**Movie Management System Documentation**

**Overview**

The Movie Management System is a Node.js application that allows users to manage a collection of movies in a MongoDB database. The system supports adding, deleting, updating, and retrieving movie records. This documentation provides an overview of the project structure, installation instructions, and details on each module.

**Project Structure**

The project consists of the following key files:

* create.js: Handles the insertion of new movie records into the database.
* delete.js: Handles the deletion of movie records based on specific conditions.
* update.js: Handles the updating of existing movie records.
* read.js: Handles the retrieval of movie records based on specified criteria.
* package-lock.json: Contains the dependency tree and versions of the packages used in the project.

**Dependencies**

The project uses several npm packages, as specified in the package-lock.json file:

* body-parser: Middleware to parse incoming request bodies.
* cors: Middleware to enable Cross-Origin Resource Sharing.
* express: Web framework for Node.js.
* mongodb: MongoDB driver for Node.js.

**Installation**

1. **Clone the repository**:

git clone <repository\_url>

cd movie-management-system

1. **Install the dependencies**:

npm install

1. **Start the server**:

node app.js

**Modules**

**create.js**

This module handles the addition of new movies to the database.

* **Dependencies**: mongodb
* **Endpoint**: POST /create
* **Request Body**:

{

"movie\_list": [

{

"title": "Movie Title",

"year": 2024,

"genre": "Drama",

"director": "Director Name"

}

]

}

* **Response**:

{

"message": "Movie(s) have been added to the database."

}

* **Error Handling**: Returns the error in the response if an error occurs during insertion.

**delete.js**

This module handles the deletion of movies from the database based on specified conditions.

* **Dependencies**: mongodb
* **Endpoint**: DELETE /delete
* **Request Body**:

{

"condition": {

"title": "Movie Title"

}

}

* **Response**:

{

"message": "The document(s) have been deleted."

}

* **Error Handling**: Returns the error in the response if an error occurs during deletion.

**update.js**

This module handles the updating of existing movie records in the database based on specified conditions.

* **Dependencies**: mongodb
* **Endpoint**: PUT /update
* **Request Body**:

{

"condition": {

"title": "Old Movie Title"

},

"new\_val": {

"title": "New Movie Title"

}

}

* **Response**:

{

"message": "The documents have been updated."

}

* **Error Handling**: Returns the error in the response if an error occurs during the update.

**read.js**

This module handles the retrieval of movie records from the database based on specified criteria.

* **Dependencies**: mongodb
* **Endpoint**: POST /read
* **Request Body**:

{

"title": "Movie Title"

}

* **Response**:

{

"movies\_list": [

{

"title": "Movie Title",

"year": 2024,

"genre": "Drama",

"director": "Director Name"

}

]

}

* **Error Handling**: Returns the error in the response if an error occurs during retrieval.

**Running the Application**

Ensure that MongoDB is running on your local machine or update the uri variable in each module to point to your MongoDB instance.

Start your server (assuming server.js is the entry point):

node app.js

You can then send requests to the specified endpoints using tools like Postman or CURL to manage your movie collection.

**Conclusion**

This documentation covers the basic setup and usage of the Movie Management System. For further customization, you can modify the MongoDB queries and enhance error handling as needed.