PASSWORD SECURITY

KLE TECHNOLOGICAL UNIVERSITY

Kiran Akadas	92
Manas Kumar	98
Manas Vyas	99

May - 2019

TABLE OF CONTENTS

INTRODUCTION

In real life applications with User authentication functionality, it is not practical to store user password as the original string in the database but it is good practice to hash the password and then store them into the database.

Hashed passwords are not unique to themselves due to the deterministic nature of hash function: when given the same input, the same output is always produced.

A rainbow table can make the exploitation of unsalted passwords easier. A rainbow table is essentially *a* pre-computed database of hashes. Dictionaries and random strings are run through a selected hash function and the input/hash mapping is stored in a table. The attacker can then simply do a password reverse lookup by using the hashes from a stolen password database.

To mitigate the damage that a rainbow table or a dictionary attack could do, we salt the passwords. A salt is a fixed-length cryptographically-strong random value that is added to the input of hash functions to create unique hashes for every input, regardless of the input not being unique. A salt makes a hash function look non-deterministic, which is good as we don't want to reveal password duplications through our hashing.

Different users, same password. Different salts, different hashes

"Hashing salts are speed bumps in an attacker's road to breaching your data. It does not matter if they are visible and unencrypted, what matters is that they are in place."

A salt is added to the hashing process to force their uniqueness, increase their complexity without increasing user requirements, and to mitigate password attacks like rainbow tables

TOOLS AND TECHNOLOGIES USED

- Angular JS

  The client side in the developed application runs on Angular

- NodeJS

  The Server of our application is based on NodeJs

- Express

  Express acts as the web application framework and is used in development of the application

- MongoDB

  All the signup information is stored in the Mongo Database.

ERRORS HANDLED

- Duplicate Username (Signup)

  The application alerts user in case of duplicate username

- Non-Existing Username while Login

  The application alerts user about the invalid entry

- Invalid Form Entries

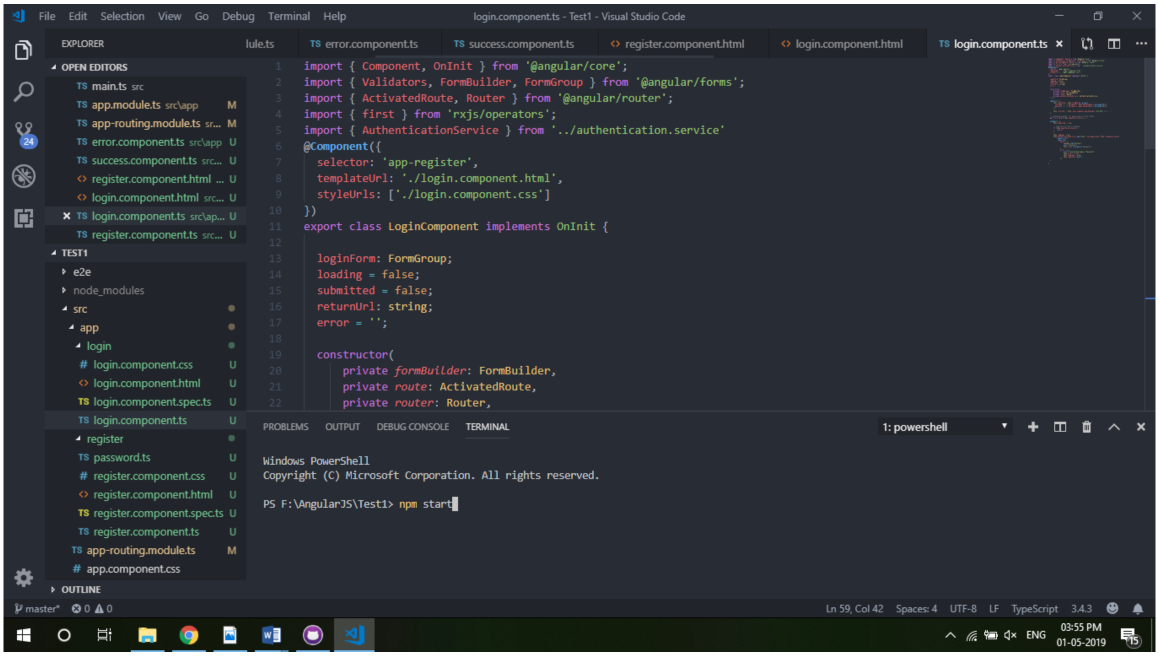
  The application alerts user during the entry

GETTING STARTED – RUNNING THE APPLICATION

- Ensure that all of the following softwares are installed on your server machine
  - Visual Studio Code
  - Robo 3T (Optional)
  - NodeJS
    - Npm
    - Mongo
    - Express
    - Body-parser
    - Crypto

Step 1:

