

## TITLE OF THE PROJECT

“ Video Sharing App (Youtube Clone) Mern ”



TEAM ID: LTVIP2024TMID08297

TEAM DEATAILS:

Register Number.	Name.	Role.	Email
SBAP0008297	Perugu Hinduja	Team Leader	<a href="mailto:peruguhindujareddy@gmail.com">peruguhindujareddy@gmail.com</a>
SBAP0008281	G.Bhanu Prakash	Team Member	<a href="mailto:gudikatibhanuprakash@gmail.com">gudikatibhanuprakash@gmail.com</a>
SBAP0008291	M.iswarya	Team Member	<a href="mailto:aishwaryaaishu71779@gmail.com">aishwaryaaishu71779@gmail.com</a>
SBAP0008293	M Ismail Zabiulla	Team. Member	<a href="mailto:mominzabiulla786@gmail.com">mominzabiulla786@gmail.com</a>
SBAP0008303	Sg Nayana	Team Member	<a href="mailto:nayana200213@gmail.com">nayana200213@gmail.com</a>

## Contents

1. INTRODUCTION
2. TECHNICAL ARCHITECTURE
3. ER DIAGRAM
4. PREREQUISITES
5. PROJECT STRUCTURE
6. APPLICATION FLOW
7. PROJECT FLOW
8. Project Setup And Configuration
9. Database Development.Configure MongoDB
10. Frontend Development
11. Project Implementation
12. Conclusion

## **INTRODUCTION :**

### **Description:**

A video sharing app allows users to upload, discover, watch, and share videos on various topics. Users can create profiles, follow channels, like and comment on videos, and interact with other users through messaging and social features. The app typically includes features for content discovery, personalized recommendations, notifications, and engagement metrics such as views, likes, and shares.

### **Scenario:**

John, a fitness enthusiast, recently started a YouTube channel to share workout routines, healthy recipes, and wellness tips. He uploads a new video every week showcasing different exercises and nutrition advice.

### **Content Creation:**

John spends a couple of days planning, filming, and editing his latest workout video, which focuses on high-intensity interval training (HIIT) for burning calories and building endurance.

### **Uploading to the App:**

After editing, John uploads the video to the video sharing app. He adds relevant tags, a catchy title ("Ultimate HIIT Workout for Beginners"), and a detailed description with timestamps for each exercise.

### **Engagement and Feedback:**

Once the video is live, John actively engages with his audience by responding to comments, answering questions, and encouraging viewers to like and share the video. He also promotes the video on his social media accounts to reach a wider audience.

## Analytics and Performance:

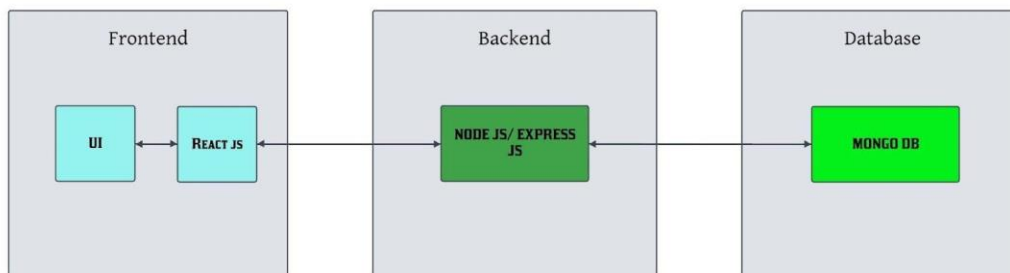
John regularly checks the app's analytics dashboard to track the video's performance. He monitors metrics such as views, likes, watch time, and audience retention to understand what content resonates most with his viewers.

## Community Building:

Over time, John builds a loyal community of fitness enthusiasts who subscribe to his channel, participate in live Q&A sessions, and share his videos with their friends. He collaborates with other fitness influencers and hosts challenges to keep his audience engaged and motivated.

**Monetization Opportunities:** As John's channel grows, he explores monetization options offered by the app, such as ads, sponsorships, and merchandise sales. He also considers launching a premium membership tier with exclusive content and perks for dedicated fans.

## TECHNICAL ARCHITECTURE:



The technical architecture of Video sharing application follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the Axios library to connect with backend easily by using RESTful APIs.

