Tranquil Fortress App

The reason behind using this name for the app is, when I created an empty Heroku app, then I got this name randomly.

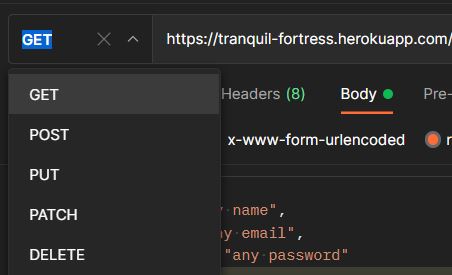
It is an imaginary app for renting a movie. (Although, real movies can be managed with this app by setting it up to a cloud storage server and adding support for movie upload)

Use postman to check the routes.

This app is running in cloud via Heroku and MongoDB Atlas

While testing routes using postman, pass objects in JSON format where needed.

In the pic below, you can see how to set method type as GET, POST, PUT, or DELETE.



When method of the request is GET or DELETE then you don’t need to pass anything in the body of the request as JSON object.

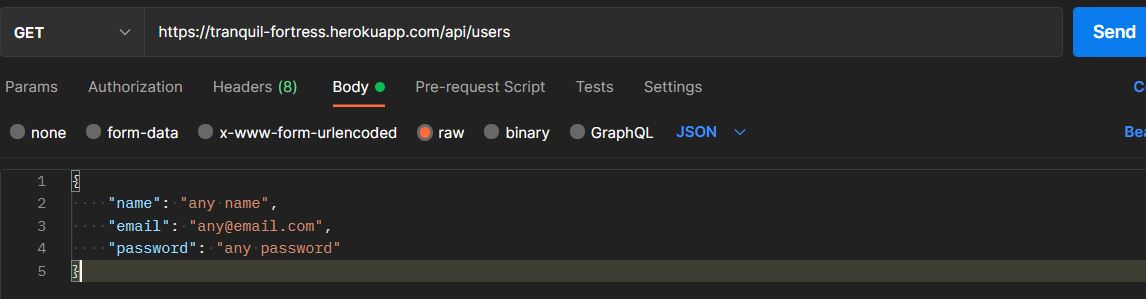
**API Routes:**

**NOTE:** Some api requests need the user to be signed in and some requests need the user to be both signed in and to be an admin. You cannot sign up as an admin directly. So, I’ve created an admin account and at the end of documentation you’ll find email and password for both admin and non-admin accounts to test the api routes.

* Users

This route is for signing up users.

1. POST <https://tranquil-fortress.herokuapp.com/api/users>



To sign up user. Pass name, email, and password in the body. Order doesn’t matter.

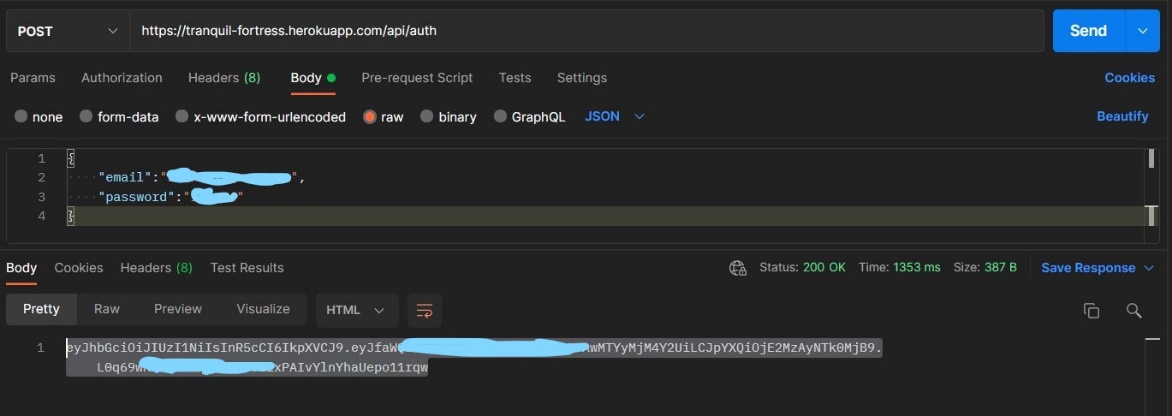
1. GET <https://tranquil-fortress.herokuapp.com/api/users/me>

To get the details of signed in user.

* Auth

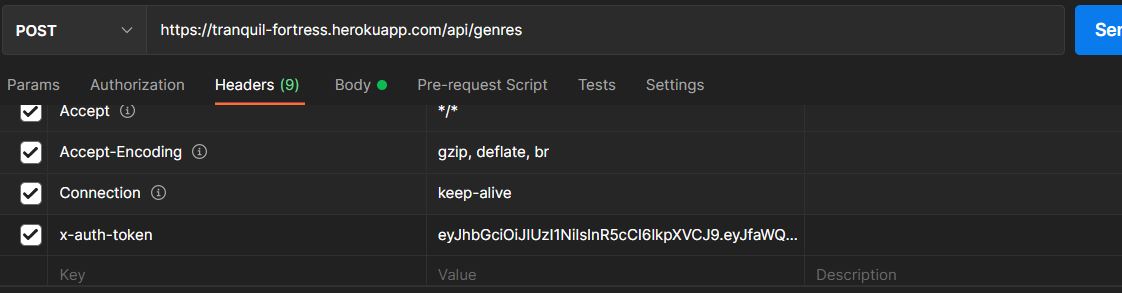
This Route is for signing in users.

