Prometheus

Single Point Multi Attack Server

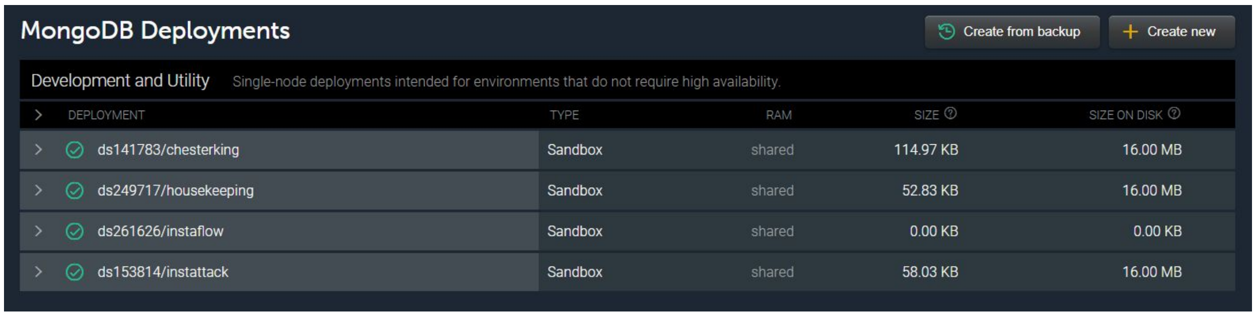
Name: Madhur Dixit
Registration Number: 17BCE2181
Lab Slot: L11+L12

<u>REVIEW -2</u>

Prometheus will be a website which attacks on the Facebook owned famous social media Instagram in two ways using automated bots masked as human actions. This can be achieved by using multiple ways like adding false cookies to the automated browser, switching off the property of webdriver of the bot. Bouncing the IP of the server. Each user will be able to login using credentials and then there are two different forms for two different kinds of attacks. At the current state instagram does not allow multiple acceptance of follow requests. One of the form bypasses this restriction of instagram by automating the process without instagram's awareness that it is actually a bot accepting all the follow request one by one. The second process is the liking N number of posts of a particular hashtag on instagram which makes the profile more vibrant and other people start to take notice and follow your profile. The main functionality lies in the backend but for database we are saving the requests made in a remote cloud mongo database MLab which is a NoSQL database.

Comprehensive Database Design:

Database used in this project is MongoDB which is a NoSQL database. The database will be hosted in a cloud service like GCP/AWS/Azure which can be accessed by MLabs. The connection between database and client-side app is highly secured as authentication and authorization is taken care by mongoDB, so the user doesn't need to worry about database security.
There are a total of 3 collections in the database which are ***Users***, ***Folls*** and ***Reqts***.

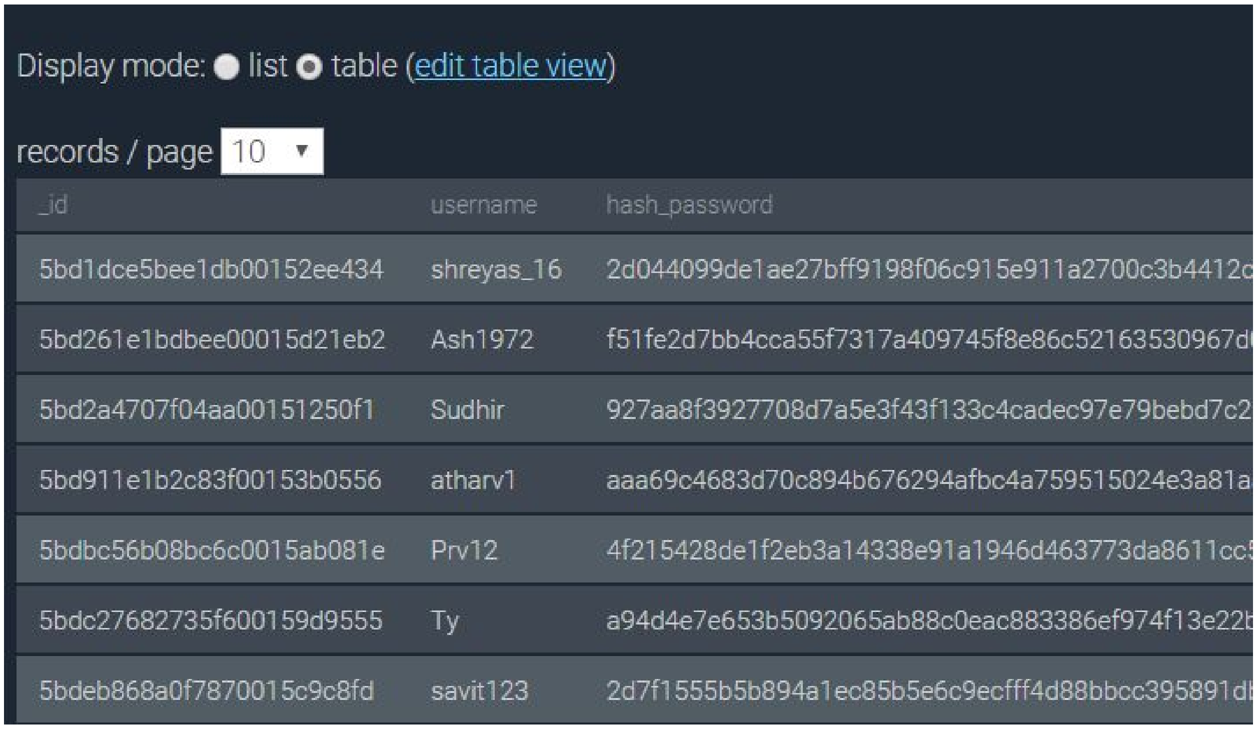

MongoDB Deployments

Development and Utility   Single-node deployments intended for environments that do not require high availability.