The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.

For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, etc. It ensures reliable and quick access to the necessary information.

Together, the frontend and backend components, along with Firebase Storage, Express.js, and MongoDB, form a comprehensive technical architecture for our Video Sharing app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive blogging experience for all users.

## **ER DIAGRAM:**

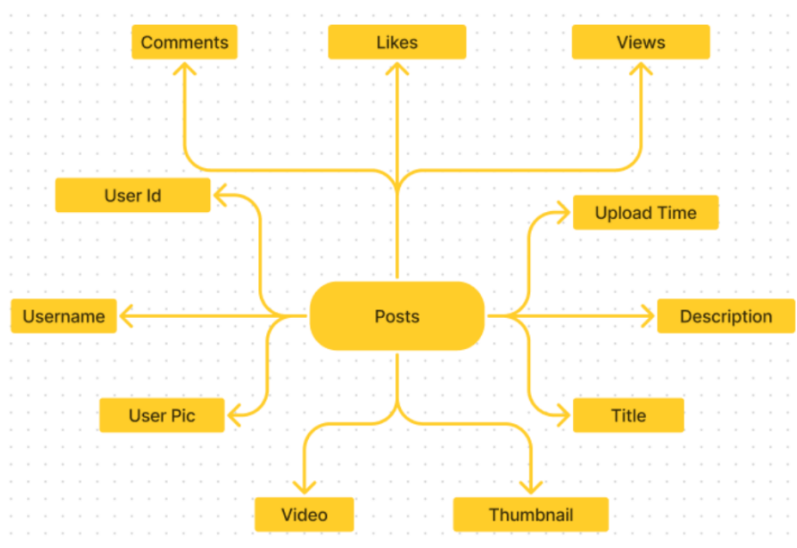
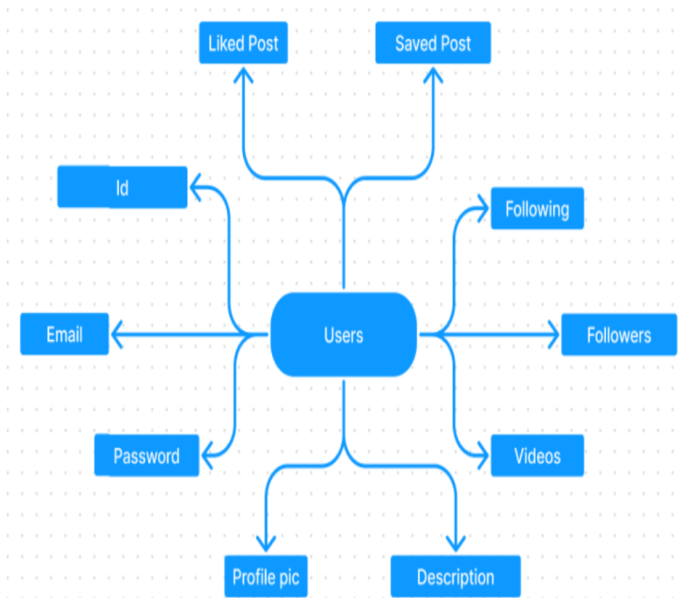
The Video sharing ER-diagram represents the entities and relationships involved in a video. It illustrates how users, posts are interconnected. Here is a breakdown of the entities and their relationships.

### **USER:**

Represents the individuals or entities who view, like, save and upload posts.

### **POSTS:**

The posts can be searched by users, and also users can set title, thumbnail, description while uploading videos. It can also count and display views and likes



## **PREREQUISITES:**

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

### **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

### **Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

### **MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.



**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Firebase: Firebase Cloud Storage is a powerful and scalable cloud-based storage solution offered by Firebase, a Google Cloud Platform service. It provides developers with a secure and efficient way to store and manage user-generated content, such as images, videos, and files, directly in the cloud. With Firebase Cloud Storage, you can easily integrate cloud storage capabilities into your applications, allowing users to upload, download, and share data seamlessly while benefiting from Google's robust infrastructure and security measures.

