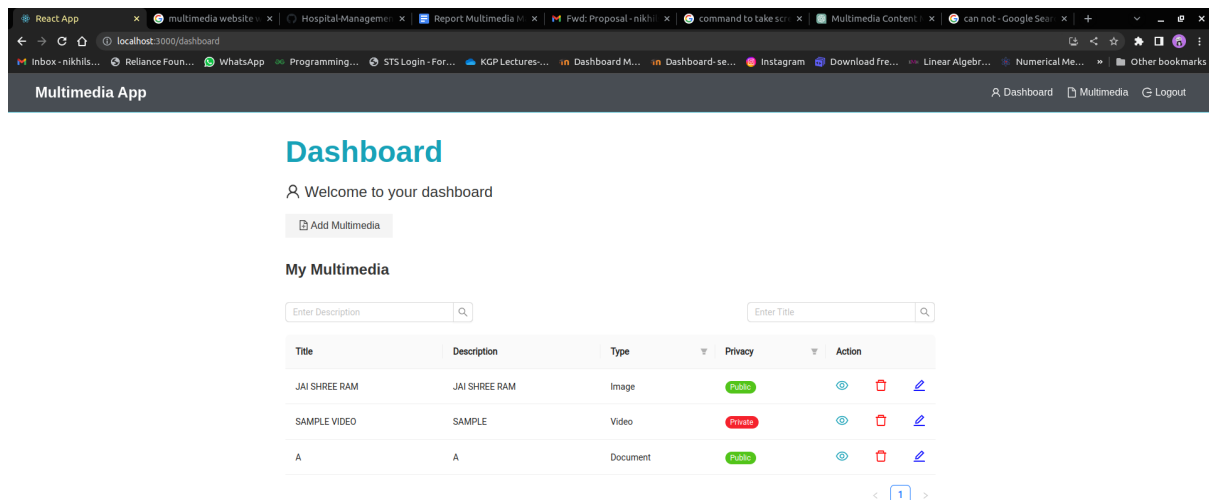
The screenshot shows the same "Sign Up" page as before, but with a red error message at the top stating "Passwords do not match". The form fields and the "Register" button are still visible. The "Sign In" link remains at the bottom.

NOTE: USER CAN'T ACCESS ANY UNAUTHORISED PAGE

Example, without signin user can't access dashboard page

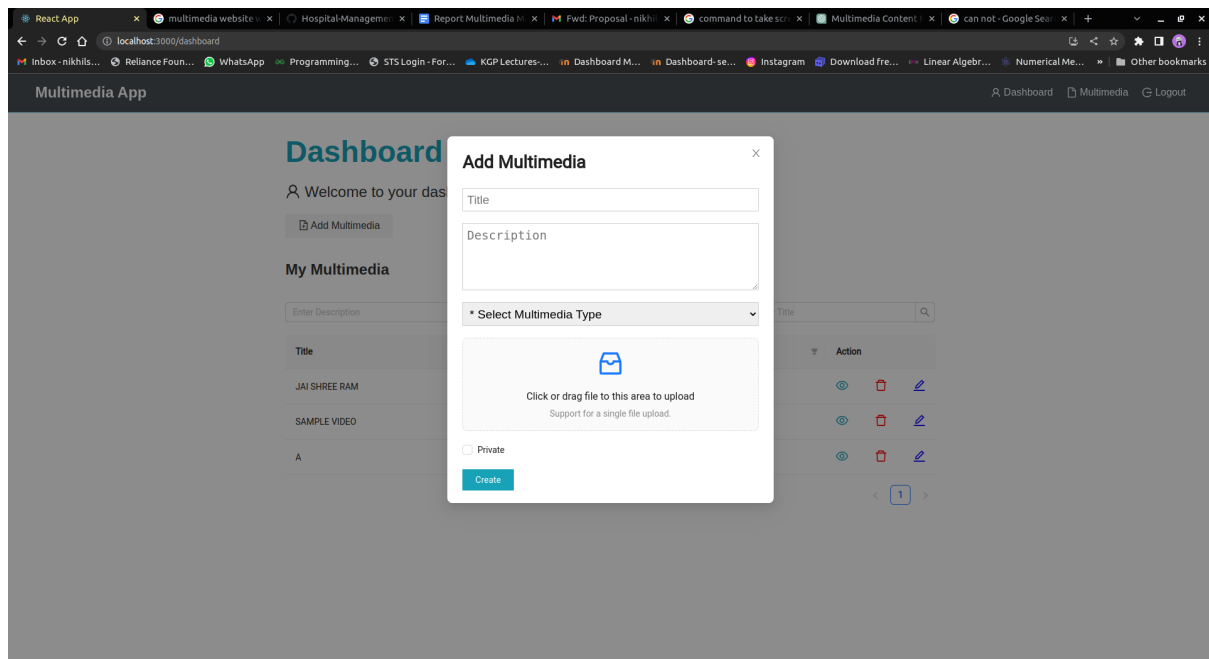
AFTER SIGN-IN:

