Technical Requirements Document (TRD) for Cinema Time

1. Introduction

Cinema Time is a MERN (MongoDB, Express, React, Node.js) stack web application designed to allow users to discover and explore movies, TV shows, and anime. It provides user registration and authentication, along with features such as watching trailers, viewing item details, adding items to favorites, searching for specific items, and managing user accounts. The application also includes an admin dashboard for managing items and viewing user data.

2. Architecture

Cinema Time follows a client-server architecture:

Frontend Architecture

- **Framework:** React.js
- **State Management:** React Context API
- **Styling:** Bootstrap, custom CSS
- **Form Validation:** Formik, Yup
- **UI Components:** Font Awesome, SweetAlert2

Backend Architecture

- **Framework:** Express.js
- **Database:** MongoDB
- **File Storage:** Cloudinary
- **Server:** Node.js
- **Live Reload:** Nodemon

3. Components and Integrations

Frontend Components

1. **App Component :** Main component handling routing and state management.
2. **AboutPage Component:** Displays information about the application.
3. **AdminPage Component :** Allows admins to manage items and view user data.
4. **DetailsPage Component :** Displays details of a specific movie, TV show, or anime.
5. **FavPage Component :** Displays user's favorite items.
6. **HomePage Component :** Displays homepage with featured items.
7. **ItemCard Component :** Displays a card for each item.
8. **ItemsPage Component :** Displays a list of items.
9. **LoginPage Component :** Handles user login.
10. **Navbar Component :** Navigation bar for easy navigation across the application.
11. **ProfilePage Component :** Displays user profile and allows account management.
12. **RegisterPage Component :** Handles user registration.

Backend Components

1. **Server:** Express.js server handling API requests.
2. **Routes:** Defines routes for user authentication, item management, and user management.
3. **Controllers:** Contains logic for handling API requests.
4. **Models:** Defines MongoDB schemas for users and items.
5. **Middleware:** Authentication middleware for verifying JWT tokens.
6. **File Upload:** Integration with Cloudinary for storing item images.

4. Context Providers

1. **Auth Context Provider:** Manages user authentication state and provides authentication-related functions to child components.
2. **Search Context Provider:** Manages search state and provides search-related functions to child components.

5. Infrastructure Requirements

Server Configuration

- **Server Type:** Node.js server
- **Server Framework:** Express.js
- **Database:** MongoDB
- **File Storage:** Cloudinary
- **Live Reload:** Nodemon

6. Development Tools, Frameworks, and Libraries Used

Frontend

- **React.js:** JavaScript library for building user interfaces.
- **Bootstrap:** Frontend framework for responsive design.
- **Axios:** Promise-based HTTP client for making API requests.
- **Formik:** Form library for React forms.
- **Yup:** JavaScript schema builder for form validation.
- **Context API:** React API for managing global state.
- **SweetAlert2:** Library for displaying beautiful alert messages.
- **Font Awesome:** Icon library for UI components.

Backend

- **Express.js:** Web application framework for Node.js.
- **Node.js:** JavaScript runtime for server-side development.
- **MongoDB:** NoSQL database for storing application data.
- **Cloudinary:** Cloud-based image and video management platform.
- **Nodemon:** Utility for automatically restarting the server during development.

7. APIs and Data Formats

Backend APIs

- **Authentication API: /api/user**
    - **POST /api/user/register**: Register a new user.
    - **POST /api/user/login**: Login an existing user.
    - **GET /api/user**: Get all users.
    - **GET /api/user/:email**: Get a specific user by email.
    - **PUT /api/user/:id**: Update an existing user by ID.
    - **DELETE /api/user/:id**: Delete a user by ID.
    - **POST /api/user/:userId/favorites/:itemId**: Add an item to favorites.
    - **GET /api/user/:userId/favorites**: Get all favorite items of a user.
    - **DELETE /api/user/:userId/favorites/:itemId**: Remove an item from favorites.
- **Item Management API: /api/item**
    - **GET /api/item/:category**: Get all items by category.
    - **GET /api/item/oneitem/:id**: Get a specific item by ID.
    - **DELETE /api/item/:id**: Delete an item by ID.
    - **PUT /api/item/:id**: Update an existing item by ID.
    - **POST /api/item**: Add a new item.
    - **GET /api/item**: Get count of all items.

Data Formats

- **Request/Response Format:** JSON
- **Authentication:** JWT Tokens
- **Data Validation:** JSON Schema

Conclusion

Cinema Time is a feature-rich web application built using modern web development technologies. Its modular architecture, along with well-defined APIs and data formats, allows for easy integration with external systems and future scalability. The infrastructure requirements are minimal, and the development tools and libraries used ensure efficient development and a seamless user experience.