**HTML, CSS, and JavaScript:**

Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:**

Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Framework:**

Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

**Version Control:**

Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

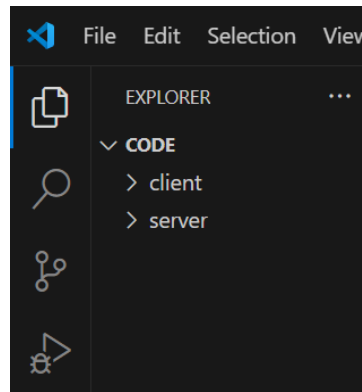
Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>

**Development Environment:**

Visual Studio Code

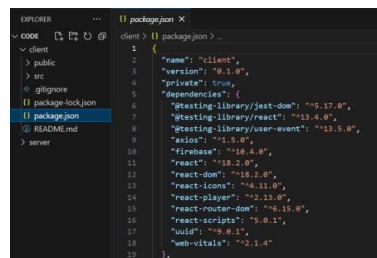
## PROJECT STRUCTURE:

- Inside the app directory, we have the following folders



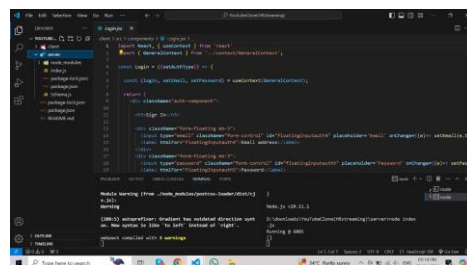
- **Client directory:**

The below directory structure represents the directories and files in the client folder (front end) where, react js is used along with Api's.



- **Server directory:**

The below directory structure represents the directories and files in the server folder (back end) where, node js, express js and mongodb are used along with Api



## **APPLICATION FLOW**

### **User:**

- The user can access all the videos on the platform.
- Users can search for the required content.
- Users can like, share, or save the video and can even like and comment on videos.
- Users can follow other creators and their posts will be recommended.

### **Content creator:**

- Every user in the platform can be a content creator.
- The role of the content creator is to upload videos to the platform.
- The users can access the content uploaded by the content creator.

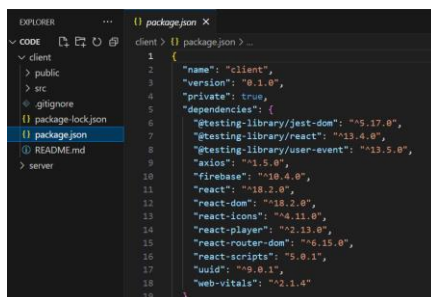
## Project Setup And Configuration:

- **Folder setup:**
- Create frontend and
- Backend folders
- **Installation of required tools:**

1. Open the frontend folder to install necessary tools

For frontend, we use:

- React
- Bootstrap
- Material UI
- Axios
- Firebase
- UUID
- React-bootstrap

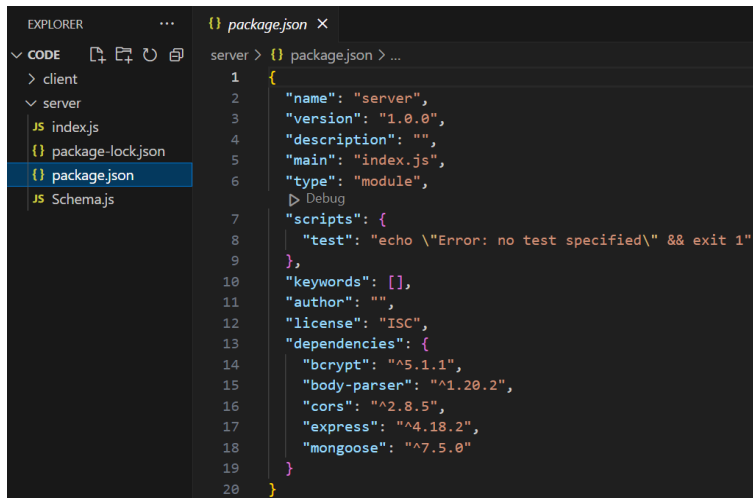


```
1 {
2   "name": "client",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.17.0",
7     "@testing-library/react": "^13.4.0",
8     "@testing-library/user-event": "^13.5.0",
9     "axios": "^1.3.0",
10    "firebase": "^10.4.0",
11    "react": "^18.2.0",
12    "react-dom": "^18.2.0",
13    "react-icons": "^4.11.0",
14    "react-player": "^2.13.0",
15    "react-router-dom": "^6.15.0",
16    "react-scripts": "^5.0.1",
17    "uuid": "^9.0.1",
18    "web-vitals": "^2.1.4"
19  },
20 }
```