Open Visual Studio Code and open the the angular project folder



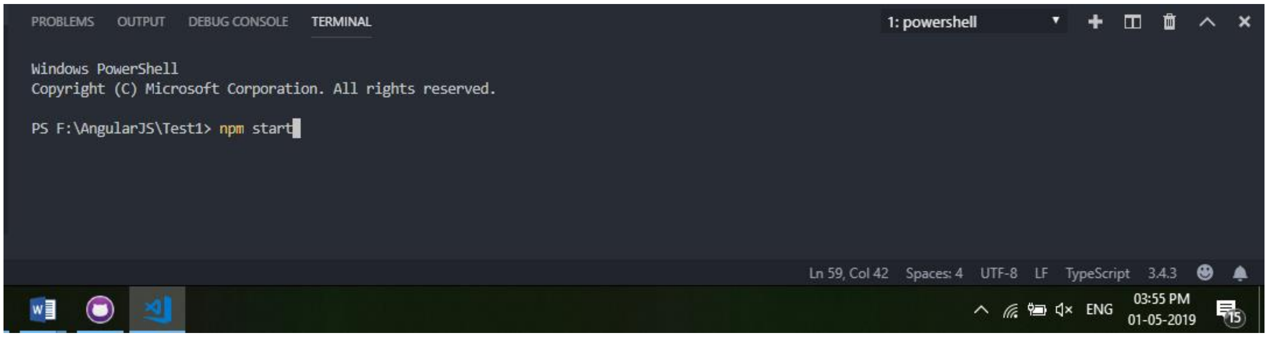
```
import { Component, OnInit } from '@angular/core';
import { Validators, FormBuilder, FormGroup } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { first } from 'rxjs/operators';
import { AuthenticationService } from '../authentication.service'
@Component({
  selector: 'app-register',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  loginForm: FormGroup;
  loading = false;
  submitted = false;
  returnUrl: string;
  error = '';

  constructor(
      private formBuilder: FormBuilder,
      private route: ActivatedRoute,
      private router: Router,
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS F:\AngularJS\Test1> npm start
```

Step 2:

Open the terminal using the 'View' tab and run the command:

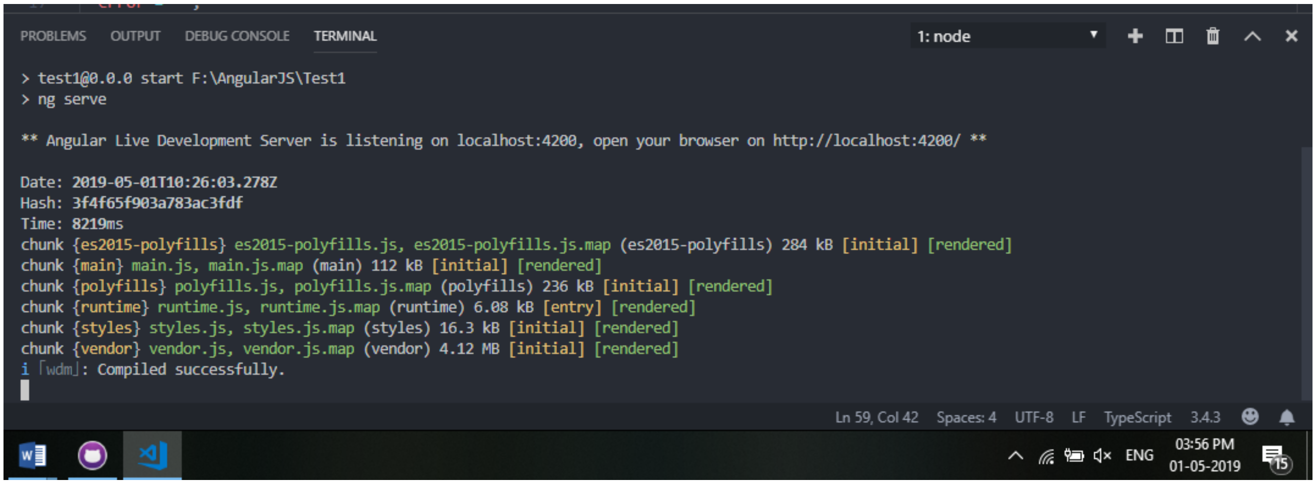
"npm start"



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS F:\AngularJS\Test1> npm start
```

Step 2:

Wait for the project to initialize and the server to host the application. The following message should appear on the terminal

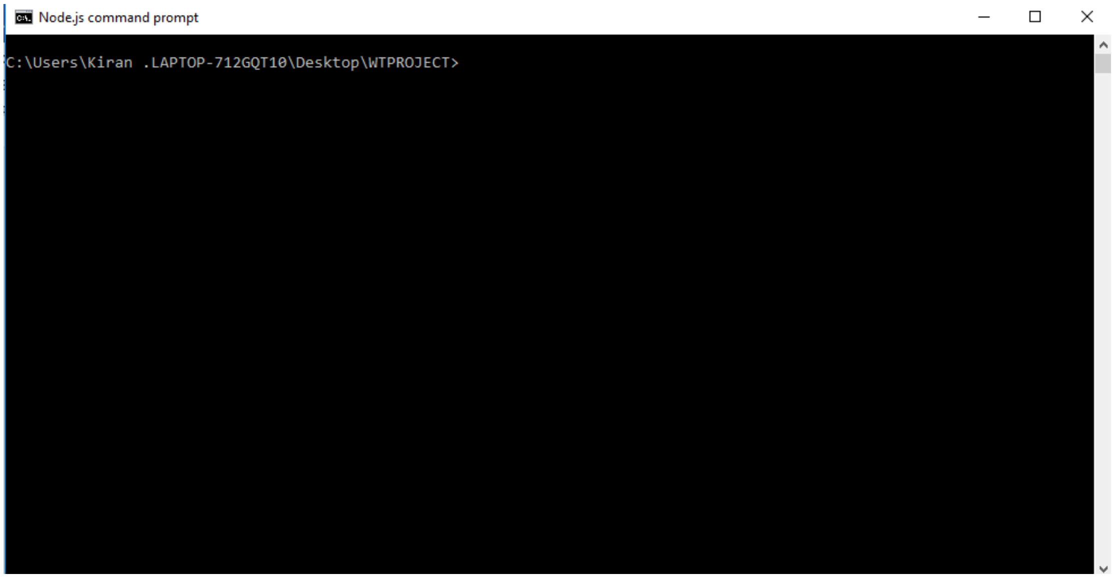