DASHBOARD:



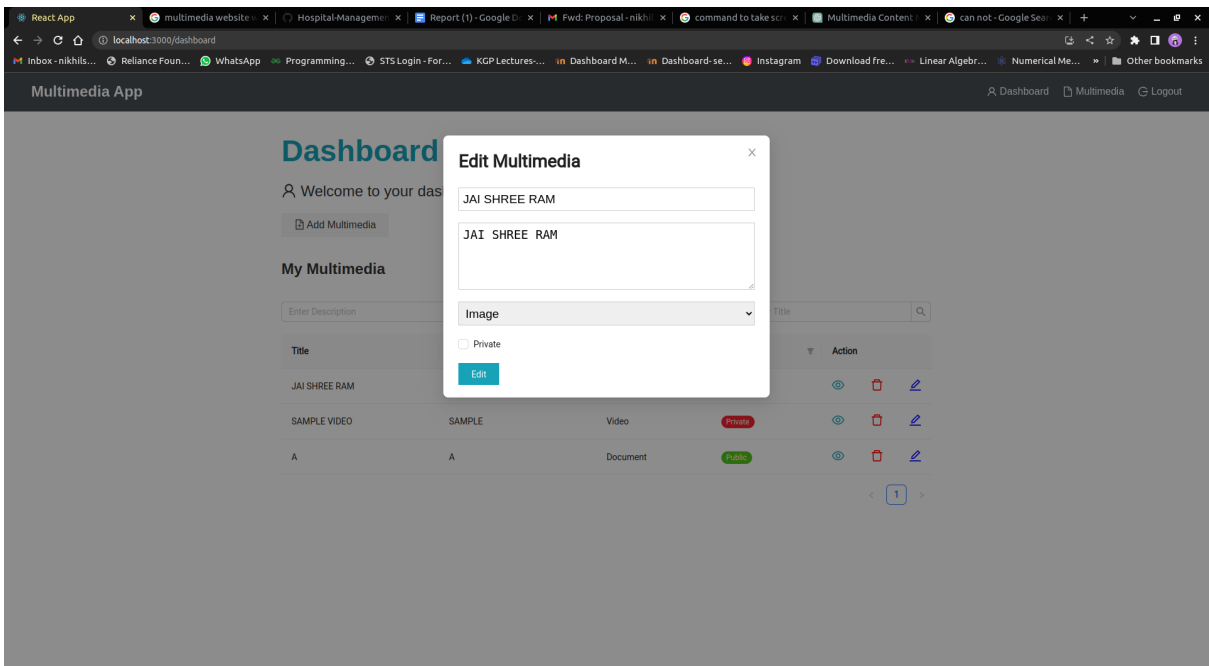
CREATE MULTIMEDIA FORM:

All the fields are compulsory and there is a checkbox, if user want to make his/her multimedia files private then only user can see his/her own multimedia files.