DEPLOYMENT	TYPE	RAM	SIZE	SIZE ON DISK
ds141783/chesterking	Sandbox	shared	114.97 KB	16.00 MB
ds249717/housekeeping	Sandbox	shared	52.83 KB	16.00 MB
ds261626/instaflow	Sandbox	shared	0.00 KB	0.00 KB
ds153814/instattack	Sandbox	shared	58.03 KB	16.00 MB

Here you can see the fourth database is the insta attack database which is been used by the web application made by me for all the database operations. All the database happen here in the fourth “instattack” database which runs in a sandbox and uses Amazon.

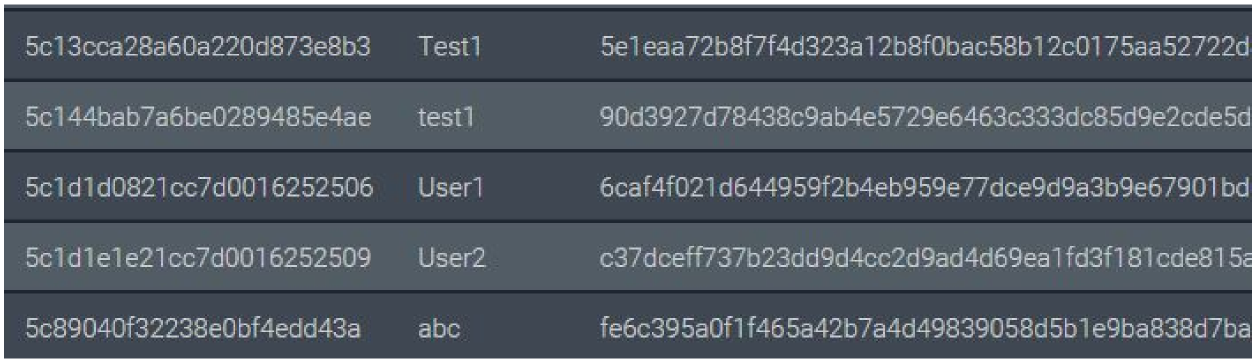
One for Users which is used for Signing up and Logging in to the website. You cannot access the services unless you are logged in to the website. The server uses this collection to authenticate the user while signing in and in case of Sign up it adds a new entry to this collection.



Display mode: list [✓ table] (edit table view)

records / page 10

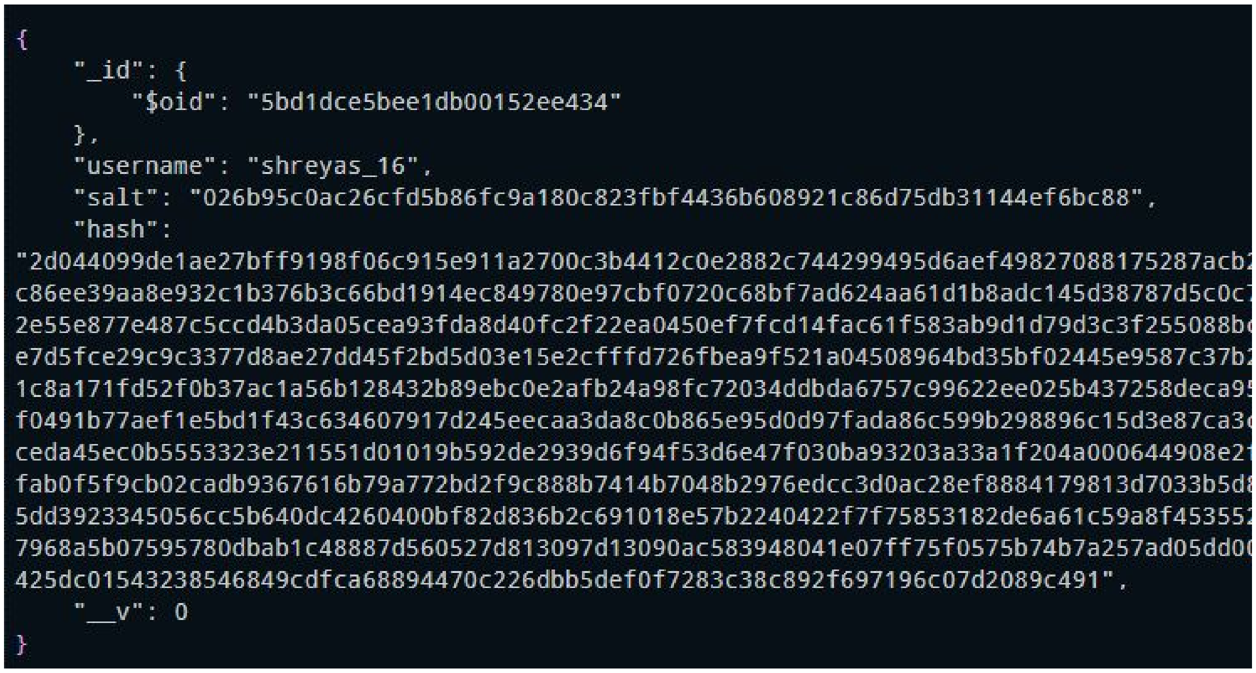
_id	username	hash_password
5bd1dce5bee1db00152ee434	shreyas_16	2d044099de1ae27bff9198f06c915e911a2700c3b4412c
5bd261e1bdbee00015d21eb2	Ash1972	f51fe2d7bb4cca55f7317a409745f8e86c52163530967d
5bd2a4707f04aa00151250f1	Sudhir	927aa8f3927708d7a5e3f43f133c4cadec97e79bebd7c2
5bd911e1b2c83f00153b0556	atharv1	aaa69c4683d70c894b676294afbc4a759515024e3a81a
5bdbc56b08bc6c0015ab081e	Prv12	4f215428de1f2eb3a14338e91a1946d463773da8611cc
5bdc27682735f600159d9555	Ty	a94d4e7e653b5092065ab88c0eac883386ef974f13e22b
5bdeb868a0f7870015c9c8fd	savit123	2d7f1555b5b894a1ec85b5e6c9ecfff4d88bbcc395891db



