

Exercises 2.8

Installed mongoose

```
MINGW32:/c/Users/Jacqueline/Documents/GitHub/movie_api

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ npm -v
6.13.4

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ npm install mongoose --save
+ mongoose@5.8.11
added 19 packages from 13 contributors and audited 223 packages in 4.839s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ npm fund
movie_api@1.0.0
|-- mongoose@5.8.11
   |-- type: opencollective
   |-- url: https://opencollective.com/mongoose

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ npm install @types/mongoose
+ @types/mongoose@5.7.0
added 3 packages from 61 contributors and audited 229 packages in 3.103s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ npm fund
movie_api@1.0.0
|-- mongoose@5.8.11
   |-- type: opencollective
   |-- url: https://opencollective.com/mongoose

Jacqueline@Jacqueline MINGW32 ~/Documents/GitHub/movie_api (master)
$ |
```

Updated package.json:

```
Project — C:\Users\Jacqueline\Documents\GitHub\movie_api — Atom
File Edit View Selection Find Packages Help

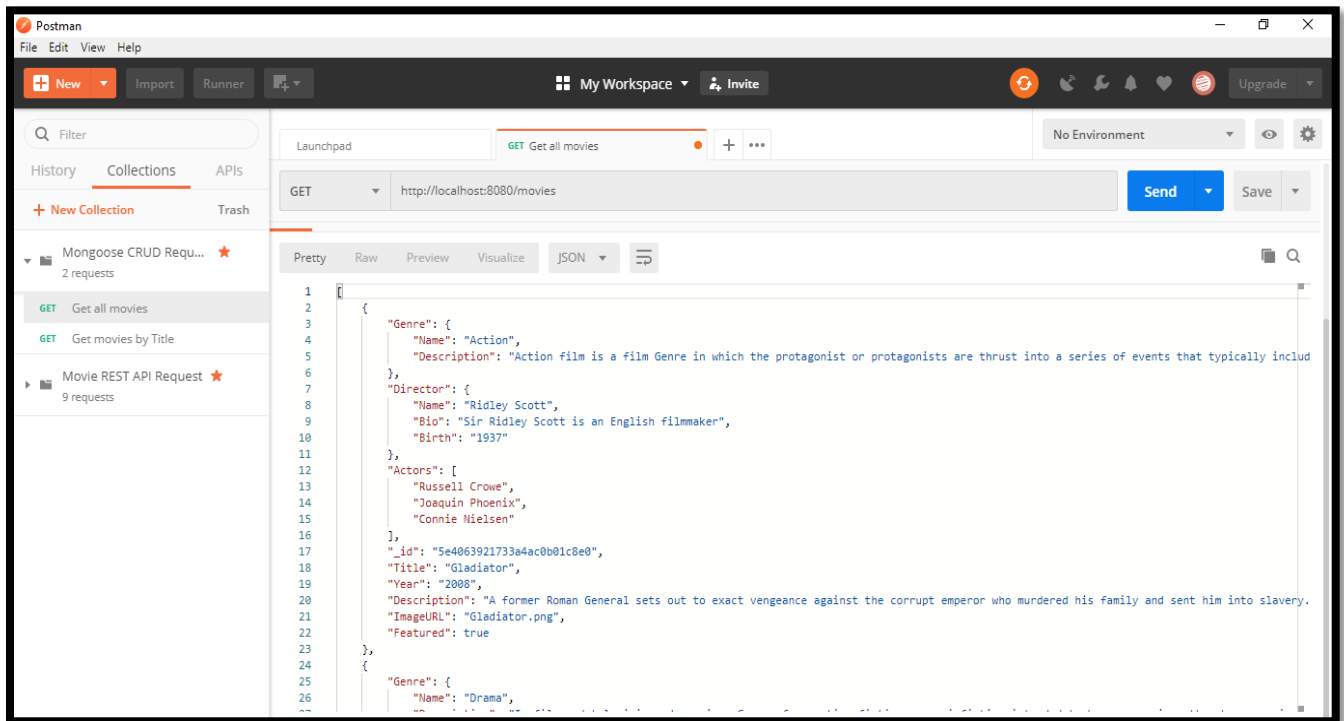
Project
  movie_api
  > .git
  > node_modules
  > public
  > > css
  > > img
  > > documentation.html
  .gitignore
  Exercise 2.7.zip
  index.js
  log.txt
  model.js
  mydata.txt
  package-lock.json
  package.json
  README.md

package-lock.json
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

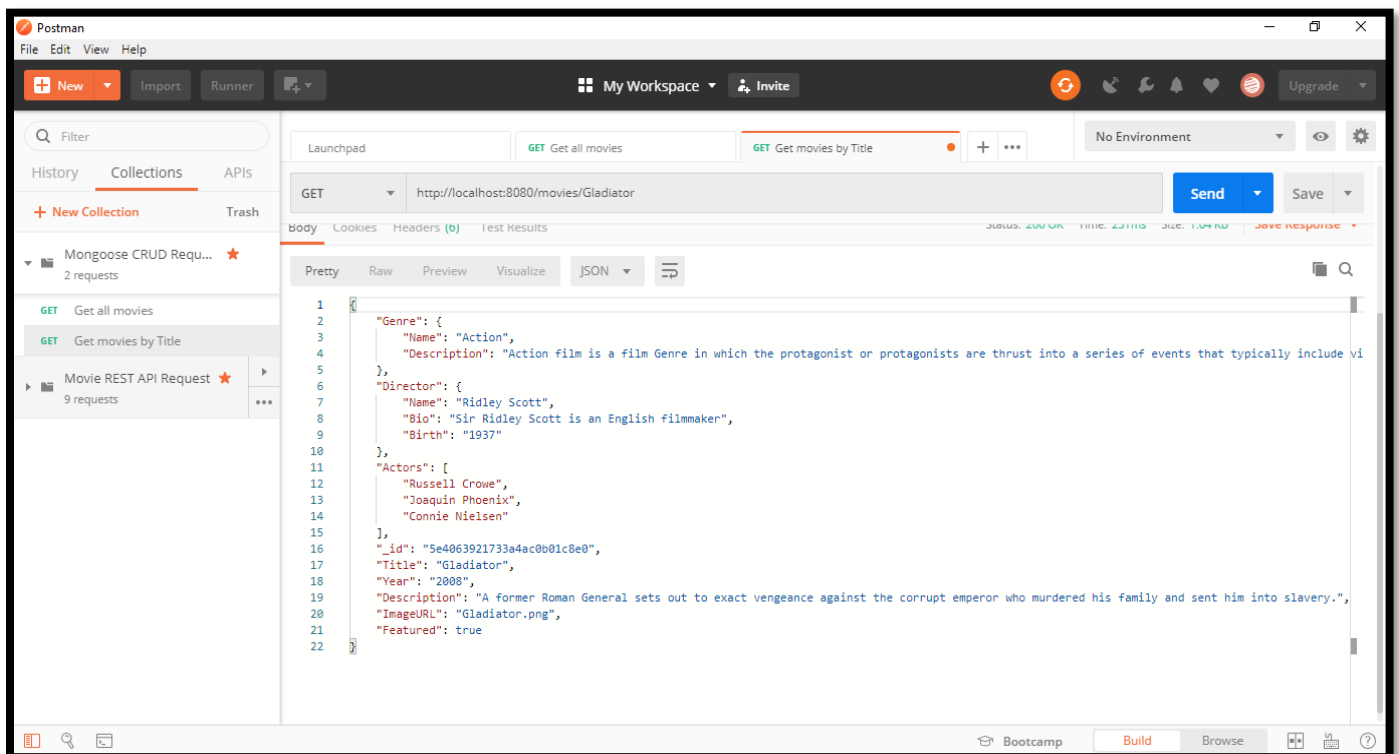
package.json
{
  "name": "movie_api",
  "version": "1.0.0",
  "description": "myFlix-database",
  "main": "index.js",
  "dependencies": {
    "@types/express": "^4.17.2",
    "@types/morgan": "^1.7.37",
    "@types/uuid": "^3.4.6",
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "mongoose": "^5.8.11",
    "morgan": "^1.9.1",
    "uuid": "^3.3.3"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/jtp2019/myFlix.git"
  },
  "keywords": [],
  "author": "Jacqueline Purification",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/jtp2019/myFlix/issues"
  },
  "homepage": "https://github.com/jtp2019/myFlix#readme"
}
```

Postman screenshots (12 requests)