## 2. Open the backend folder to install necessary tools

For backend, we use:

- cors
- bcryptjs
- express
- dotenv
- mongoose
- Nodemon
- Jsonwebtoken



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with a 'server' folder expanded, containing 'index.js', 'package-lock.json', 'package.json', and 'Schema.js'. The 'package.json' file is selected and its content is displayed in the main editor. The code is a JSON object defining the project's metadata and dependencies.

```
1 {
2   "name": "server",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "bcrypt": "^5.1.1",
15    "body-parser": "^1.20.2",
16    "cors": "^2.8.5",
17    "express": "^4.18.2",
18    "mongoose": "^7.5.0"
19  }
20 }
```

## Backend Development

### Setup express server

- Create index.js file in the server (backend folder).
- define port number, mongodb connection string and JWT key in env file to access it.
- Configure the server by adding cors, body-parser.

### Add authentication:

- For authentication, we need to define a url for login and register processes in the backend. By getting the data from a request from the client, we need to perform required operations.

### Error Handling:

- Implement error handling middleware to catch and handle any errors that occur during the API requests.
- Return appropriate error responses with relevant error messages and HTTP status codes.

## Database Development.Configure MongoDB

### Configure schema

- Import mongoose.
- Add database connection from config.js file present in config folder
- Create a model folder to store all the DB schemas user and blog schemas.

### Connect database to backend

- Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.

```
//  
const PORT = process.env.PORT || 6001;  
mongoose.connect(process.env.MONGO_URL, {  
  useNewUrlParser: true,  
  useUnifiedTopology: true  
}).then(()=>{  
  
  server.listen(PORT, ()=>{  
    console.log(`Running @ ${PORT}`);  
  });  
  
}).catch((err)=>{  
  console.log("Error: ", err);  
})
```

## Frontend Development

### Login/Register:

- Create a Component which contains a form for taking the username and password.
- If the given inputs matches the data of user, then navigate it to their respective home page

### About User:

- Create a about component which renders all the details of the users and also can change the profile details

### Statistics:

- Create a component which displays all the stats like views, comments , likes etc.

### Uploaded Videos:

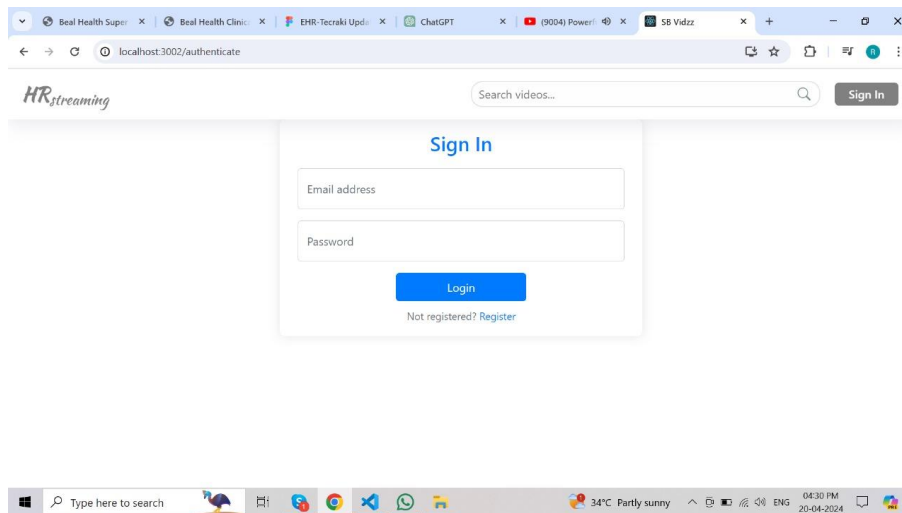
- Create an upload component which navigates to the page where it asks to upload a video file and add title, description and thumbnail to it.