```
> test1@0.0.0 start F:\AngularJS\Test1
> ng serve

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2019-05-01T10:26:03.278Z
Hash: 3f4f65f903a783ac3fdf
Time: 8219ms
chunk {es2015-polyfills} es2015-polyfills.js, es2015-polyfills.js.map (es2015-polyfills) 284 kB [initial] [rendered]
chunk {main} main.js, main.js.map (main) 112 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 236 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.3 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.12 MB [initial] [rendered]
i ｢wdm｣: Compiled successfully.
```

This indicates successful hosting of your client-side application

Step 3:

Open Node.Js Command Prompt and Change your directory to the directory with the node js project.

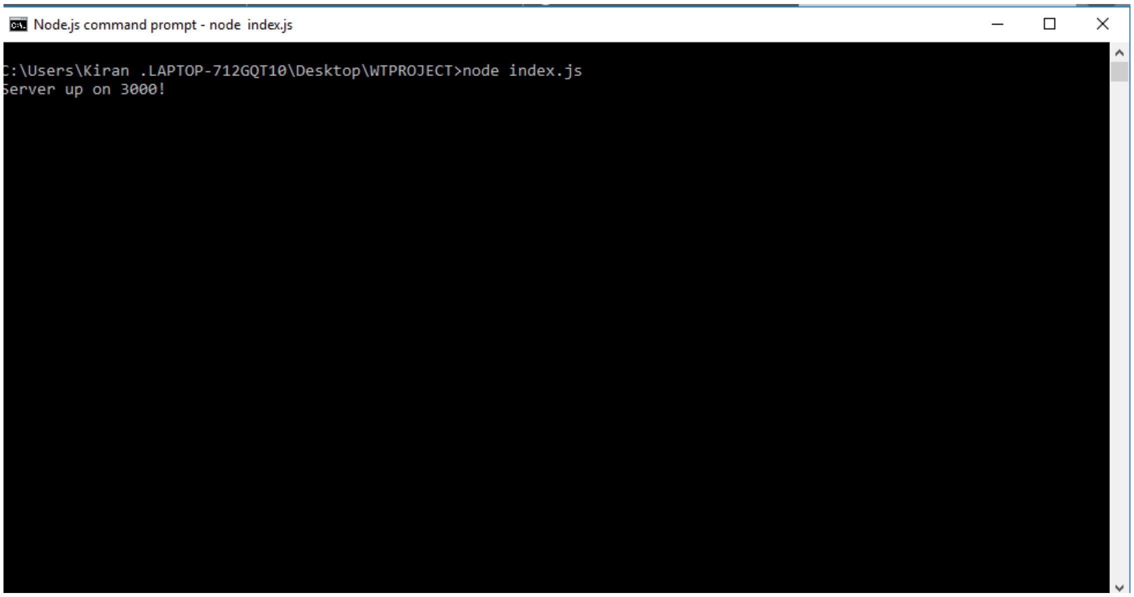


```
C:\Users\Kiran .LAPTOP-712GQT10\Desktop\WTPROJECT>
```

Step 4:

Type the following command which launches the server application:

node index.js