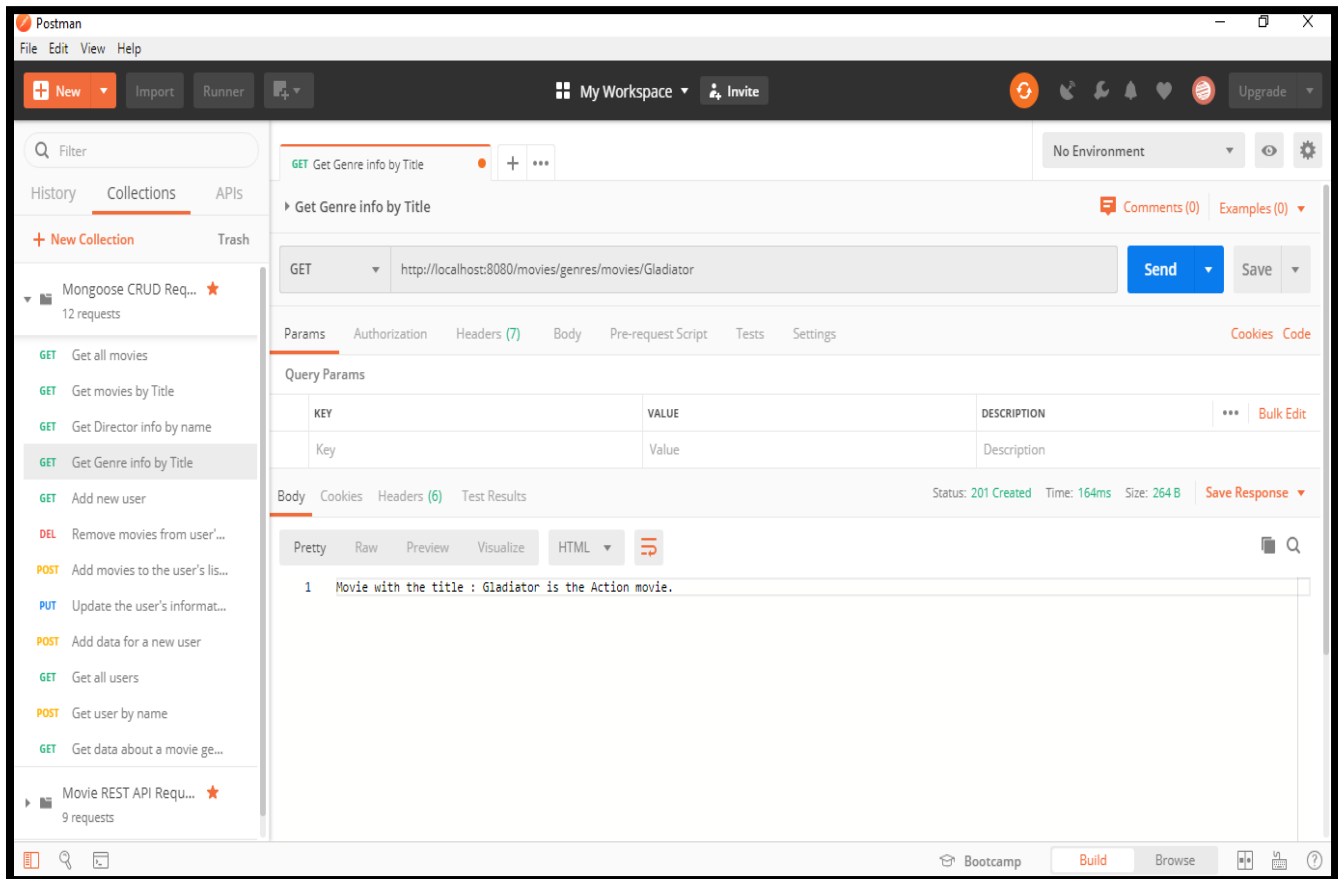
One: Return a list of ALL movies to the user:



Two: Return data (description, genre, director, image URL, whether it's featured or not) about a single movie by title to the user:



Three: Return data about a genre by the movie title (e.g., “Gladiator”):