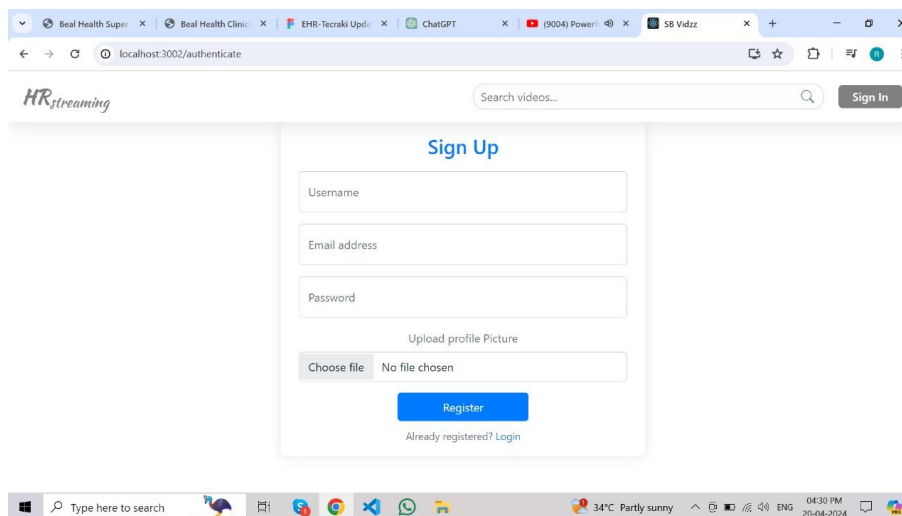
## Project Implementation:

Finally, after finishing coding the projects we run the whole project to test its working process and look for bugs. Now, let's have a final look at the working of our Video Sharing Application