5c13cca28a60a220d873e8b3	Test1	5e1eaa72b8f7f4d323a12b8f0bac58b12c0175aa52722d
5c144bab7a6be0289485e4ae	test1	90d3927d78438c9ab4e5729e6463c333dc85d9e2cde5d
5c1d1d0821cc7d0016252506	User1	6caf4f021d644959f2b4eb959e77dce9d9a3b9e67901bd
5c1d1e1e21cc7d0016252509	User2	c37dceff737b23dd9d4cc2d9ad4d69ea1fd3f181cde815a
5c89040f32238e0bf4edd43a	abc	fe6c395a0f1f465a42b7a4d49839058d5b1e9ba838d7ba

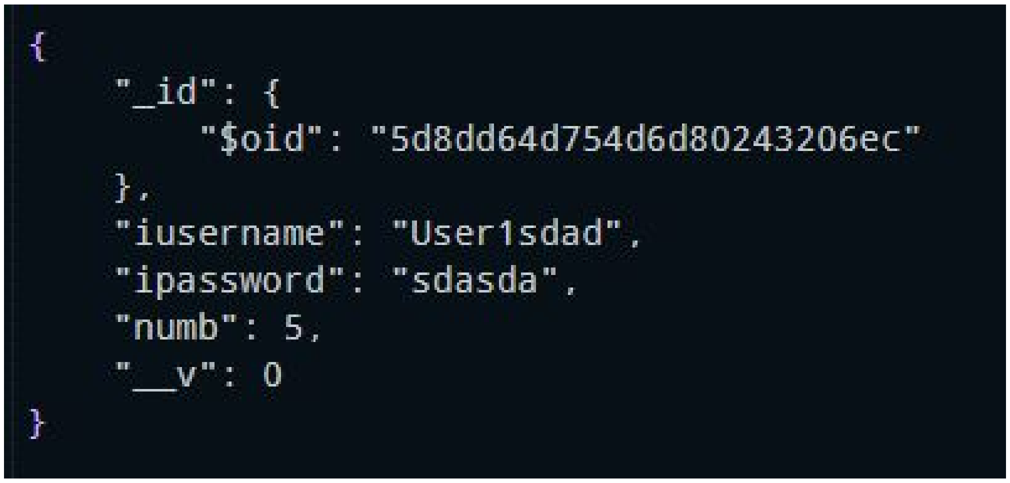
Here is the User Document enteries in the remote database. As you can see here the password is hashed. So in case the database gets stolen then also the users privacy is protected.\

Users collection is used to store all the information related to a user like username, salt, hashed password etc. Each user document will have a unique id stored in _id field, which will be generated by mongoDB itself.
Structure of user document is represented in the image.



```
{
    "_id": {
        "$oid": "5bd1dce5bee1db00152ee434"
    },
    "username": "shreyas_16",
    "salt": "026b95c0ac26cfd5b86fc9a180c823fbf4436b608921c86d75db31144ef6bc88",
    "hash":
"2d044099de1ae27bff9198f06c915e911a2700c3b4412c0e2882c744299495d6aef49827088175287acb
c86ee39aa8e932c1b376b3c66bd1914ec849780e97cbf0720c68bf7ad624aa61d1b8adc145d38787d5c0c
2e55e877e487c5ccd4b3da05cea93fda8d40fc2f22ea0450ef7fcd14fac61f583ab9d1d79d3c3f255088b
e7d5fce29c9c3377d8ae27dd45f2bd5d03e15e2cfffd726fbea9f521a04508964bd35bf02445e9587c37b
1c8a171fd52f0b37ac1a56b128432b89ebc0e2afb24a98fc72034ddbda6757c99622ee025b437258deca9
f0491b77aef1e5bd1f43c634607917d245eecaa3da8c0b865e95d0d97fada86c599b298896c15d3e87ca3
ceda45ec0b5553323e211551d01019b592de2939d6f94f53d6e47f030ba93203a33a1f204a000644908e2
fab0f5f9cb02cadb9367616b79a772bd2f9c888b7414b7048b2976edcc3d0ac28ef8884179813d7033b5d
5dd3923345056cc5b640dc4260400bf82d836b2c691018e57b2240422f7f75853182de6a61c59a8f45355
7968a5b07595780dbab1c48887d560527d813097d13090ac583948041e07ff75f0575b74b7a257ad05dd0
425dc01543238546849cdfca68894470c226dbb5def0f7283c38c892f697196c07d2089c491",
    "__v": 0
}
```