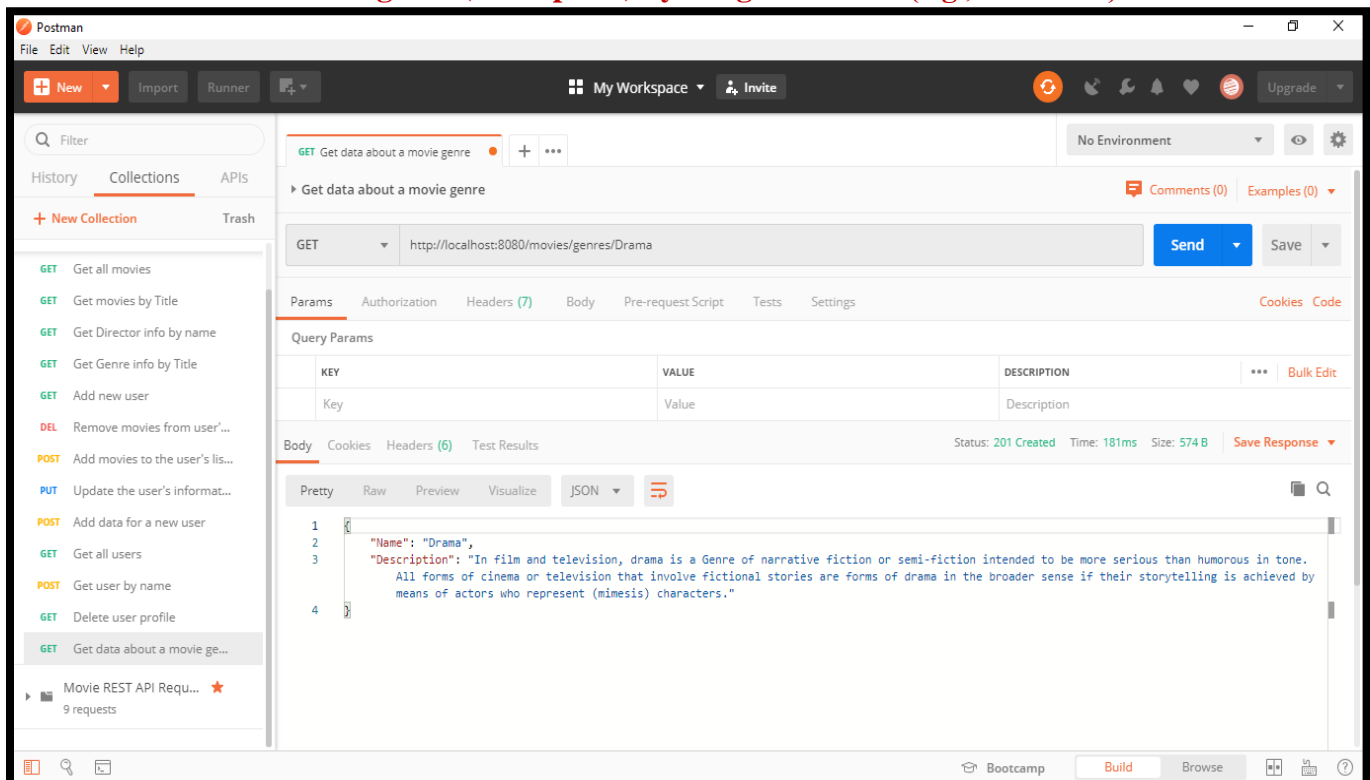


Postman interface showing a GET request to `http://localhost:8080/movies/genres/movies/Gladiator`. The response is a plain text message: "Movie with the title : Gladiator is the Action movie."

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body: 1 Movie with the title : Gladiator is the Action movie.

Four: Return data about a genre (description) by the genre Name (e.g., “Drama”):



Postman interface showing a GET request to `http://localhost:8080/movies/genres/Drama`. The response is a JSON object:

```
1 {
2   "Name": "Drama",
3   "Description": "In film and television, drama is a Genre of narrative fiction or semi-fiction intended to be more serious than humorous in tone.
4   All forms of cinema or television that involve fictional stories are forms of drama in the broader sense if their storytelling is achieved by means of actors who represent (mimesis) characters."
}
```

Five: Return data about a director (bio, birth year, death year) by name:

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/movies/directors/Steven Spielberg`. The response is a JSON object with the following data:

```
{
  "Name": "Steven Spielberg",
  "Bio": "Steven Allan Spielberg is an American director and producer",
  "Birth": "1946"
}
```