```
:\Users\Kiran .LAPTOP-712GQT10\Desktop\WTPROJECT>node index.js
Server up on 3000!
```

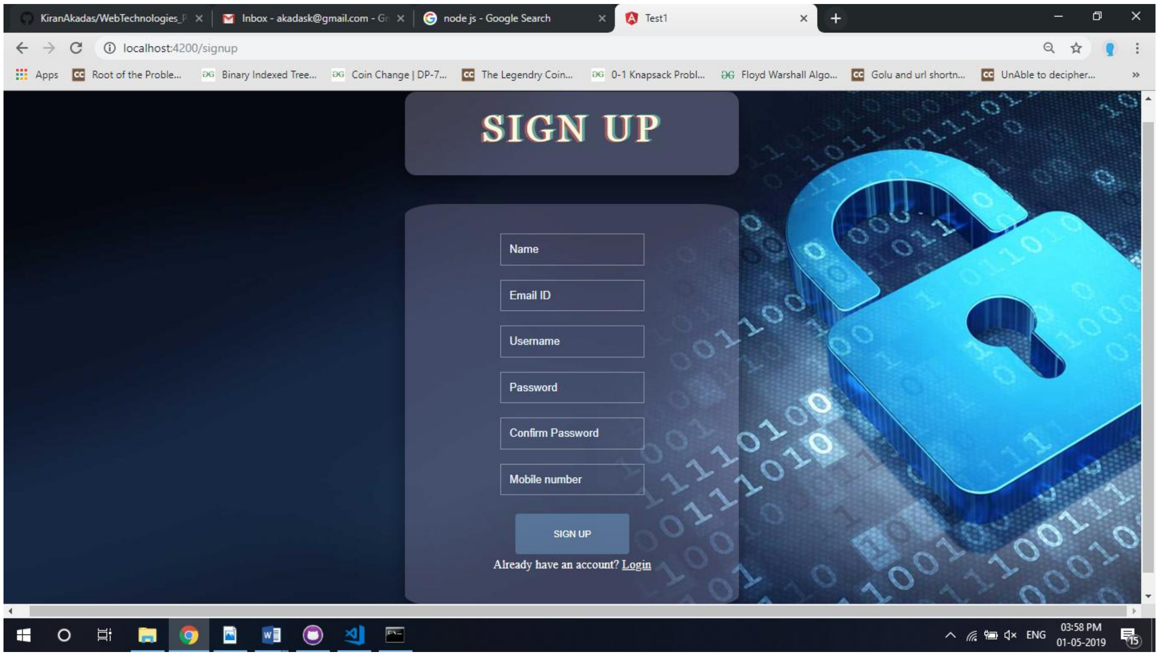
The above message appears on the screen if your application is hosted successfully

Now, We are ready to roll!

Step 5:

Open your web browser and open the url:

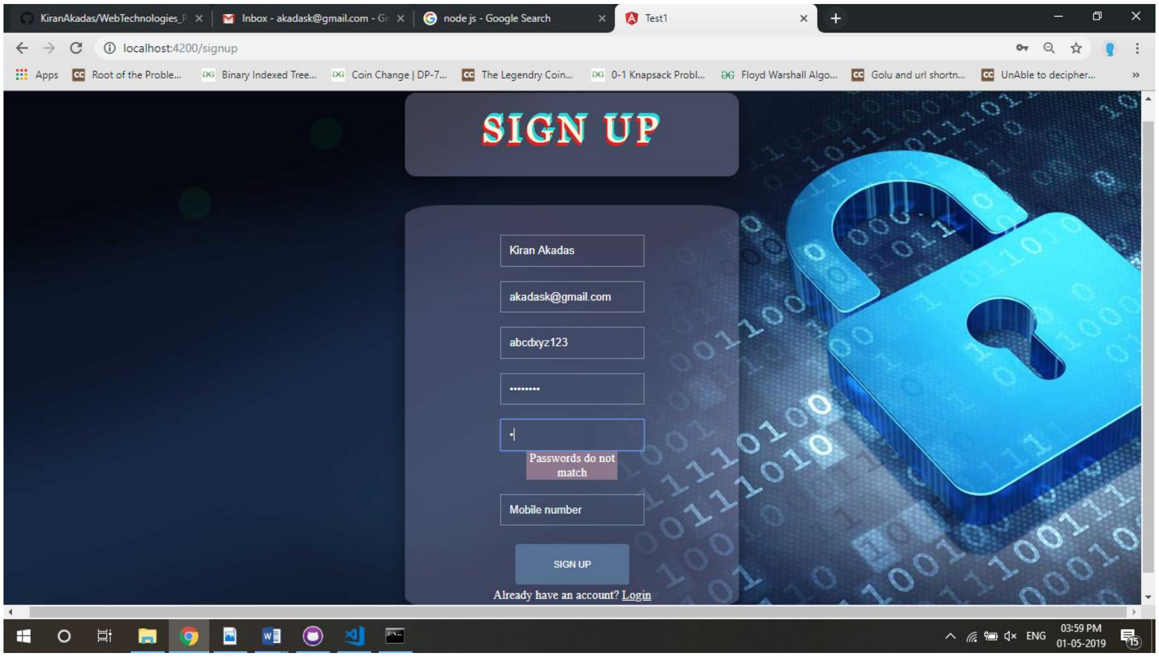
http://localhost:4200/signup



The above webpage opens up.

Step 6:

Enter all your credentials to sign up. Ensure that you’ve maintained all the requirements for the fields:



KiranAkadas/WebTechnologies_P
Inbox - akadask@gmail.com - Gr
node js - Google Search