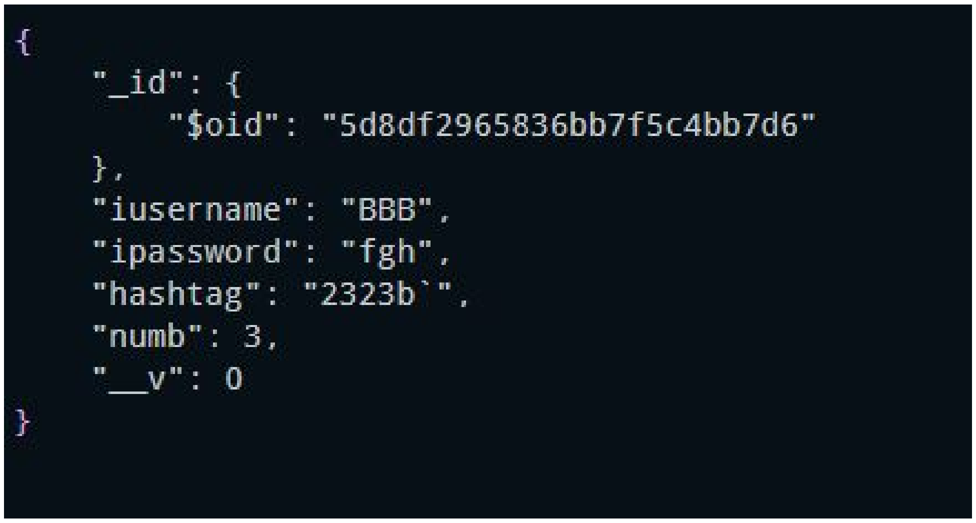
The second collection is **Folls** Collection which stores all the entries of the Instagram Follow attacks. Each time the user fills the follow form and hits send the entry is done in this cloud database and the Puppeteer RESTapi made my me gets triggered and it starts the attack on Instagram with the given credentials.



```
{
    "_id": {
        "$oid": "5d8dd64d754d6d80243206ec"
    },
    "iusername": "User1sdad",
    "ipassword": "sdasda",
    "numb": 5,
    "__v": 0
}
```

These are the values of the fields of the **Foll** database. The api uses the iusername and ipassword to login to the instagram account and then it accpets the numb number of request in that instagram account if available. If the request's are not available it shows a message of no more follow requests are available to follow.

The third collection is **Reqt** Collection which stores all the entries of the Instagram Follow attacks. Each time the user fills the Request form and hits send the entry is done in this cloud database and the Puppeteer RESTapi made my me gets triggered and it starts the attack on Instagram with the given credentials.



```
{
    "_id": {
        "$oid": "5d8df2965836bb7f5c4bb7d6"
    },
    "iusername": "BBB",
    "ipassword": "fgh",
    "hashtag": "2323b`",
    "numb": 3,
    "__v": 0
}
```

These are the values of the fields of the **Reqt** database. The api uses the iusername and ipassword to login to the instagram account and then it opens the posts of that specific hashtag and likes the top “numb” number of posts of that specific hashtag. This will increase the vibrancy of the instagram account which will in turn result in more popularity and thus more people will take notice and follow the instagram profile.

Various Schema Models used:

User Schema:

```
var mongoose=require("mongoose");
var
passportLocalMongoose=require("passport-local-mongoose");

var UserSchema=new mongoose.Schema({
    email: String,
    password: String
});

UserSchema.plugin(passportLocalMongoose);

module.exports=mongoose.model("User",UserSchema);
```

Foll Schema:

```
var mongoose = require('mongoose');
var passportLocalMongoose =
require('passport-local-mongoose');

var FollSchema = new mongoose.Schema({
  iusername: String,
  ipassword: String,
  numb: Number
});

FollSchema.plugin(passportLocalMongoose);

module.exports = mongoose.model('Foll', FollSchema);
```