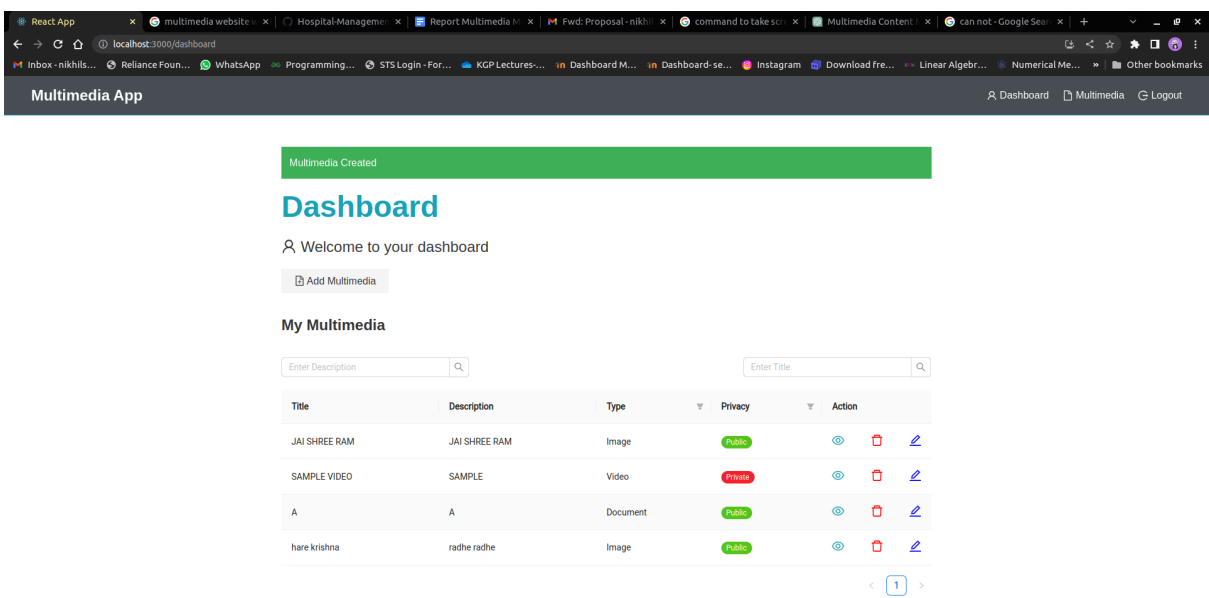


EDIT MULTIMEDIA:

User can edit multimedia files (change privacy, edit title, description etc.)