Test1
localhost:4200/signup
SIGN UP
SIGN UP
Already have an account? Login

Kiran Akadas

akadask@gmail.com

abcdxyz123

••••••••

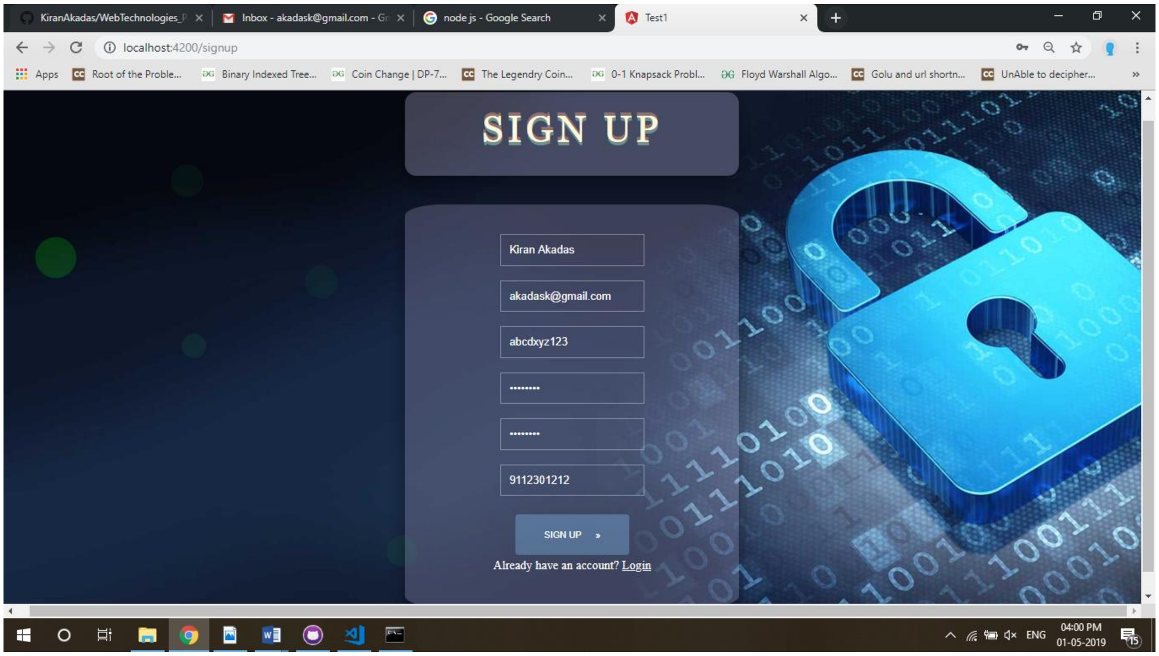
•

Passwords do not match

Mobile number

Step 7:

After Entering all the details, click on the SignUp button.



KiranAkadas/WebTechnologies_P
Inbox - akadask@gmail.com - Gr
node js - Google Search
Test1
localhost:4200/signup
SIGN UP
SIGN UP
Already have an account? Login

Kiran Akadas

akadask@gmail.com

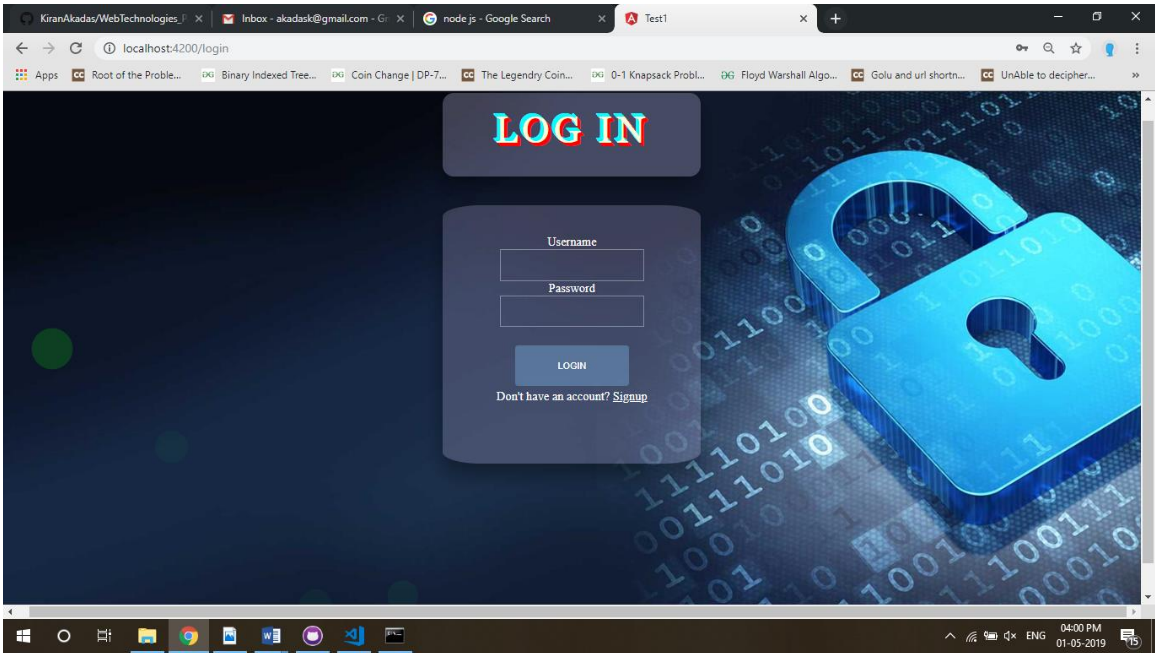
abcdxyz123

••••••••

••••••••

9112301212

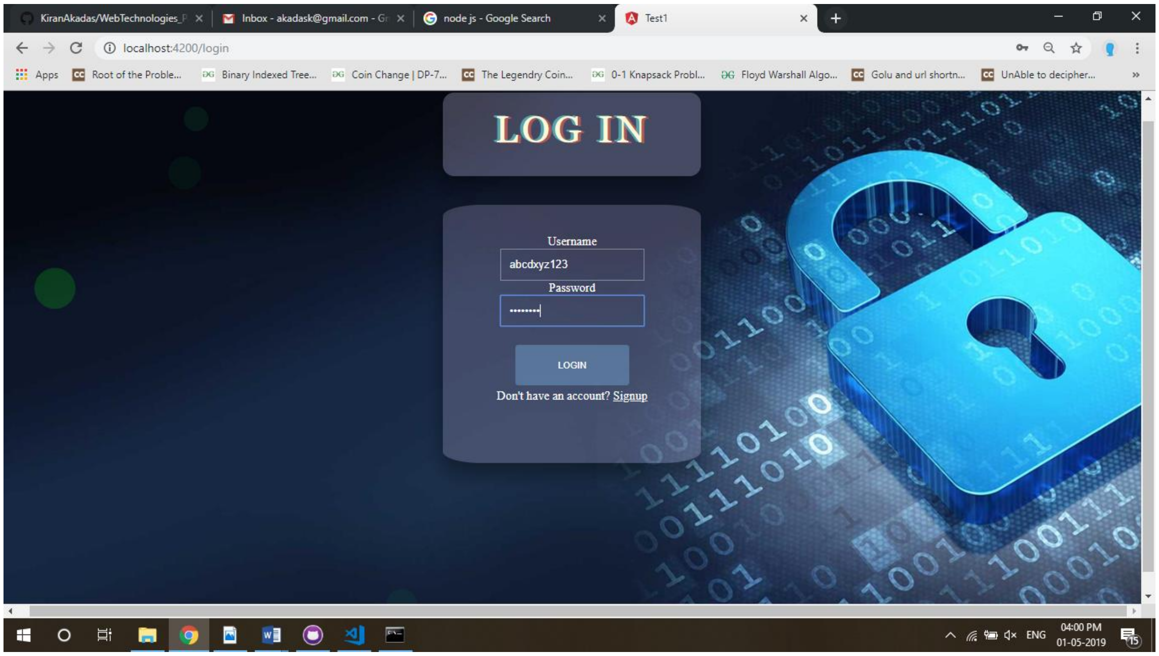
