**IMDB-CLONE-API-ENDPOINTS**

1. Movie upload endpoint:-  
   [**URL:**/movie/upload](URL:/movie/upload)  
     
   **REQUEST(POST):  
   body(multipart/form-data):**{  
    title: “string” (title of the movie in 5 to 200 character),  
    description: “string” (description of the movie in 10 to 2000 character),  
    length: “integer” (length of the movie in mins from 30 to 180),  
    genres: “string” (space separated genres of the movie),  
    trailer: “media file” (video file of trailer in 5 to 200 mb),  
    poster: “media file” (image file of the poster in 50kb to 5mb)  
   }  
   **header:**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   NONE  
     
   **RESPONSE:  
   body(application/json):**{  
    id: “string” (id of the movie generated by the server),  
    title: “string” (title of the movie),  
    description: “string” (description of the movie),  
    timestamp: “long” (time when movie is saved),  
    length: “integer” (length of the movie in mins)  
   }  
     
   **status:**{  
    201: “when movie uploaded successfully”,  
    401: “when non admin requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
2. Movie info endpoint:-  
   [**URL:**/movie/info](URL:/movie/infou)  
     
   **REQUEST(GET):  
   body:**NONE  
   **header(optional):**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    movieId: “id of the movie to search”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    userInfo: {  
    id: “string”,  
    userId: “string”,  
    isAdmin: “boolean”,  
    profilePictureLink: “boolean”  
    },  
    movieInfo: {  
    id: “string”,  
    title: “string”,  
    description: “string”  
    timestamp: “long”,  
    length: “integer”,  
    posterLink: “string”,  
    trailerLink” “string”,  
    noOfReviews: “integer”,  
    noOfRatings: “integer”,  
    avgRating: “double”,  
    genres: “string”,  
    userRating: “integer”,  
    isWishListed: “boolean”,  
    isWatchListed: “boolean”  
    }  
   }  
     
   **status:**{  
    200: “when data fetch successfull”,  
    401: “when provided authorization headers(u\_id, Authorization) are wrong”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
3. Movie rating upload endpoint:-  
   [**URL:** movie/rating](URL:/movie/upload)  
     
   **REQUEST(PUT):  
   body(application/json):**{  
    rating: “integer” (contains the current user rating)   
   }  
   **header:**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    movieId: “id of the movie to upload rating”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    userId: “string”,  
    movieId: “string”,  
    rating: “integer”,  
    timestamp: “long”,  
    noOfRatings: “long”,  
    avgRating: “double”  
   }  
     
   **status:**{  
    200: “when rating uploaded successfully”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
4. User wishlist upload endpoint:-  
   [**URL:** user/wishlist](URL:/movie/upload)  
     
   **REQUEST(PUT):  
   body:**NONE  
   **header:**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    movieId: “id of the movie to upload to wishlist”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    sourceId: “string”,  
    targetId: “string”,  
    status: “boolean”,  
    size: “long”   
   }  
     
   **status:**{  
    200: “when uploaded successfully”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
5. User watchlist upload endpoint:-  
   [**URL:** user/watchlist](URL:/movie/upload)  
     
   **REQUEST(PUT):  
   body:**NONE  
   **header:**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    movieId: “id of the movie to upload to watchlist”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    sourceId: “string”,  
    targetId: “string”,  
    status: “boolean”,  
    size: “long”   
   }  
     
   **status:**{  
    200: “when uploaded successfully”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
6. User follow request endpoint:-  
   [**URL:** user/followrequest](URL:/movie/upload)  
     
   **REQUEST(PUT):  
   body:**NONE  
   **header:**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    targetUserId: “id of the user to follow”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    sourceId: “string”,  
    targetId: “string”,  
    status: “boolean”,  
    size: “long”   
   }  
     
   **status:**{  
    200: “when uploaded successfully”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
7. View user wish list endpoint:-  
   [**URL:** user/wishlist](URL:/movie/upload)  
     