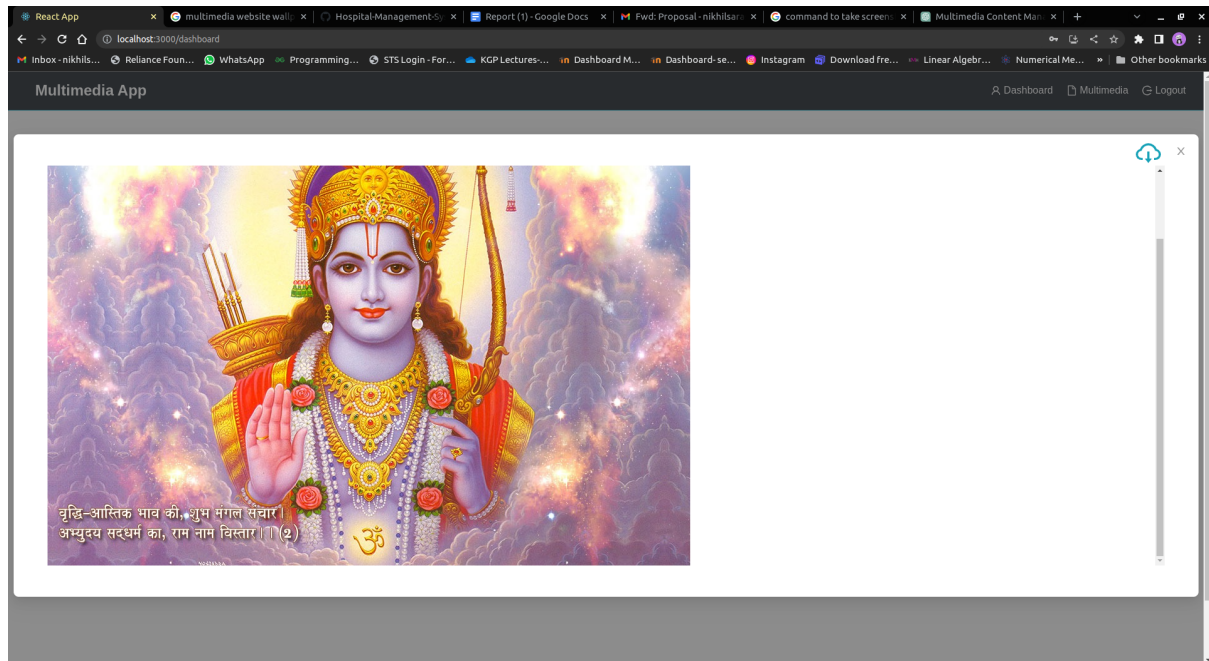


SUCCESSFUL UPLOAD OF MULTIMEDIA FILE:

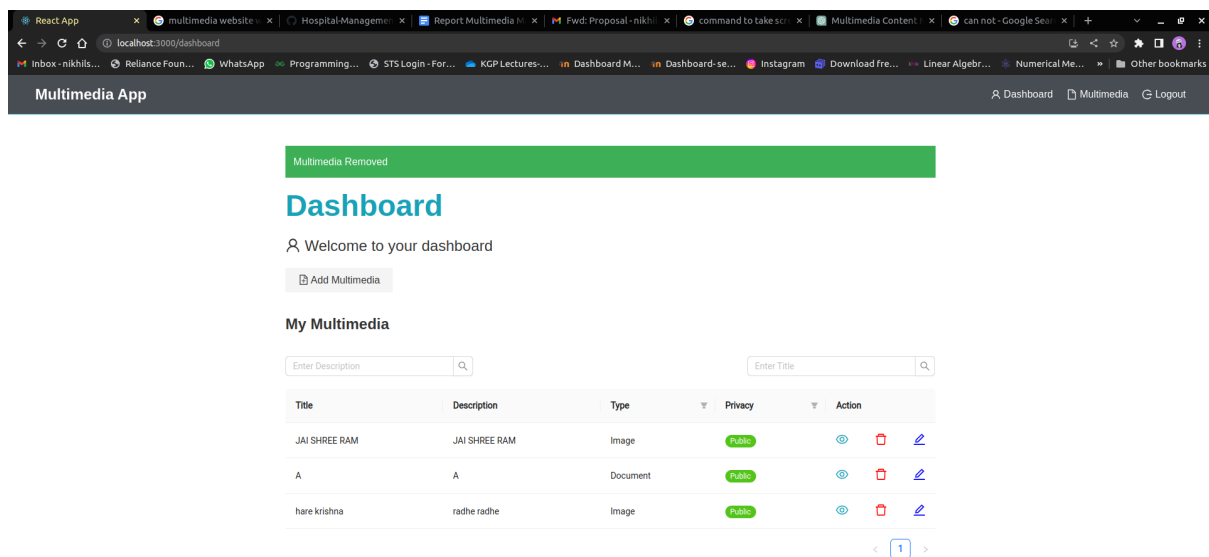


PREVIEW OF MULTIMEDIA FILE:

User can preview a file (video/images) by clicking on preview button and there is a button for download the file.