On Successful Registration, you’ll be redirected to the log-in page.



Step 8:

Now, we try logging in to our application

Enter the same username and password used for signup.

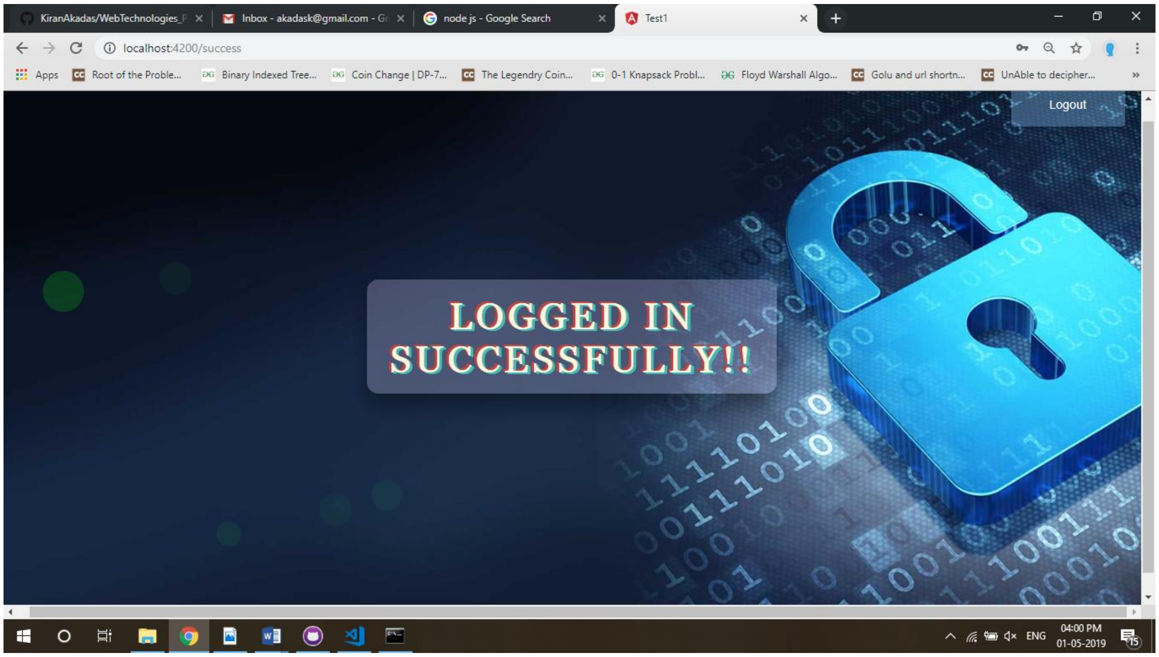


Username: abcdxyz123

Password: ••••••••

Step 9:

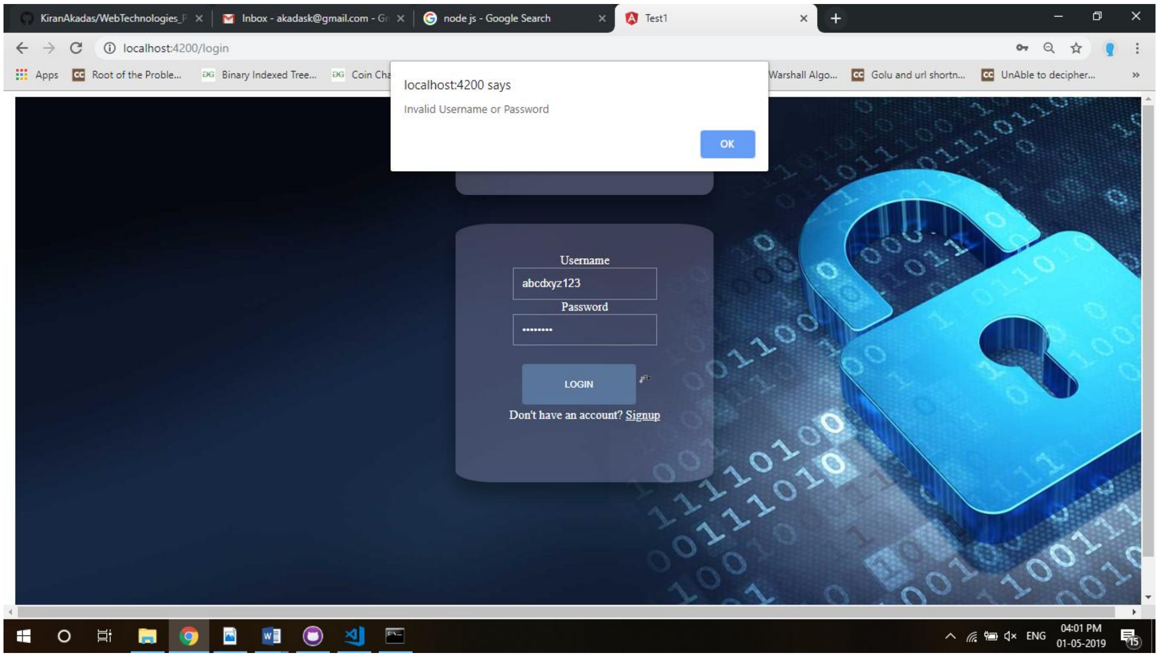
Click On the Login Button, and you'll be directed to a success page.


localhost:4200/success

LOGGED IN SUCCESSFULLY!!

Step 10:

Try entering a wrong password and you will be alerted with an error message as shown below.


localhost:4200/login

localhost:4200 says