   **REQUEST(GET):  
   body:**NONE  
   **header(optional):**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    userId: “id of the user to get wish list”,  
    pageId: “page id no of the wish list”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    id: “integer”,  
    size: “integer”,  
    length: “integer”,  
    result: [  
    {  
    id: “string”,  
    title: “string”,  
    description: “string”,  
    genres: “string”,  
    poster: “string”,  
    totalRating: “long”,  
    noOfRatings: “long”,  
    userRating: “integer”  
    }  
    ]  
   }  
     
   **status:**{  
    200: “when successfull”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
8. View user watch list endpoint:-  
   [**URL:** user/watchlist](URL:/movie/upload)  
     
   **REQUEST(GET):  
   body:**NONE  
   **header(optional):**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    userId: “id of the user to get watch list”,  
    pageId: “page id no of the watch list”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    id: “integer”,  
    size: “integer”,  
    length: “integer”,  
    result: [  
    {  
    id: “string”,  
    title: “string”,  
    description: “string”,  
    genres: “string”,  
    poster: “string”,  
    totalRating: “long”,  
    noOfRatings: “long”,  
    userRating: “integer”  
    }  
    ]  
   }  
     
   **status:**{  
    200: “when successfull”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
9. view user rating list endpoint:-  
   [**URL:** user/ratinglist](URL:/movie/upload)  
     
   **REQUEST(GET):  
   body:**NONE  
   **header(optional):**  
   {  
    u\_id: “string” (contains user id of the logged in user),  
    Authorization: “string” (contains jwt token of the logged in user)  
   }  
     
   **query:**  
   {  
    userId: “id of the user to get rating list”,  
    pageId: “page id no of the rating list”  
   }  
     
   **RESPONSE:  
   body(application/json):**{  
    id: “integer”,  
    size: “integer”,  
    length: “integer”,  
    result: [  
    {  
    id: “string”,  
    title: “string”,  
    description: “string”,  
    genres: “string”,  
    poster: “string”,  
    totalRating: “long”,  
    noOfRatings: “long”,  
    userRating: “integer”  
    }  
    ]  
   }  
     
   **status:**{  
    200: “when successfull”,  
    401: “when non authorized requests are made”,  
    400: “when something is wrong in the client request”,  
    500: “when something is wrong in the server side”  
   }
10. review upload endpoint:-  
    [**URL:** movie/review](URL:/movie/upload)  
      
    **REQUEST(POST):  
    body:**{  
     content: “string”  
    }  
    **header:**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     movieId: “id of the movie to upload review”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “string”,  
     userId: “string”,  
     movieId: “string”,  
     content: “string”,  
     timestamp: “long”  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
11. review reaction upload endpoint:-  
    [**URL:** movie/review/react](URL:/movie/upload)  
      
    **REQUEST(PUT):  
    body:**NONE  
    **header:**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     reviewId: “id of the review to upload reaction”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “string”,  
     userId: “string”,  
     timestamp: “long”,  
     status: “integer”,  
     noOfLikes: “long”   
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
12. view review endpoint:-  
    [**URL:** movie/review](URL:/movie/upload)  
      
    **REQUEST(GET):  
    body:**NONE  
    **header(optional):**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     reviewId: “id of the review to view”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     userInfo: {  
     id: “string”,  
     userId: “string”,  
     isAdmin: “boolean”,  
     profilePictureLink: “boolean”  
     },  
     review: {  
     id: “string”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     noOfReplies: “long”,  
     userReact: “boolean”,  
     replyList: {  
     id: “integer”,  
     size: “long”,  
     length: “long”,  
     result: [  
     {  
     id: “string”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     userReact: “boolean”  
     }  
     ]  
     }  
      
     }  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
13. reply upload endpoint:-  
    [**URL:** movie/reply](URL:/movie/upload)  
      
    **REQUEST(POST):  
    body:**{  
     content: “string”  
    }  
    **header:**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     reviewId: “id of the review to upload reply”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “string”,  
     userId: “string”,  
     reviewId: “string”,  
     content: “string”,  
     timestamp: “long”  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
14. reply reaction upload endpoint:-  
    [**URL:** movie/reply/react](URL:/movie/upload)  
      