1. POST <https://tranquil-fortress.herokuapp.com/api/auth>



For sign in, pass email and password to the route and it will return a token (JSON Web Token). Copy that and paste it in the headers section as a value and pass x-auth-token as a key parameter for token.

In the pic below you can see in the Headers section, pass x-auth-token as key and paste the token as value, and the in the body section pass the required object and send the request.



* Customers

This Route is for managing customers.

1. GET <https://tranquil-fortress.herokuapp.com/api/customers>

To get all the customers data, like customer id, name etc.

1. GET <https://tranquil-fortress.herokuapp.com/api/customers/id>

Search a customer by id. At the end of the URL replace the word id and paste the object id of the customer.

1. POST <https://tranquil-fortress.herokuapp.com/api/customers>

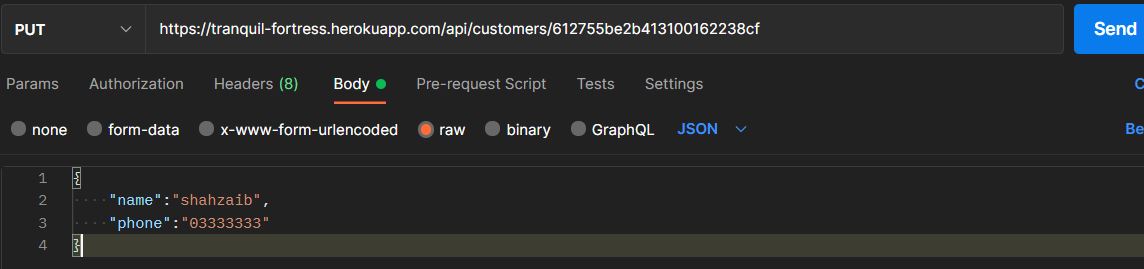
User should be signed in and be an admin to process this request.

Add a customer by passing name (customer name) and phone in the body of the request.

1. PUT <https://tranquil-fortress.herokuapp.com/api/customers/id>

User should be signed in and be an admin to process this request.

Edit any customer record. In the URL replace id with the customer’s object id. In the body of the request pass the new values. You can see the example below.



1. DELETE <https://tranquil-fortress.herokuapp.com/api/customers/id>

User should be signed in and be an admin to process this request.

Delete a customer by adding customer id at the end of URL.

* Genres

This Route is for managing genres.

1. GET <https://tranquil-fortress.herokuapp.com/api/genres>

Get all the genres record.

1. GET <https://tranquil-fortress.herokuapp.com/api/genres/id>

Find a genre by providing genre id at the end of URL

1. POST <https://tranquil-fortress.herokuapp.com/api/genres>

User should be signed in to process this request.

Add a new genre by passing name (Genre name) in the body of request.

1. PUT <https://tranquil-fortress.herokuapp.com/api/genres/id>

User should be signed in and be an admin to process this request.

Edit a genre by adding genre id at the end of URL and adding new value in the body of request.

1. DELETE <https://tranquil-fortress.herokuapp.com/api/genres/id>

User should be signed in and be an admin to process this request.

Delete a genre by adding genre id at the end of URL.

* Movies

This Route is for managing movies.

1. GET <https://tranquil-fortress.herokuapp.com/api/movies>

Get all the movies record.

1. GET <https://tranquil-fortress.herokuapp.com/api/movies/id>

Find a movie by providing movie id at the end of URL

1. GET <https://tranquil-fortress.herokuapp.com/api/movies/genre/id>

Get all movies by genre type by providing genre id at the end of URL

1. POST <https://tranquil-fortress.herokuapp.com/api/movies>

User should be signed in to process this request.

Add a new movie by passing (name (Movie name), genreId, numerbInStock, and dailyRentalRate) in the body of request.

Note: genreId should be valid and exist in the database.

For example:

{

    "title": "Any",

    "genreId": "612755be2b413100162238cf",

    "numberInStock": "12",

    "dailyRentalRate": "4"

  }

1. PUT <https://tranquil-fortress.herokuapp.com/api/movies/id>

User should be signed in and be an admin to process this request.

Edit a movie by adding movie id at the end of URL and adding new value in the body of request.

1. DELETE <https://tranquil-fortress.herokuapp.com/api/movies/id>

User should be signed in and be an admin to process this request.

Delete a movie by adding movie id at the end of URL.

* Rentals

This Route is for managing rentals.

1. GET <https://tranquil-fortress.herokuapp.com/api/rentals>

Get all the Rentals record.

1. GET <https://tranquil-fortress.herokuapp.com/api/rentals/id>

Find a rental by providing rental id at the end of URL

1. POST https://tranquil-fortress.herokuapp.com/api/rentals

Add a new Rental by passing (customerId, and movieId) in the body of request.

Note: Both customerId and movieId should be valid and exist in the database.

For example:

{

    "customerId":"612755be2b413100162238cf",

    "movieId":"61296654d64dbb001630c4b8"

}

After adding new rental the numberInStock of the movie will be reduced by 1.

* Returns

This Route is for returning rentals.

1. POST <https://tranquil-fortress.herokuapp.com/api/returns>

User should be signed in and be an admin to process this request.

A user can return a rental by passing valid customerId and movieId in the body of the request. The app will check if the rental is already returned else it will return the movie and the numberInStock of the movie will be increased by 1 and rental fee will be charged by calculating (rental date created - return date) x daily rental rate.

**Accounts for testing APIs:**

Non-admin account:

{

    "email":"shahzaib@any.com",

    "password":"123456"

}

Admin account:

{

    "email":"shahzeb@any.com",

    "password":"123456"

}