Invalid Username or Password

Username: abcdxyz123

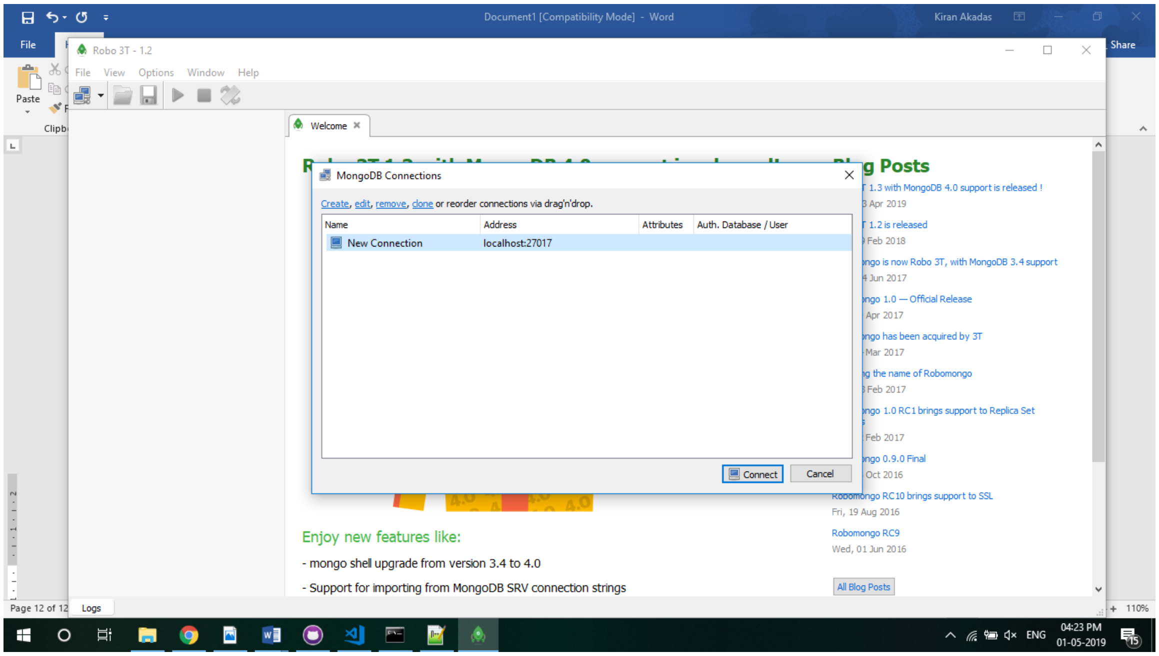
Password: ••••••••

Don't have an account? Signup

VERIFICATION

Step 1:

Open Robo 3T (GUI for Mongo DB)



Create, edit, remove, clone or reorder connections via drag'n'drop.

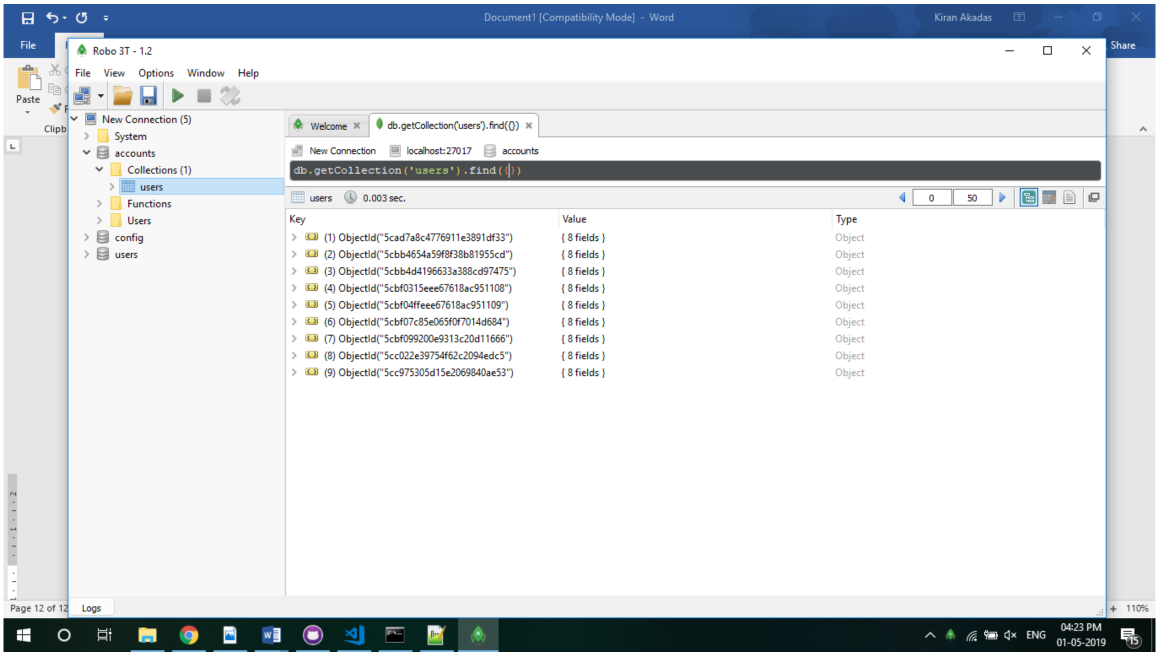
Name	Address	Attributes	Auth. Database / User
New Connection	localhost:27017		

Enjoy new features like:

- mongo shell upgrade from version 3.4 to 4.0
- Support for importing from MongoDB SRV connection strings

Step 2:

Connect to localhost:27017

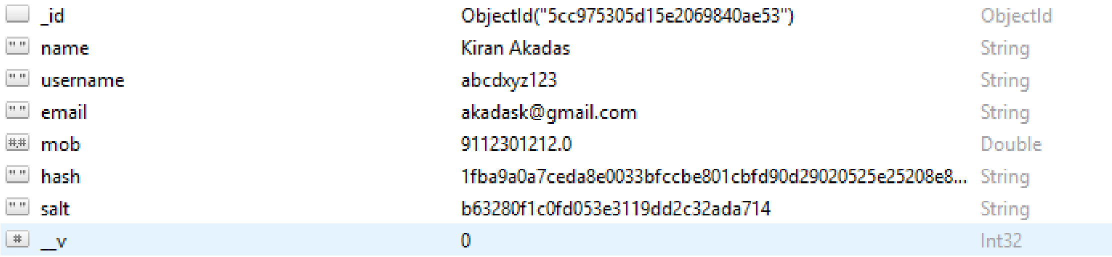


