# Full Stack Development with MERN

# Project Documentation format

**1. Introduction**

**Project Title**: Video Sharing App MERN
**Team Members**:

- Mohana Krishnan G
- Maadhavan S
- Vishal S
- Arshad Ariff

**2. Project Overview**

**Purpose**:
The Video Sharing Platform allows users to upload, share, and view videos online. The platform supports video streaming, user interactions, subscriptions, and content recommendations. The aim is to provide a seamless and engaging user experience for content creators and viewers alike .

**Features**:

- User Signup and Signin
- Video Upload and Processing
- Video Playback and Streaming
- User Comments and Interactions
- Search and Recommendations
- User Profile Management

**3. Architecture**

**Frontend**:
The frontend is built using React.js. It includes components like App.js for routing, various reusable UI components, and specific pages for different functionalities. State management is handled using Redux, and Material-UI is used for consistent and responsive design .

**Backend**:
The backend is implemented using Node.js and Express.js. Key components include controllers for handling different requests, models for database schemas, routes for defining API endpoints, and middleware for request processing. The backend communicates with MongoDB using Mongoose ODM .

**Database**:
The database schema includes collections for users, videos, and comments, each with relevant attributes and relationships. MongoDB is used as the database management system, with Mongoose as the object-document mapper(ODM).

**4. Setup Instructions**

**Prerequisites**:

- Node.js
- MongoDB

**Installation**:

1. Clone the repository: git clone
2. Navigate to the project directory: cd video-sharing-app
3. Install dependencies for the backend: npm install
4. Navigate to the client directory: cd client
5. Install dependencies for the frontend: npm install
6. Set up environment variables: Create a .env file in the root directory and add necessary configurations (e.g., database connection strings, API keys).

**5. Folder Structure**

**Client**:
The React frontend is structured into components, pages, and other necessary directories. Key components include Navbar, Video, Home, Signin, and more. Pages manage specific parts of the application like home, search results, and individual video pages .

**Server**:
The Node.js backend is organized into controllers, models, routes, and config directories. Each directory serves a specific purpose in handling requests, interacting with the database, and managing the application's logic .

**6. Running the Application**

**Frontend**:
Navigate to the client directory and run: npm start

**Backend**:
Navigate to the server directory and run: npm start

## 7. API Documentation

**Endpoints**:

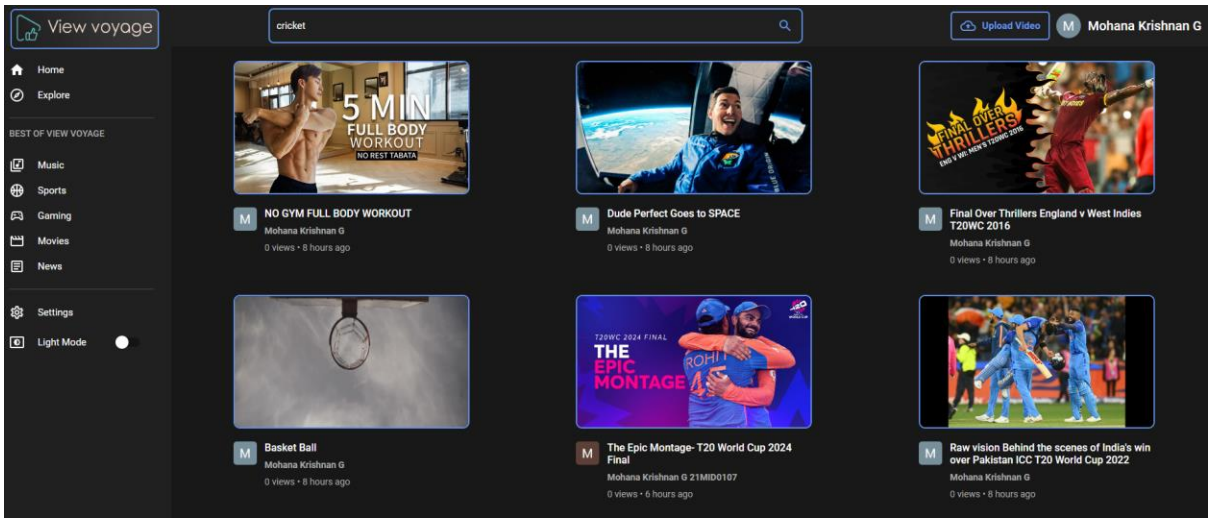
- **User Authentication**:
  - POST /api/auth/signup: Register a new user.
  - POST /api/auth/signin: Authenticates a user and returns a token.
- **User Management**:
  - GET /api/users/:id: Retrieves user information by ID.
  - PUT /api/users/:id: Updates user information by ID.
- **Videos**:
  - GET /api/videos: Retrieves all videos.
  - POST /api/videos: Uploads a new video.
  - GET /api/videos/:id: Retrieves a specific video by ID.
  - PUT /api/videos/:id: Updates a video by ID.