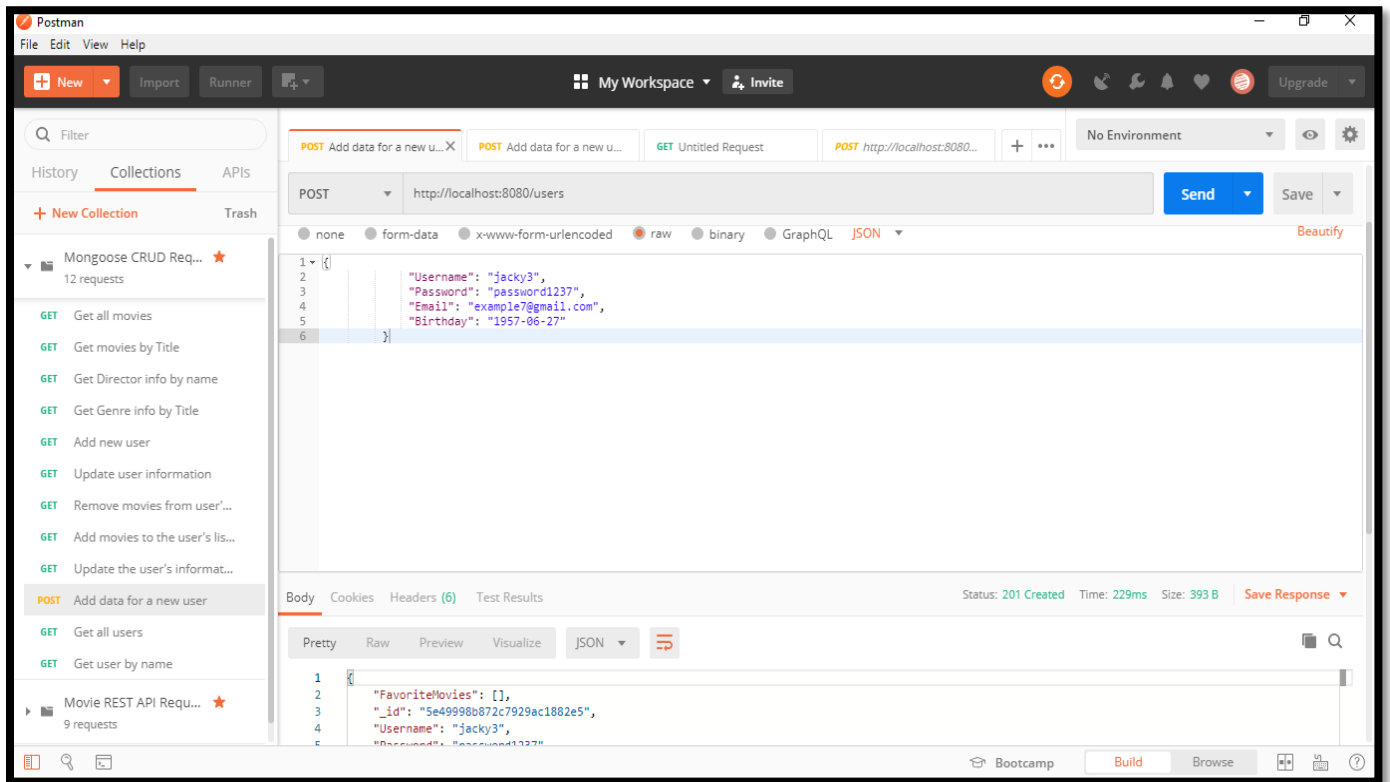
The status is 200 OK, with a response time of 86ms and a size of 323 B.

Six: Return a list of ALL users:

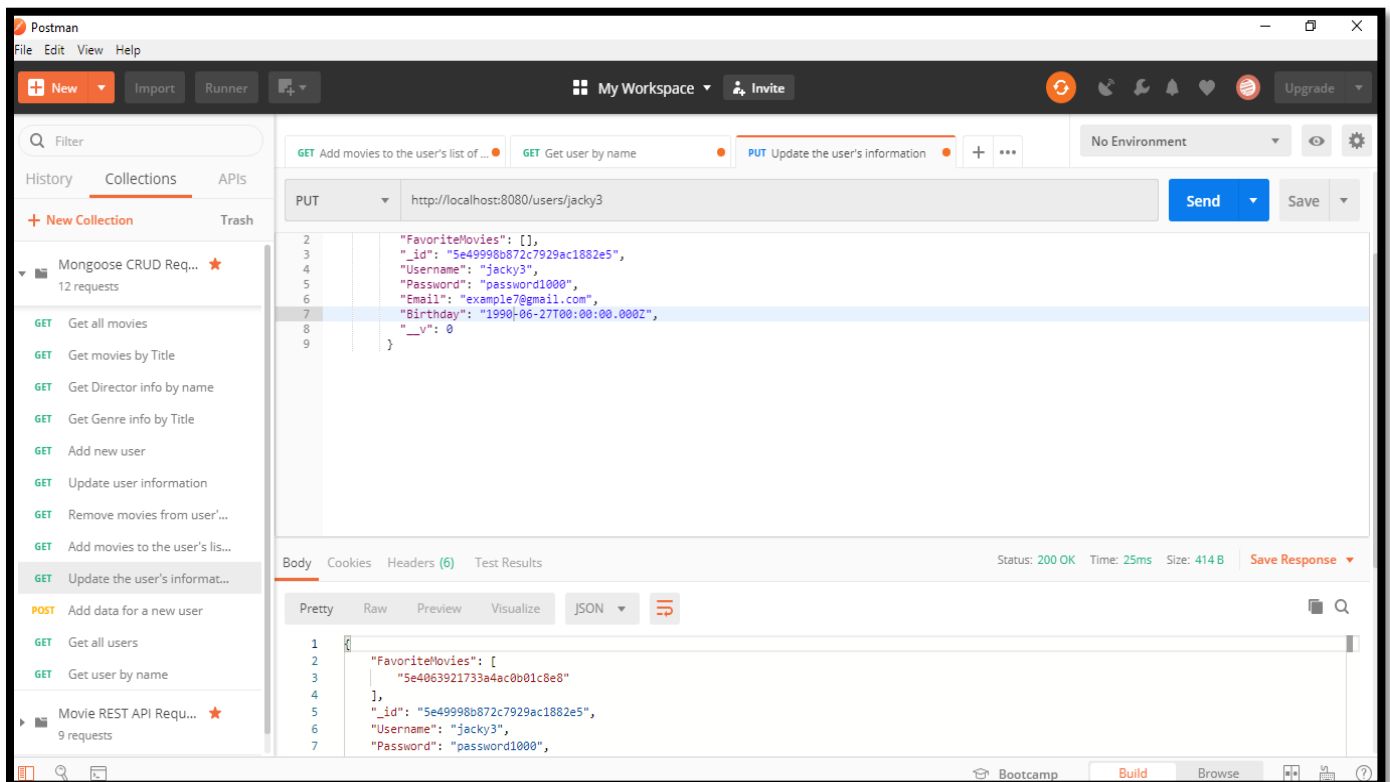
The screenshot shows the Postman interface with a GET request to `http://localhost:8080/users`. The response is a JSON array containing two user objects:

```
[
  {
    "FavoriteMovies": [
      "5e4063921733a4ac0b01c8e0",
      "5e4063921733a4ac0b01c8e1",
      "5e4063921733a4ac0b01c8e2"
    ],
    "_id": "5e44a1afc8733e0a91a9dab7",
    "username": "jonedoe123",
    "password": "password123",
    "email": "example1@gmail.com",
    "Birthday": "1977-02-19T00:00:00.000Z"
  },
  {
    "FavoriteMovies": [
      "5e4063921733a4ac0b01c8e3",
      "5e4063921733a4ac0b01c8e4",
      "5e4063921733a4ac0b01c8e5"
    ],
    "_id": "5e44a1afc8733e0a91a9dab8",
    "username": "jack1",
    "password": "password1234",
    "email": "example2@gmail.com",
    "Birthday": "1988-04-20T00:00:00.000Z"
  }
]
```

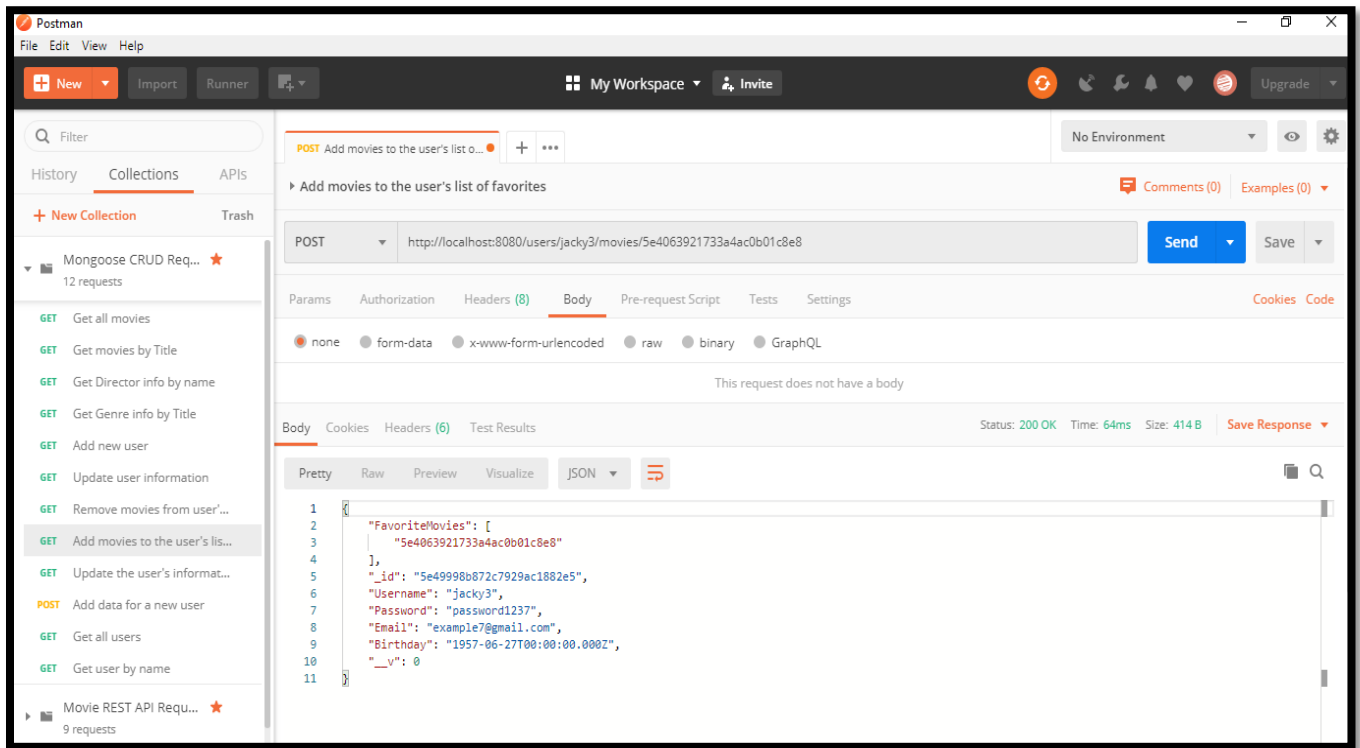
Seven: Allow new users to register:



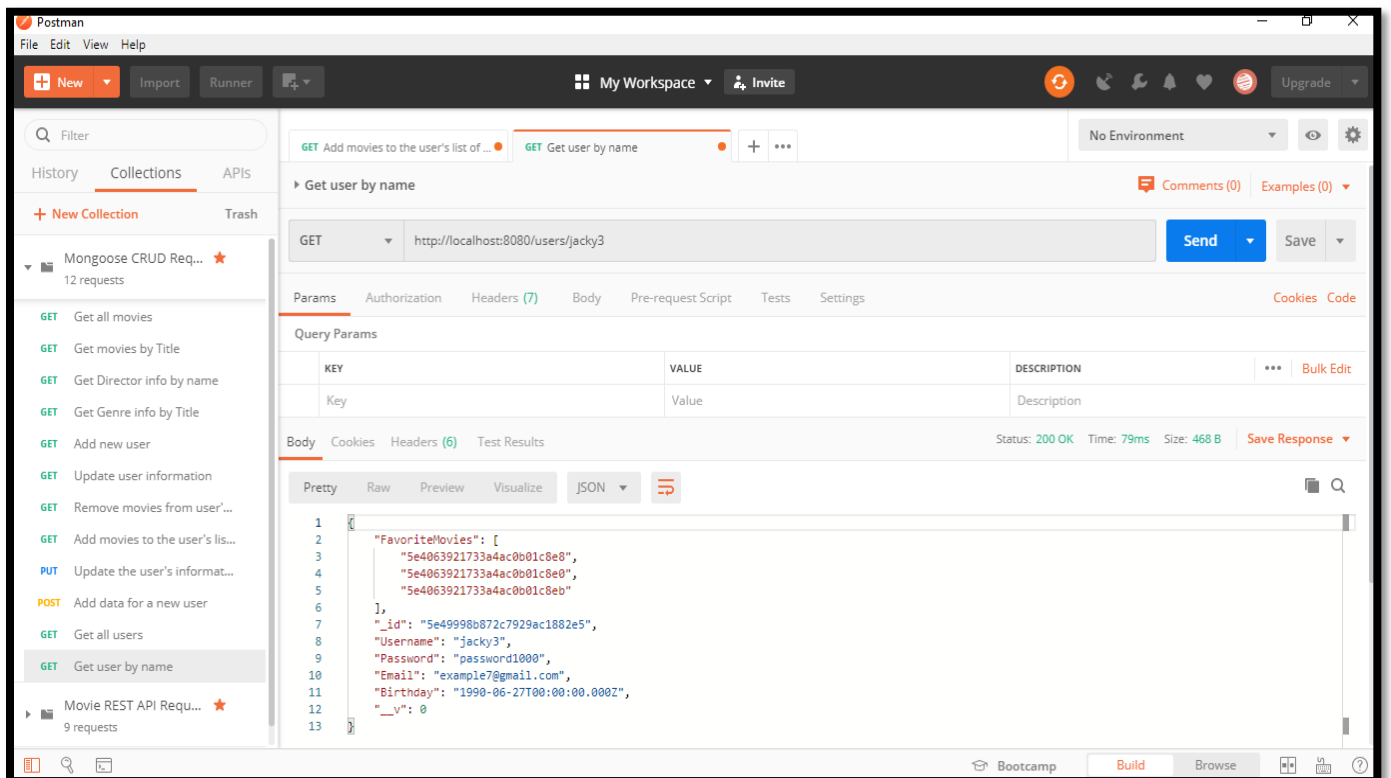
Eight: Allow users to update their user info (username, password, email, date of birth):