```
db.getCollection('users').find({})
```

Key	Value	Type
(1) ObjectId("5cad7a8c4776911e3891df33")	{ 8 fields }	Object
(2) ObjectId("5cbb4654a59f8f38b81955cd")	{ 8 fields }	Object
(3) ObjectId("5cbb4d4196633a388cd97475")	{ 8 fields }	Object
(4) ObjectId("5cbf0315eee67618ac951108")	{ 8 fields }	Object
(5) ObjectId("5cbf04ffeee67618ac951109")	{ 8 fields }	Object
(6) ObjectId("5cbf07c85e065f0f7014d684")	{ 8 fields }	Object
(7) ObjectId("5cbf099200e9313c20d11666")	{ 8 fields }	Object
(8) ObjectId("5cc022e39754f62c2094edc5")	{ 8 fields }	Object
(9) ObjectId("5cc975305d15e2069840ae53")	{ 8 fields }	Object

Now, select System in the left panel, go to accounts -> Collections -> Users

On clicking the users table, all the entries made till now are displayed. Click on the latest entry and you can view the information you entered.



_id	ObjectId("5cc975305d15e2069840ae53")	ObjectId
name	Kiran Akadas	String
username	abcdxyz123	String
email	akadask@gmail.com	String
mob	9112301212.0	Double
hash	1fba9a0a7ceda8e0033bfccbe801cbfd90d29020525e25208e8...	String
salt	b63280f1c0fd053e3119dd2c32ada714	String
__v	0	Int32

We see that the passwords entered are never stored in the server and are stored as hashes and salt.

During Login, the salt is obtained for a username, added to the password and the resulting string is hashed giving a new hashcode which is compared with the above hash present in the table.