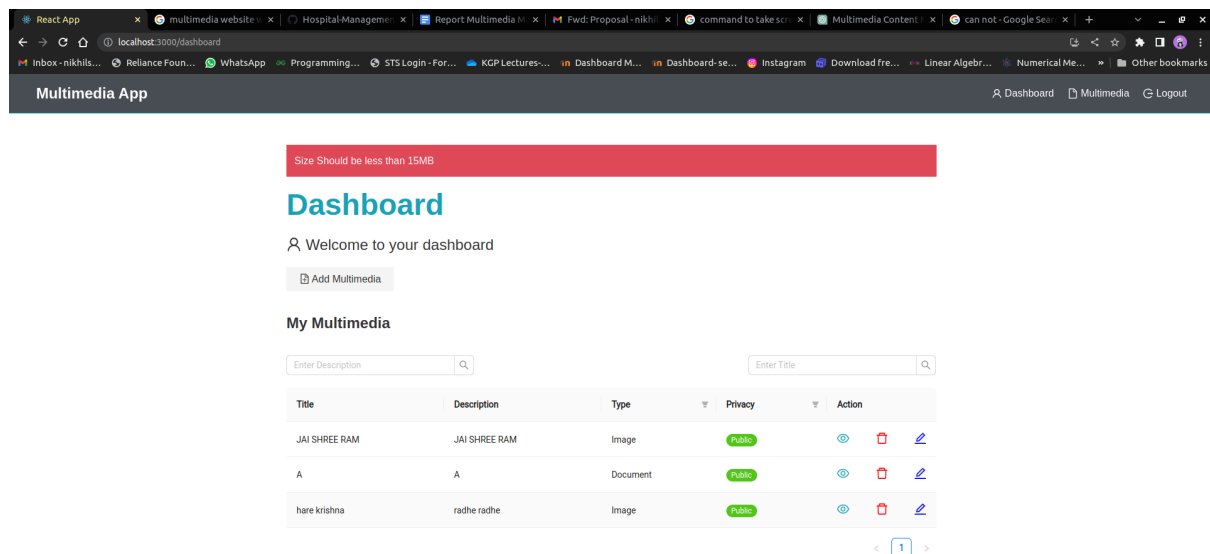


DELETION OF MULTIMEDIA FILE:



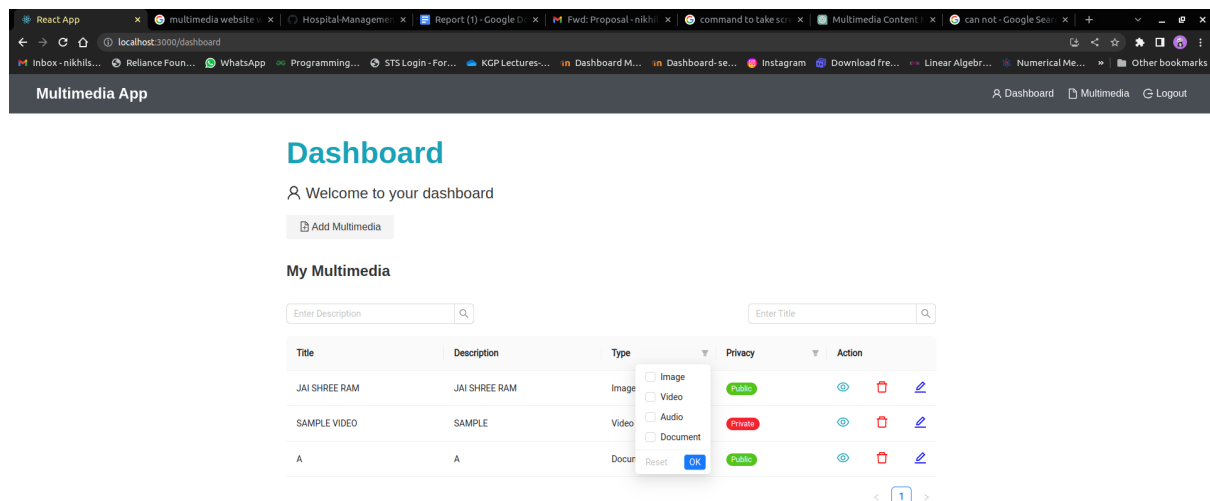
LIMIT ON MAX-SIZE OF FILE:

We have a max limit on the file size to be uploaded, so that no mysterious file which is of large size can not be uploaded, which will keep our file-system safe.