    **REQUEST(PUT):  
    body:**NONE  
    **header:**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     replyId: “id of the reply to upload reaction”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “string”,  
     userId: “string”,  
     timestamp: “long”,  
     status: “integer”,  
     noOfLikes: “long”   
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
15. view reply endpoint:-  
    [**URL:** movie/reply](URL:/movie/upload)  
      
    **REQUEST(GET):  
    body:**NONE  
    **header(optional):**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     replyId: “id of the review to view”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “string”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     noOfReplies: “long”,  
     userReact: “boolean”,  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
16. view review list endpoint:-  
    [**URL:** movie/review](URL:/movie/upload)  
      
    **REQUEST(GET):  
    body:**NONE  
    **header(optional):**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     movieId: “id of the movie to get review”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “integer”,  
     size: “long”,  
     length: “long”,  
     result: [  
     {  
     id: “string”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     noOfReplies: “long”,  
     userReact: “boolean”,  
     replyList: {  
     id: “integer”,  
     size: “long”,  
     length: “long”,  
     result: [  
     {  
     id: “string”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     userReact: “boolean”  
     }  
     ]  
     }  
     }  
     ]  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
17. view reply list endpoint:-  
    [**URL:** movie/reply](URL:/movie/upload)  
      
    **REQUEST(GET):  
    body:**NONE  
    **header(optional):**  
    {  
     u\_id: “string” (contains user id of the logged in user),  
     Authorization: “string” (contains jwt token of the logged in user)  
    }  
      
    **query:**  
    {  
     reviewId: “id of the review to view reply list”  
    }  
      
    **RESPONSE:  
    body(application/json):**{  
     id: “integer”,  
     size: “long”,  
     length: “long”,  
     result: [  
     {  
     id: “integer”,  
     timestamp: “long”,  
     content: “string”,  
     userObject: {  
     id: “string”,  
     profilePictureLink: “string”  
     },  
     noOfLikes: “long”,  
     noOfReplies: “long”,  
     userReact: “boolean”,  
     }  
     ]  
    }  
      
    **status:**{  
     200: “when successfull”,  
     401: “when non authorized requests are made”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
18. user signup endpoint:-  
    [**URL:**/auth/signup](URL:/auth/signup)  
      
    **REQUEST(POST):  
    body(application/json):**{  
     userId: “string” (userid within 10 to 30 character),  
     password: “string” (password with atleast 10 character),  
     emailId: “string” (a valid email id)  
    }  
    **header:**  
    NONE  
      
    **query:**  
    NONE  
      
    **RESPONSE:  
    body(application/json):**{  
     u\_id: “string”,  
     token: “string”,  
     isAdmin: “boolean”  
    }  
      
    **status:**{  
     201: “when user uploaded successfully”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }
19. user signin endpoint:-  
    [**URL:**/auth/signin](URL:/auth/signin)  
      
    **REQUEST(POST):  
    body(application/json):**{  
     userId: “string” (userid within 10 to 30 character),  
     password: “string” (password with atleast 10 character),  
    }  
    **header:**  
    NONE  
      
    **query:**  
    NONE  
      
    **RESPONSE:  
    body(application/json):**{  
     u\_id: “string”,  
     token: “string”,  
     isAdmin: “boolean”  
    }  
      
    **status:**{  
     200: “when user signined successfully”,  
     400: “when something is wrong in the client request”,  
     500: “when something is wrong in the server side”  
    }

20.

user profile endpoint:-  
[**URL:**/user/profile](URL:/user/profile)  
  
**REQUEST(GET):  
body(application/json):**{  
 userId: “string” (userid within 10 to 30 character),  
 password: “string” (password with atleast 10 character),  
}  
**header:**  
NONE  
  