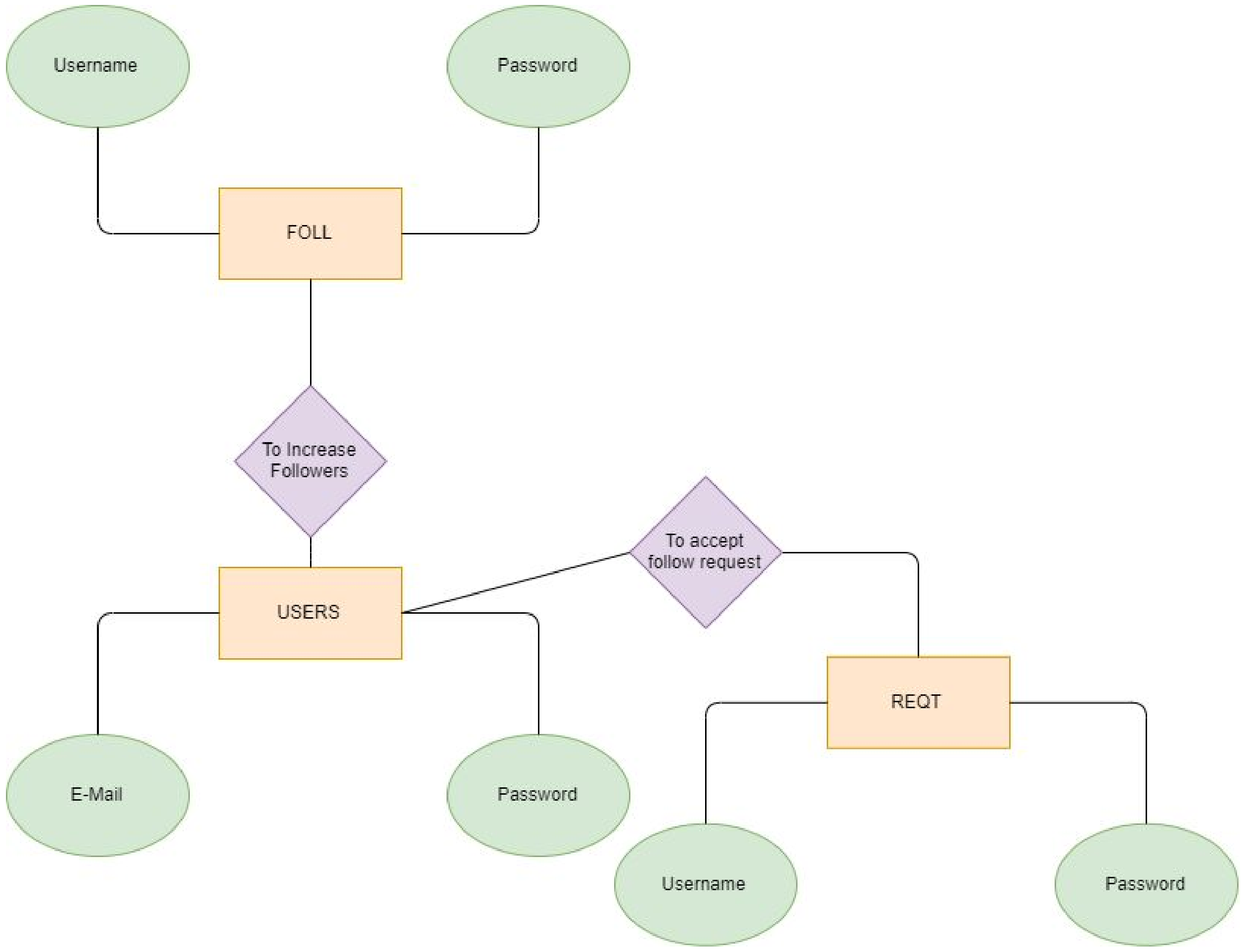
Reqt Schema:

```
var mongoose = require('mongoose');
var passportLocalMongoose =
require('passport-local-mongoose');

var ReqtSchema = new mongoose.Schema({
  iusername: { type: String, sparse: true, unique: false
},
  ipassword: String,
  hashtag: String,
  numb: Number
});

ReqtSchema.plugin(passportLocalMongoose);

module.exports = mongoose.model('Reqt', ReqtSchema);
```

ER Diagram:

Since the database used in this project is NoSQL database and therefore logically concept of Entity-relationship Diagram can't be stretched to the database used in this project, but if we map the presented NoSQL database to a relational database then the following Entity-Relationship diagram can be sketched for the proposed database design.

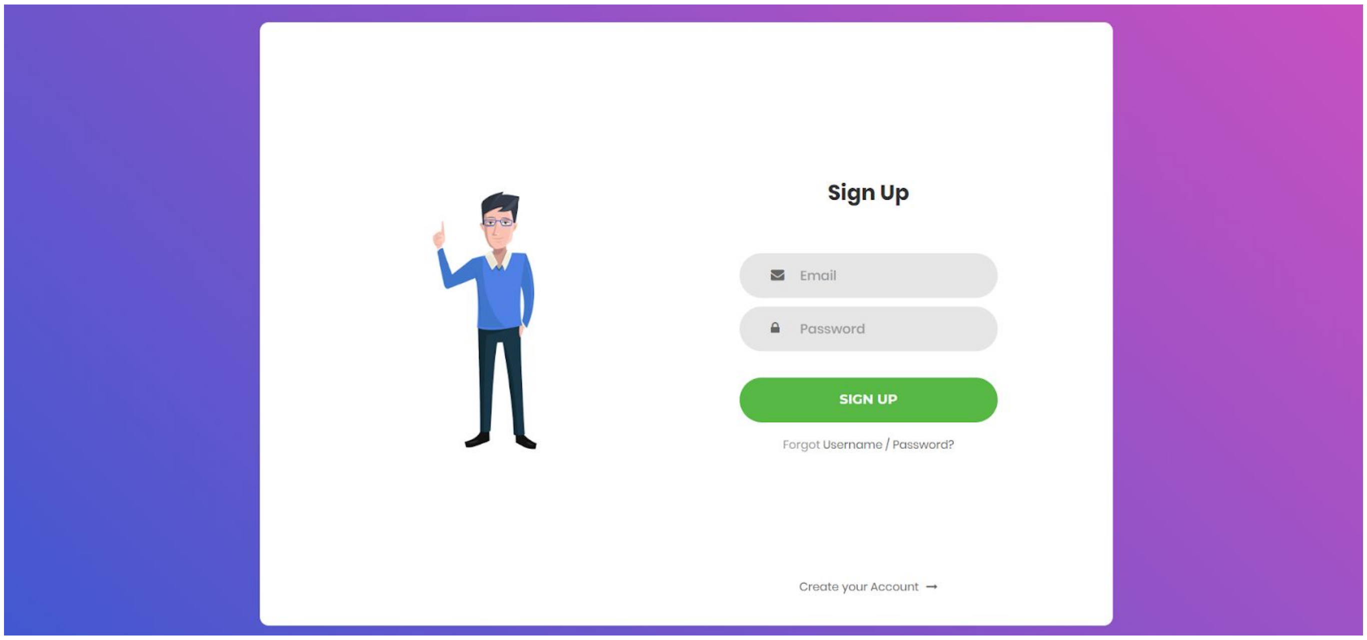
Also since all the three database collections are practically independent to each other so there will be three separate small ER diagrams which won't be linking to each other.