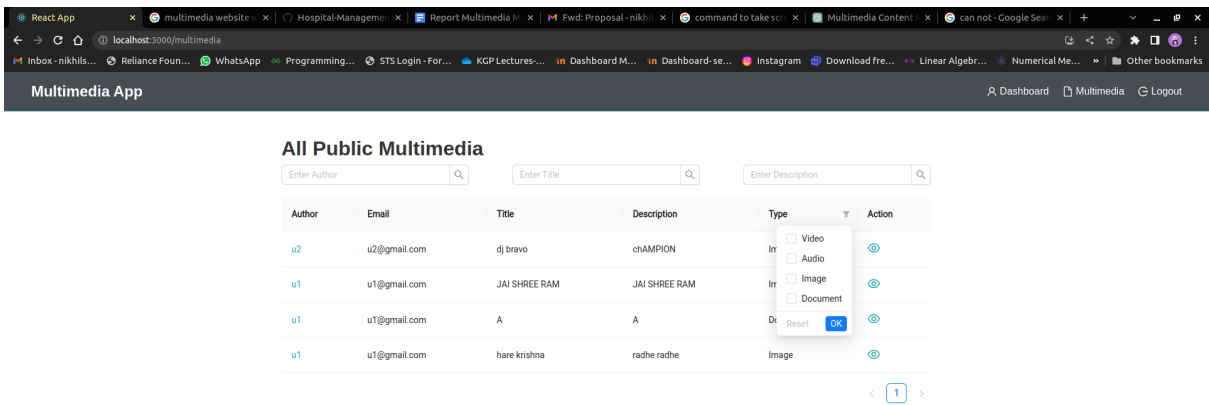


FILTER :

User can filter all files in his/her dashboard based on either type or privacy in his/her own dashboard which will show only his/her files.

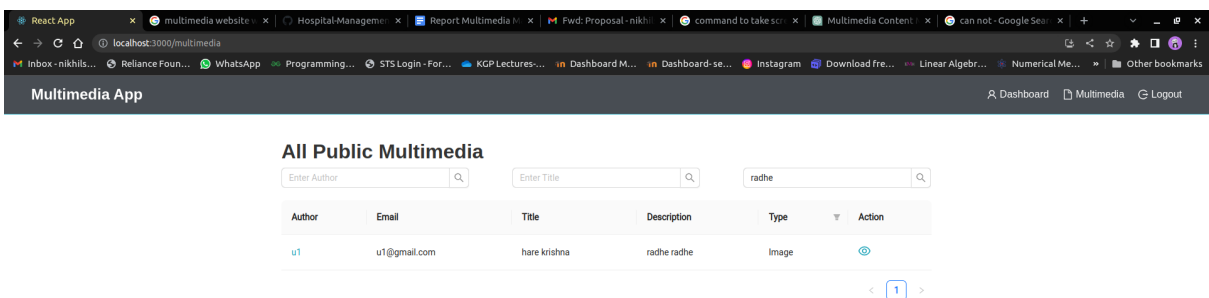


User can also filter all public files of all users in Multimedia section based on type.



SEARCHING:

User can search based on title, author and description.



CONCLUSION:

All the features mentioned in proposal of multimedia database have been implemented successfully. This is an advanced solution for efficient storage and retrieval of text/structured data/music/images/video files. The project will make use of advanced technologies such as nodejs, mysql, react and express for frontend and backend development. The multimedia database will feature a user-friendly interface, ensuring that users can upload, search for, and retrieve multimedia files promptly and effortlessly. The project's cloudinary API integration will guarantee secure and dependable storage of multimedia files. Overall, the proposed multimedia database is an impressive solution for businesses and individuals alike, catering to their diverse multimedia storage and retrieval needs.

All the source files for the project are as following:

Frontend Module: [GitHub Link](#)

Backend Module: [GitHub Link](#)

USER GUIDE:

FRONTEND:

Installation:

Multimedia Management System requires Node.js v10+ to run.

Install the dependencies and devDependencies and start the server.

```
cd Multimedia-Management-System-Frontend  
  
npm i  
  
npm start
```

Note: Before Running Please Run Backend Parallely.

BACKEND:

Create the Database:

Create your MySQL Database and change ./utils/config.js file as per your configuration and run database.sql in your Database for creating all the required table.

Installation:

Multimedia Management System requires Node.js v10+ to run.

Install the dependencies and devDependencies and start the server.

```
cd Multimedia-Management-System-Backend  
  
npm i  
  
node app
```

Note: Before Running Please Include Your Own **Cloudinary Environment Variables**.
Sample .env file should look like this:

```
CLOUDINARY_CLOUD_NAME=<your cloud name>  
  
CLOUDINARY_KEY=<your cloud key>  
  
CLOUDINARY_SECRET=<your cloud secret>
```