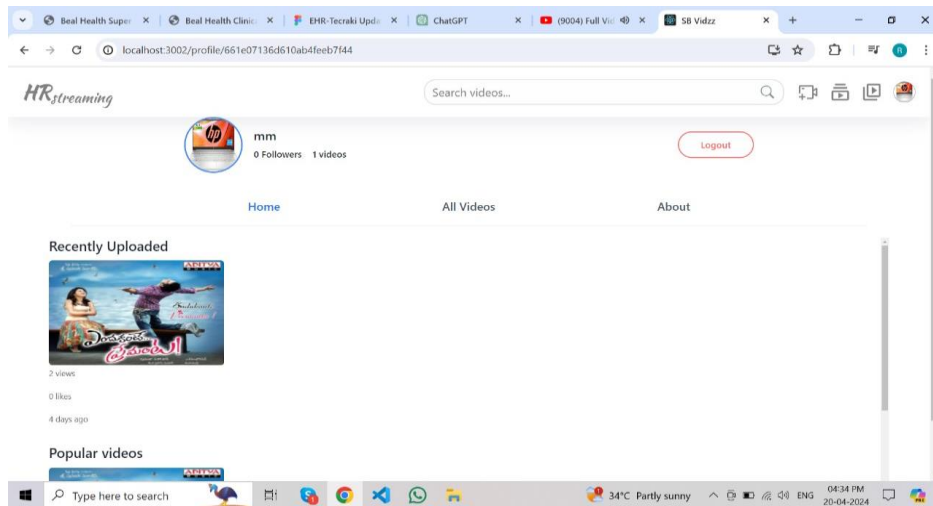
- **Login Page:**



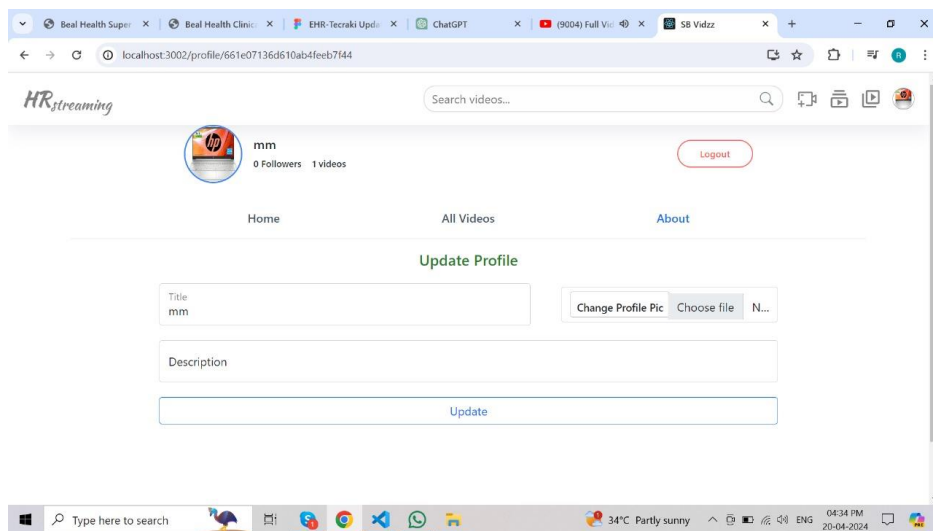
- **Registration Page:**



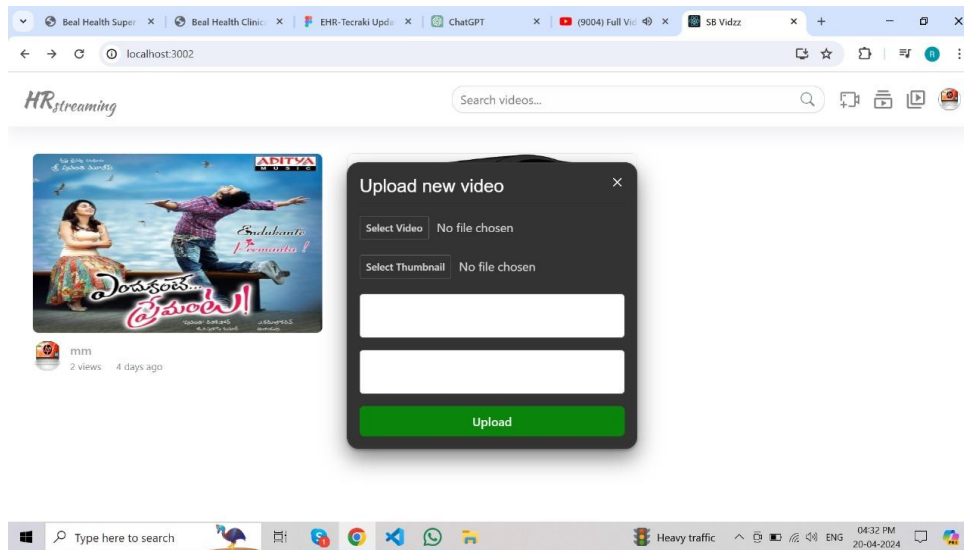
- Profile Page:



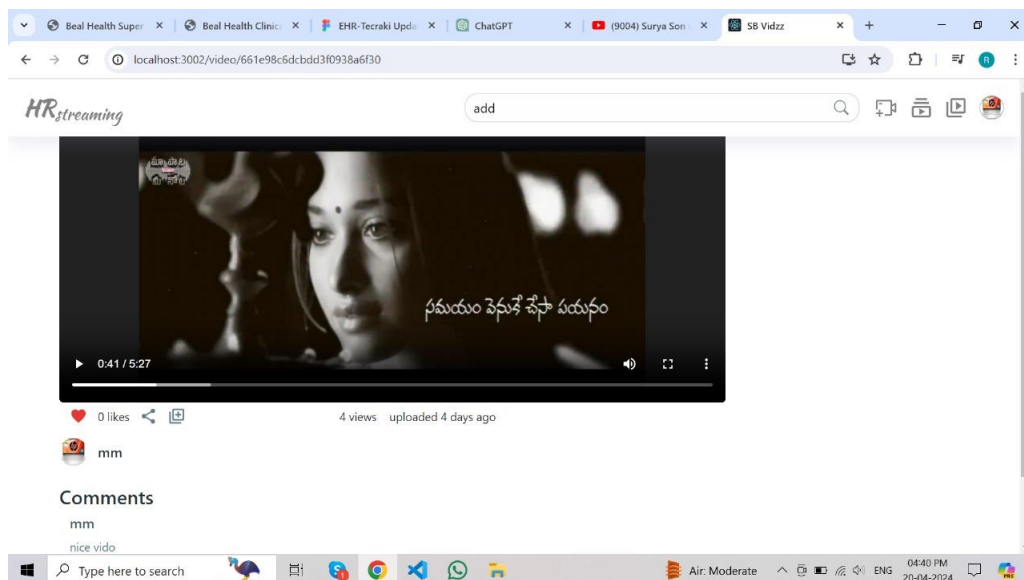
- Update Profile Page:



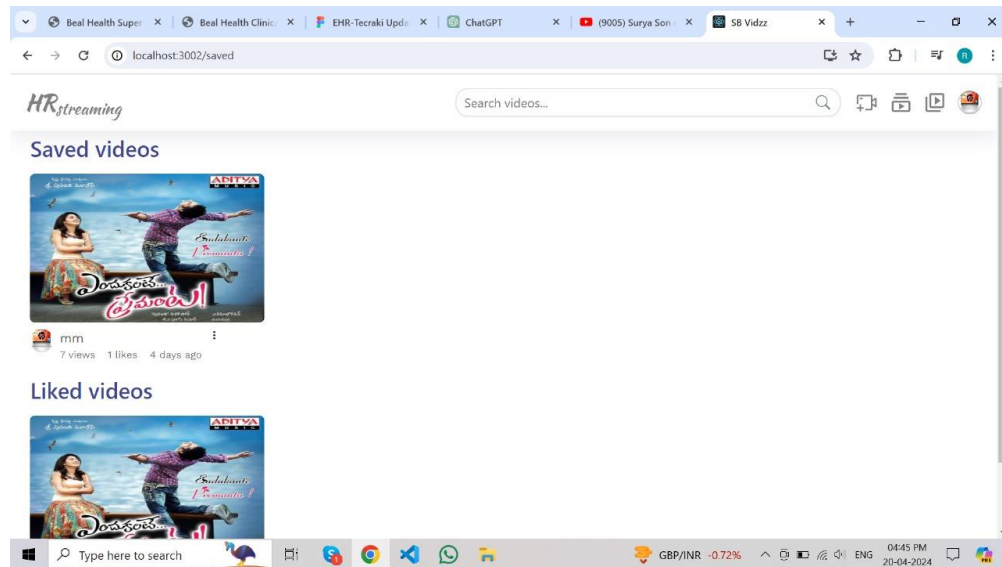
- **Upload Video Page:**



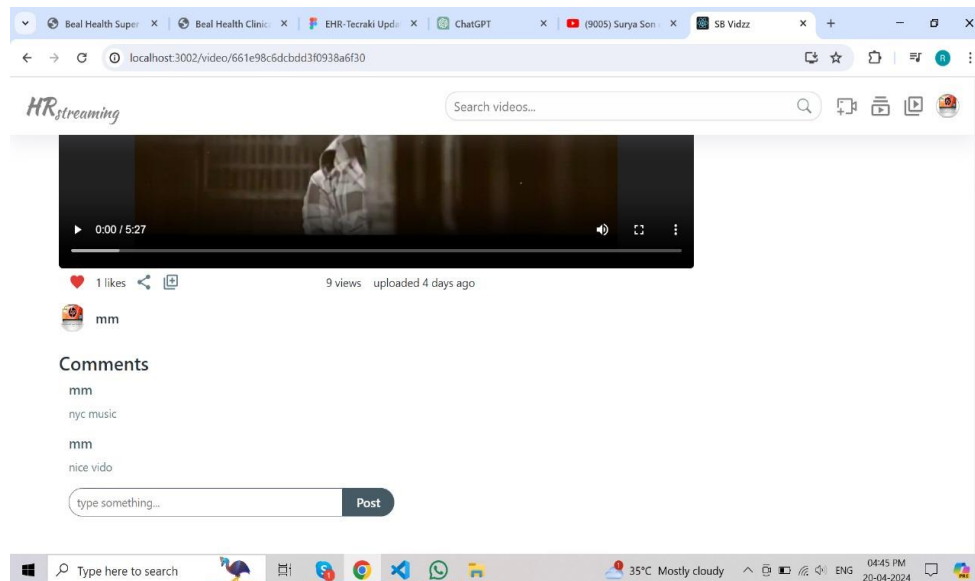
- **Like & Share video:**



- **Liked & Saved Videos:**



- **Comments:**



- **Conclusion:**

In conclusion, the video streaming app offers a seamless and convenient way for users to access a wide range of content. Its user-friendly interface, diverse content library, and robust streaming capabilities make it a compelling choice in today's digital landscape. With continuous updates and improvements, it stands poised to remain a top contender in the competitive streaming market.