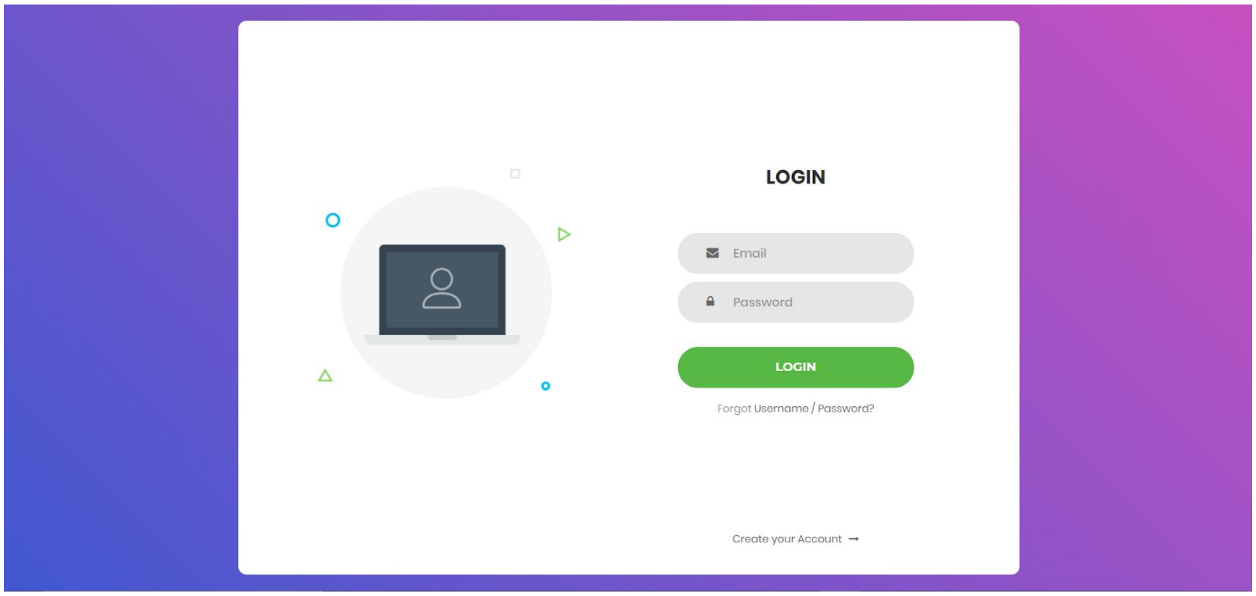


As in the ER diagram you can see that there are three different tables of Users, Foll and Reqt. All of then are connected to each other from relationships of what they are doing.

DB Integration with Front-end:

There are 4 pages with a form in each page. Each form has a specific purpose.
1st form is for Sign Up
2nd form is for Login
3rd form is for Follower attack on Instagram
4th form is for Request acceptor attack





LOGIN

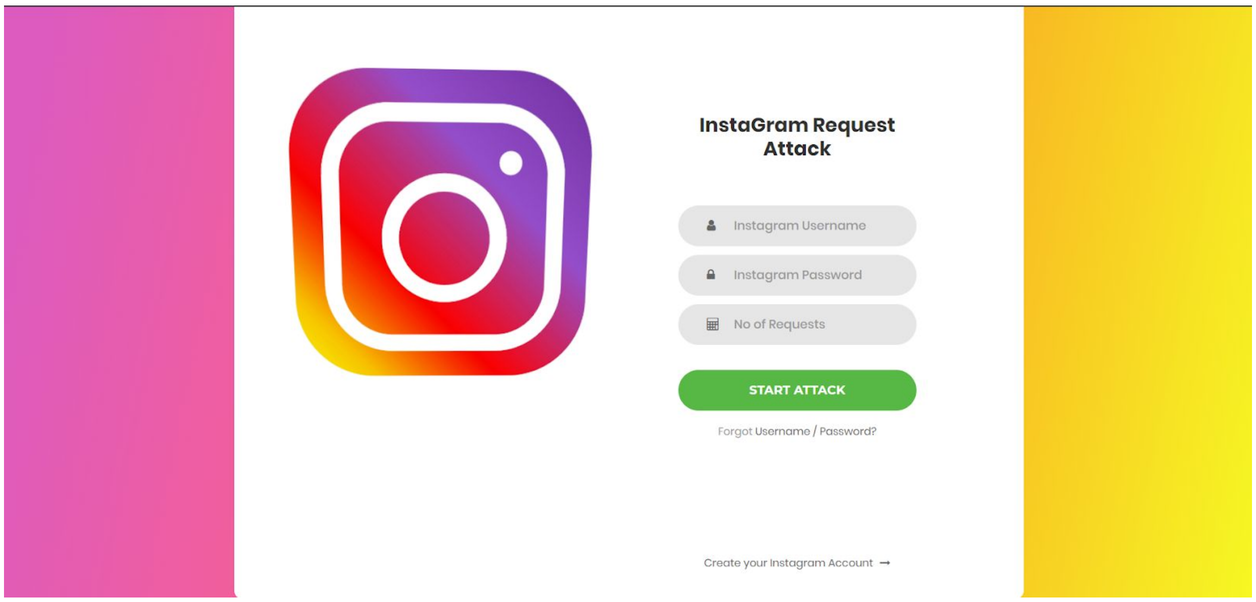
Email

Password

LOGIN

Forgot Username / Password?