**query:**  
NONE  
  
**RESPONSE:  
body(application/json):**{

id: "string",

userId: "string",

profilePictureLink: "string",

noOfFollowers: ”integer”,

noOfFollowings: ”integer”,

noOfMovieReviewed: “integer”,

noOfMovieRated: “integer”,

wishListLength: ”integer”,

watchListLength: ”integer”,

ratingList: {

id: “integer”,

size: ”integer”,

length: “integer”,

result: [

{

id: "string",

title: "string",

description: "string",

genres: "string",

posterLink: "string",

avgRating: ”double”,

noOfRatings: “integer”,

userRating: “integer”

}

]

},

"wishList": {

id: “integer”,

size: ”integer”,

length: “integer”,

result: [

{

id: "string",

title: "string",

description: "string",

genres: "string",

posterLink: "string",

avgRating: ”double”,

noOfRatings: “integer”,

userRating: “integer”

}

]

},

"watchList": {

id: “integer”,

size: ”integer”,

length: “integer”,

result: [

{

id: "string",

title: "string",

description: "string",

genres: "string",

posterLink: "string",

avgRating: ”double”,

noOfRatings: “integer”,

userRating: “integer”

}

]

},

editable: ”boolean”,

admin: ”boolean”,

following: “boolean”

}  
  
**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

21. user quick info endpoint:-

[**URL:**/user/info](URL:/user/info)  
  
**REQUEST(GET):  
body(application/json):**{  
 userId: “string” (userid within 10 to 30 character),  
 password: “string” (password with atleast 10 character),  
}  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
{  
 userId: “string”  
}  
  
**RESPONSE:  
body(application/json):**{

id: "string",

userId: "string",

profilePictureLink: "string",

admin: ”boolean”,

}  
  
**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

22. search page endpoint:-

[**URL:**/movie/search](URL:/movie/search)  
  
**REQUEST(GET):  
body(application/json):**NONE  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
{  
 query: “string”,  
 pageId: “integer”  
}  
  
**RESPONSE:  
body(application/json):**{

"userInfo": {

"id": "string",

"userId": "string",

"profilePictureLink": "string",

"admin": “boolean”

},

"searchResult": {

"id": ”integer”,

"size": “integer”,

"length": ”integer”,

"result": [

{

"id": "string",

"title": "string",

"length": 129,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

}

}  
  
**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

23. home page endpoint:-

[**URL:**/movie/home](URL:/movie/home)  
  
**REQUEST(GET):  
body(application/json):**NONE  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
NONE  
  
**RESPONSE:  
body(application/json):**{

"userInfo": {

"id": "string",

"userId": "string",

"profilePictureLink": "string",

"admin": “boolean”

},

"recentlyAdded": {

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

},

"topRated": {

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

},

"leastRated": {

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

}  
}  
  
**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

24. recent added endpoint:-

[**URL:**/movie/recent](URL:/movie/recent)  
  
**REQUEST(GET):  
body(application/json):**NONE  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
NONE  
  
**RESPONSE:  
body(application/json):**{

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

}

**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

25. top rated endpoint:-

[**URL:**/movie/top](URL:/movie/top)  
  
**REQUEST(GET):  
body(application/json):**NONE  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
NONE  
  
**RESPONSE:  
body(application/json):**{

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

}

**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}

26. least rated endpoint:-

[**URL:**/movie/least](URL:/movie/least)  
  
**REQUEST(GET):  
body(application/json):**NONE  
**header(OPTIONAL):**  
{  
 u\_id: “string”,

Authorozation: “string”  
}  
  
**query(OPTIONAL):**  
NONE  
  
**RESPONSE:  
body(application/json):**{

"id": “integer”,

"size": “integer”,

"length": “integer”,

"result": [

{

"id": "string",

"title": "string",

"length": “integer”,

"posterLink": "string",

"noOfRatings": “integer”,

"avgRating": “double”,

"genres": "string",

"userRating": “integer”,

"wishListed": “boolean”,

"watchListed": “boolean”

}

]

}

**status:**{  
 200: “when user signined successfully”,  
 400: “when something is wrong in the client request”,  
 500: “when something is wrong in the server side”  
}