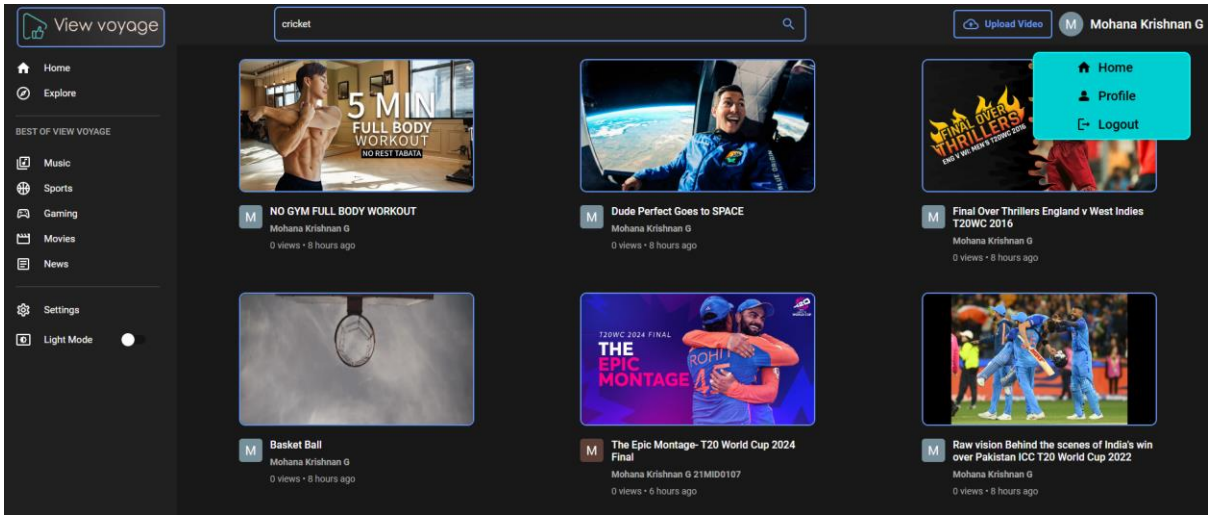
## 8. Authentication

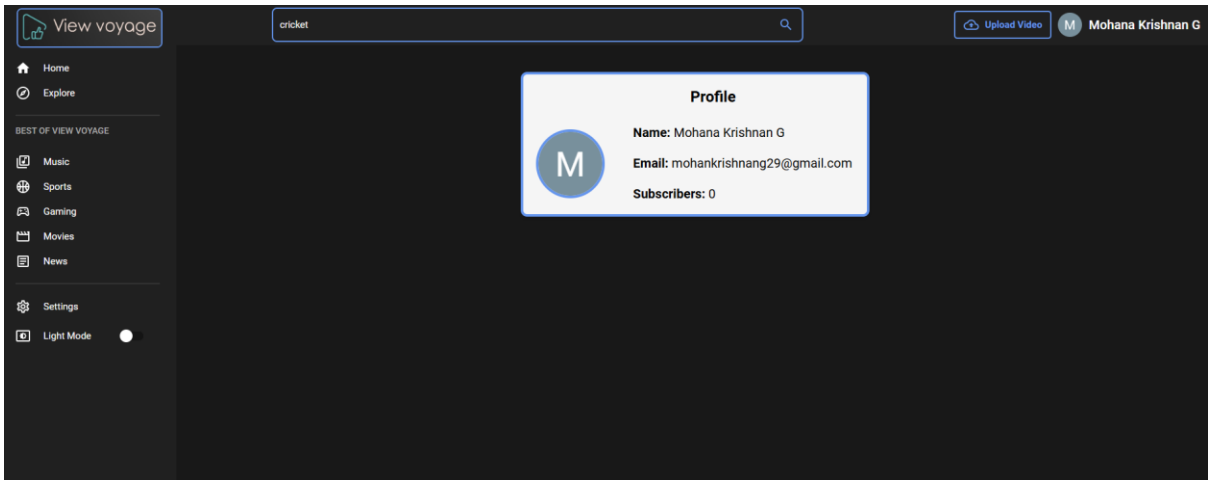
Authentication is handled using JSON Web Tokens (JWT). Tokens are issued upon user login and are used to verify user identity for subsequent requests. Middleware functions like verifyToken.js are used to protect routes and ensure only authenticated users can access certain endpoints.

## 9. User Interface

**Images:**






## 10. Testing

Testing strategy includes both unit and integration tests. Tools like Jest and Mocha could be used for writing and running tests. User Acceptance Testing (UAT) involves scenarios like user registration, login, video upload, and playback.

## 11. Screenshots or Demo

Link to DEMO Video - https://drive.google.com/file/d/1RRC0qlNYTrGOs2Q6bhEF48vO-qWUVIAe/view?usp=sharing

## 12. Known Issues

- Video upload fails for large files.
- Comments not appearing after submission.
- Subscription notifications not received.

**13. Future Enhancements**

- Implement more robust video processing and compression.
- Improve search algorithm for better relevance.
- Add more social features like sharing videos on external platforms.
- Enhance user profile customization options .