Create your Account →



InstaGram Request Attack

Instagram Username

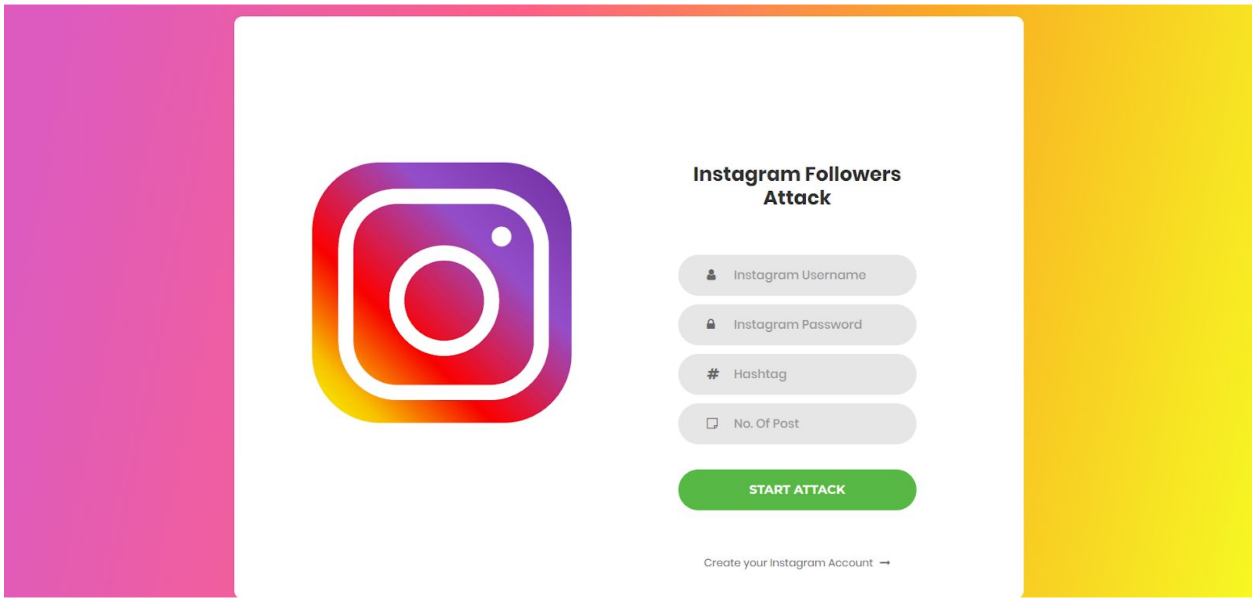
Instagram Password

No of Requests

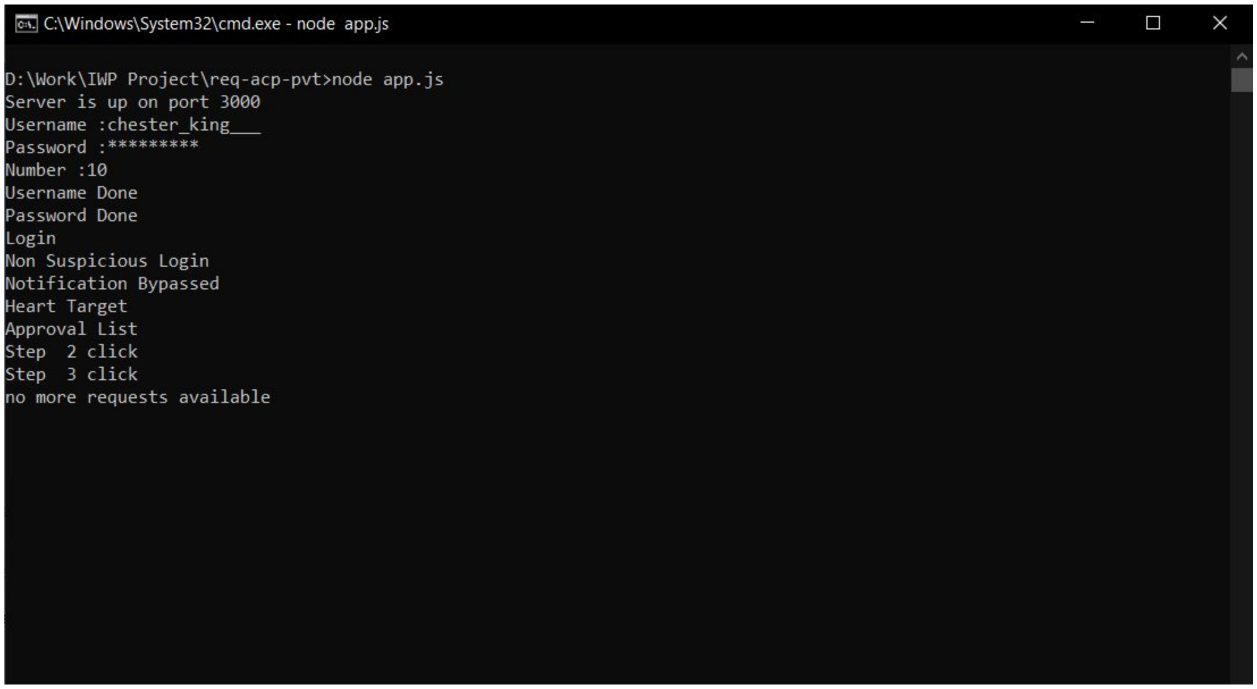
START ATTACK

Forgot Username / Password?