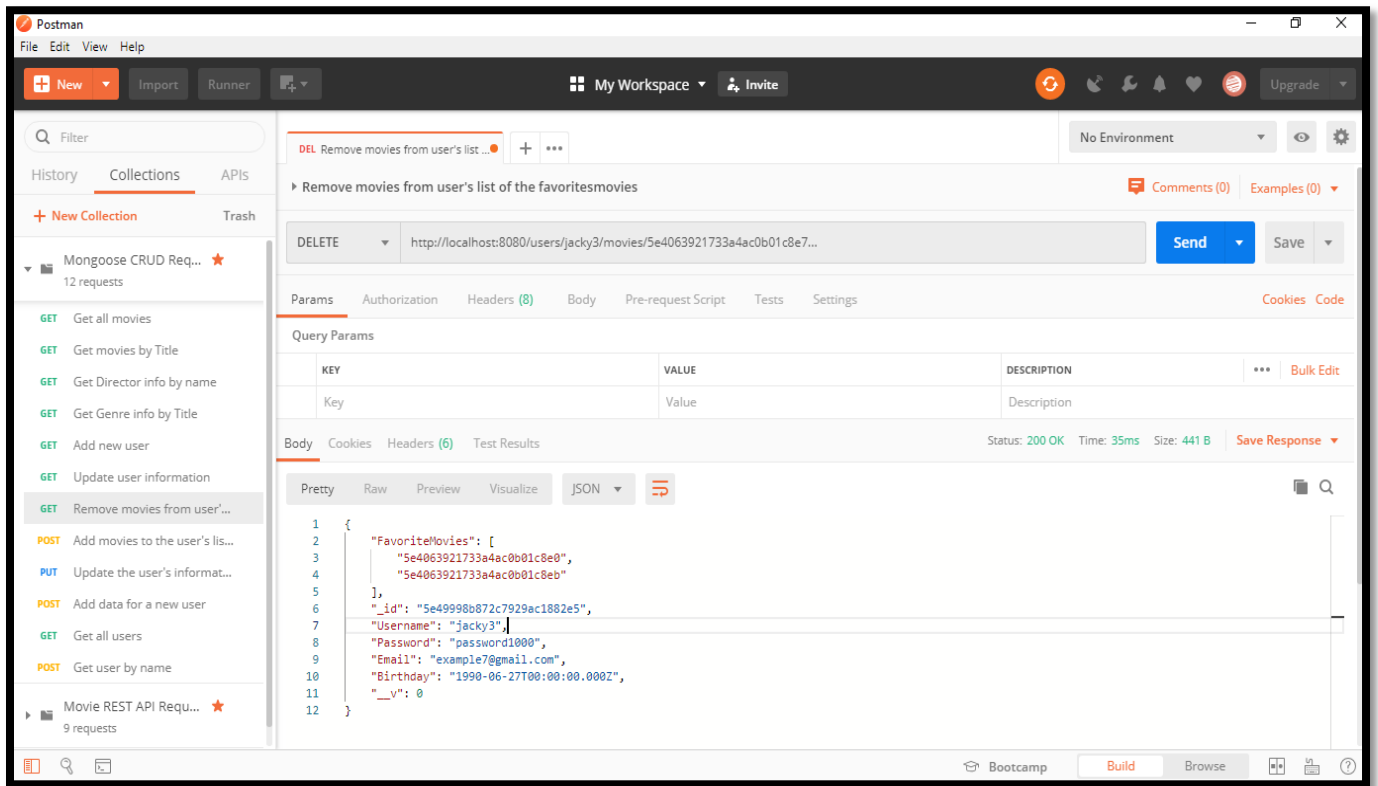
Nine: Allow users to add a movie to their list of favorites:



Ten: Get user by Name:



Eleven: Allow users to remove a movie from their list of favorites:



Twelve: Allow existing users to deregister:

