API Documentation for Movie Database Application

# Base URL

http://localhost:5000

## Authentication APIs

### 1. Login

- Endpoint: /login  
- Method: POST  
- Description: Authenticates a user using their email and password. Returns a JWT token on successful login.  
- Request Body:  
{  
 "email": "user@example.com", // The user's email  
 "password": "password123" // The user's password (plaintext)  
}  
- Response:  
 Success (200 OK):  
 {  
 "token": "your\_jwt\_token" // JWT token to be used in subsequent requests  
 }  
 Error (400 Bad Request):  
 {  
 "message": "Invalid credentials" // Invalid email or password  
 }  
 Error (500 Internal Server Error):  
 {  
 "message": "Server error"  
 }  
- Example:  
 POST /login  
 Content-Type: application/json  
 {  
 "email": "user@example.com",  
 "password": "password123"  
 }

### 2. Logout

- Endpoint: /logout  
- Method: POST  
- Description: Logs the user out by removing the JWT token. This requires the frontend to delete the token from storage (e.g., localStorage).  
- Request Body: None  
- Response:  
 Success (200 OK):  
 {  
 "message": "Logged out successfully"  
 }  
 Error (400 Bad Request):  
 {  
 "message": "Logout failed"  
 }  
- Example:  
 POST /logout

## Movie Management APIs

### 3. Get All Movies

- Endpoint: /movies  
- Method: GET  
- Description: Fetches a list of all movies stored in the database.  
- Request Body: None  
- Response:  
 Success (200 OK):  
 [  
 {  
 "\_id": "movie\_id\_1",  
 "title": "Movie Title",  
 "publishingYear": 2020,  
 "poster": "uploads/movie\_poster.jpg"  
 },  
 {  
 "\_id": "movie\_id\_2",  
 "title": "Another Movie",  
 "publishingYear": 2021,  
 "poster": "uploads/another\_movie\_poster.jpg"  
 }  
 ]  
 Error (400 Bad Request):  
 {  
 "message": "Unable to fetch movies"  
 }  
- Example:  
 GET /movies

### 4. Create a New Movie

- Endpoint: /movies  
- Method: POST  
- Description: Creates a new movie entry in the database. The poster image is uploaded as a binary file (multipart/form-data).  
- Request Body:  
{  
 "title": "Movie Title", // The title of the movie  
 "publishingYear": 2020, // The publishing year of the movie  
 "poster": "<binary\_image>" // The poster image (multipart/form-data)  
}  
- Response:  
 Success (201 Created):  
 {  
 "\_id": "new\_movie\_id",  
 "title": "Movie Title",  
 "publishingYear": 2020,  
 "poster": "uploads/movie\_poster.jpg"  
 }  
 Error (400 Bad Request):  
 {  
 "message": "Error creating movie"  
 }  
- Example:  
 POST /movies  
 Content-Type: multipart/form-data  
 {  
 "title": "Movie Title",  
 "publishingYear": 2020,  
 "poster": <image-file> // The image file should be attached here  
 }

### 5. Update an Existing Movie

- Endpoint: /movies/:id  
- Method: PUT  
- Description: Updates an existing movie. The poster image is optional and can be uploaded as a binary file.  
- Request Body:  
{  
 "title": "Updated Movie Title", // Updated title of the movie  
 "publishingYear": 2021, // Updated publishing year  
 "poster": "<binary\_image>" // Updated poster image (optional, can be binary)  
}  
- Response:  
 Success (200 OK):  
 {  
 "\_id": "existing\_movie\_id",  
 "title": "Updated Movie Title",  
 "publishingYear": 2021,  
 "poster": "uploads/updated\_movie\_poster.jpg"  
 }  
 Error (400 Bad Request):  
 {  
 "message": "Error updating movie"  
 }  
- Example:  
 PUT /movies/12345  
 Content-Type: multipart/form-data  
 {  
 "title": "Updated Movie Title",  
 "publishingYear": 2021,  
 "poster": <image-file> // The updated image file should be attached here  
 }

### 6. Delete a Movie

- Endpoint: /movies/:id  
- Method: DELETE  
- Description: Deletes a movie from the database by its id.  
- Request Body: None  
- Response:  
 Success (200 OK):  
 {  
 "message": "Movie deleted successfully"  
 }  
 Error (400 Bad Request):  
 {  
 "message": "Error deleting movie"  
 }  
- Example:  
 DELETE /movies/12345

## JWT Token Authentication

### How to Use JWT Token

- After logging in successfully, you will receive a JWT token.  
- This token should be included in the Authorization header for any protected routes that require authentication.  
Example:  
 Authorization: Bearer <your\_jwt\_token>

### Example Protected Route

If you wish to protect any of the movie-related routes (such as updating or deleting movies), you can use the following middleware to verify the JWT token:  
```js  
const jwt = require('jsonwebtoken');  
  
const verifyToken = (req, res, next) => {  
 const token = req.header('Authorization')?.replace('Bearer ', '');  
  
 if (!token) {  
 return res.status(401).json({ message: 'Access denied. No token provided.' });  
 }  
  
 try {  
 const decoded = jwt.verify(token, process.env.JWT\_SECRET);  
 req.user = decoded;  
 next(); // Proceed to the next middleware/route handler  
 } catch (error) {  
 res.status(400).json({ message: 'Invalid token' });  
 }  
};  
```