Create your Instagram Account →

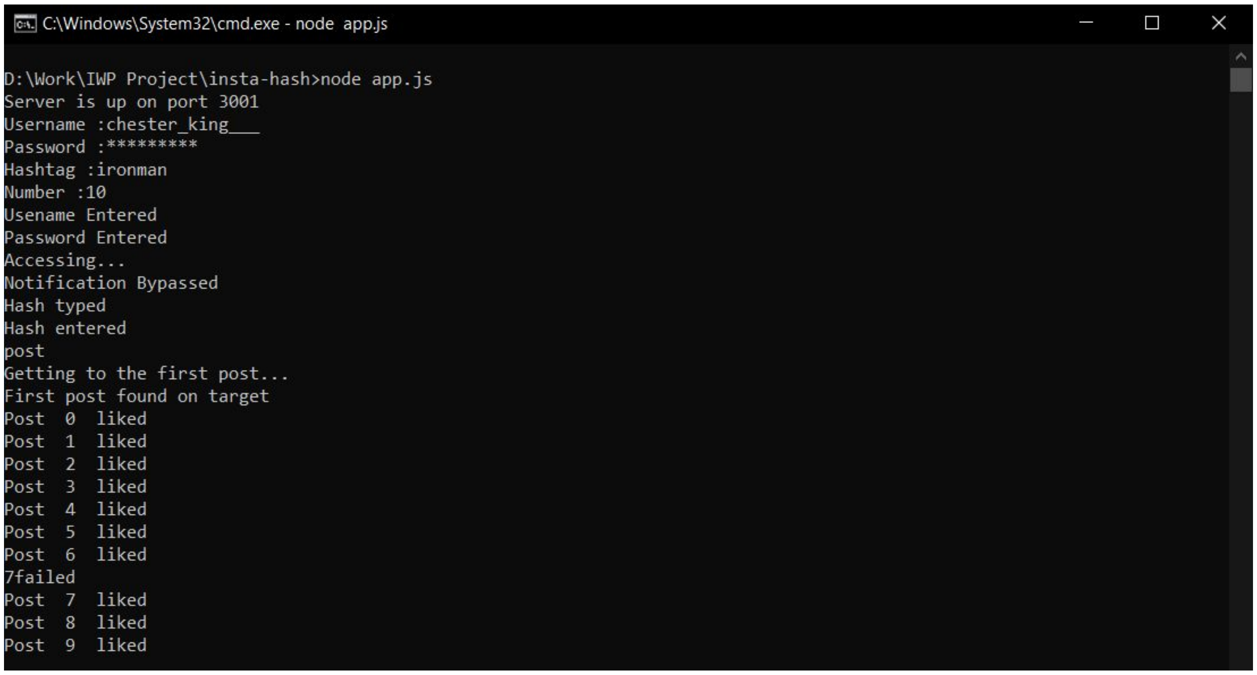


The pages are responsive and has special effects in them. For instance if you hover over the Instagram or other logos in the picture then you can see them skewing over the areas where you hover.



```
D:\Work\IWP Project\req-acp-pvt>node app.js
Server is up on port 3000
Username :chester_king___
Password :*********
Number :10
Username Done
Password Done
Login
Non Suspicious Login
Notification Bypassed
Heart Target
Approval List
Step  2 click
Step  3 click
no more requests available
```

Here you can see the request api running in action as the entire process is automated and headless for request it works silently in the background without opening additional windows for the operation. Everything is done in command line itself.



```
D:\Work\IWP Project\insta-hash>node app.js
Server is up on port 3001
Username :chester_king___
Password :*********
Hashtag :ironman
Number :10
Usename Entered
Password Entered
Accessing...
Notification Bypassed
Hash typed
Hash entered
post
Getting to the first post...
First post found on target
Post  0  liked
Post  1  liked
Post  2  liked
Post  3  liked
Post  4  liked
Post  5  liked
Post  6  liked
7failed
Post  7  liked
Post  8  liked
Post  9  liked
```

This is the API made for the instagram follower increase. It fetches the first post of an hashtag and likes the number of post given in as request.

Instructions to execute:

There are 4 folders.

1. InstaForms only contains frontend pages which will show you pure HTML Code with all the other JS and CSS files.
2. Insta-hash has the follower request increase API coded in NodeJS. You can open the Command Line at insta-hash and then use command: ***npm install*** to install all the dependencies. Then hit with post man at port 3001.

   ```
   localhost:3001?username=chester_king___&password=<instaPassword>
   &hashtag=ironman&number=10
   ```

3. Req-acp-pvt accepts specific number of follow request. You can open the Command Line at req-acp-pvt and then use command: ***npm install*** to install all the dependencies. Then hit with post man at port 3000.

   ```
   localhost:3000?username=chester_king___&password=Sertoli.01&number=10
   ```

4. Insta-Attack has the main application with the frontend integrated with the Database.
   Do npm install at this folder and then
   You can http://localhost:3000/reqAttack to fill req form and
   You can http://localhost:3000/folAttack to fill fol form.
   It will save the details to my mlab cloud Database.