**Sign In and Signup with Encryption**

**Requirement2:** As in requirement 1 anyone can delete other meows. So let’s implement Only Letting You Delete Your Own. To achieve this, below recipes need to be follow.

Recipe5: Front-End Routing for Making a Signup Page + Storing Users in DB.

Recipe6: Encrypt Passwords with bcryptjs and Make a Login Route.

Recipe7: Login and Save a JSON Web Token in a Cookie.

Recipe8: Showing the Author of the Meows + Only Letting You Delete Your Own.

Let’s start with these recipes.

**Recipe5:** **Front-End Routing for Making a Signup Page + Storing Users in DB.**

Add below code inside SignupController to implement Signup functionality.

$scope.submitSignup=function(){

var newUser= {

username: $scope.username,

password: $scope.password

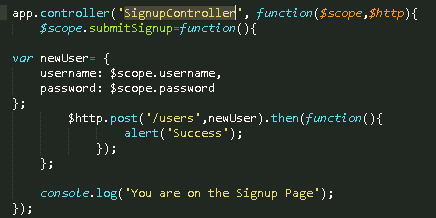
}; $http.post('/users',newUser).then(function(){

alert('Success');

});

};

console.log('You are on the Signup Page');



In server.js file add user model in mongoose.

var User=mongoose.model('User', {

username: String, password: String });

Now add below post method to save user in database.

app.post('/users', function(req, res, next){

var newUser= new User({

username: req.body.username,

password: req.body.password

});

newUser.save( function(err){

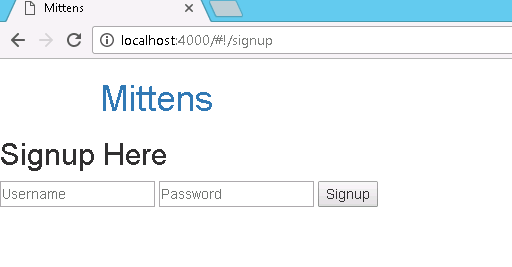
return res.send();

});

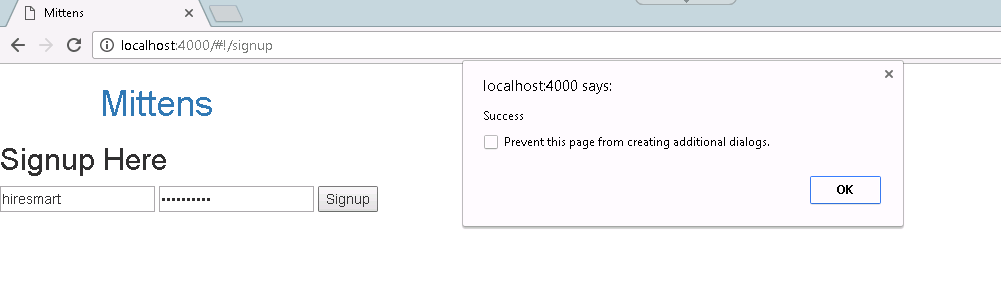
}) ;

Save it and restart your server.

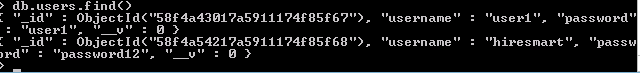
Open up the browser and click on Signup button.



Enter username and password and click on Signup.



Open your database and check users in it.

Hurray user is added in database.

**Recipe6:** **Encrypt Passwords with bcryptjs and Make a Login Route.**

As you can see in db password is visible and any one can read it. So we use the hash() function to encrypt passwords. Add bcrypt variable in server.js file

var bcrypt = require('bcryptjs');

Now use the hash() function to encrypt passwords. And change user POST method as below.

app.post('/users', function(req, res, next){

bcrypt.genSalt(10, function(err,salt){

bcrypt.hash(req.body.password,salt,function(err,hash){

var newUser= new User({

username: req.body.username,

password: hash

});

newUser.save( function(err){

return res.send();

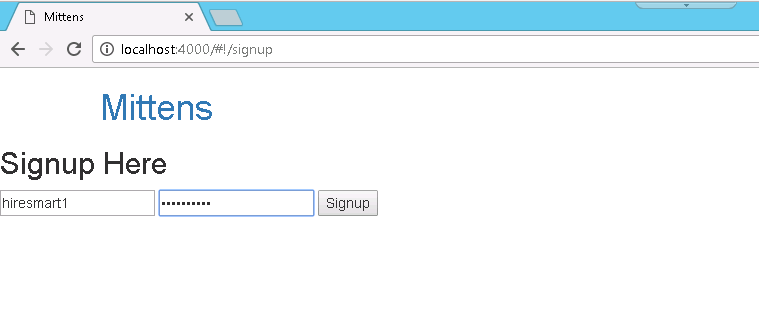
});

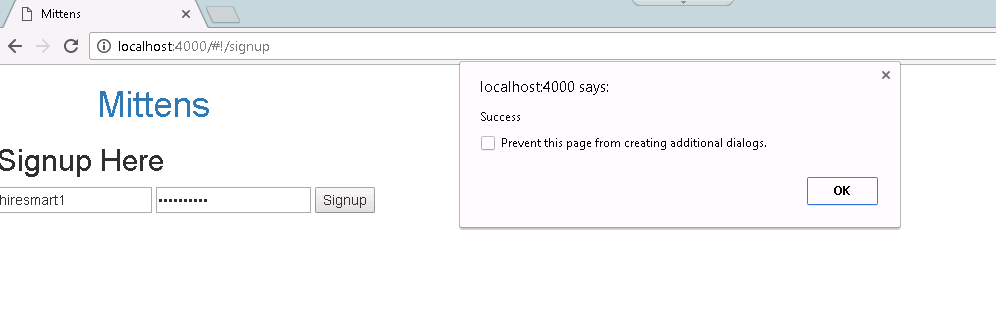
});

}) ;

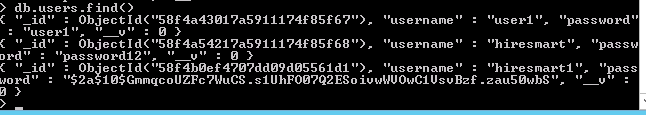
});

Restart the server and add a new user



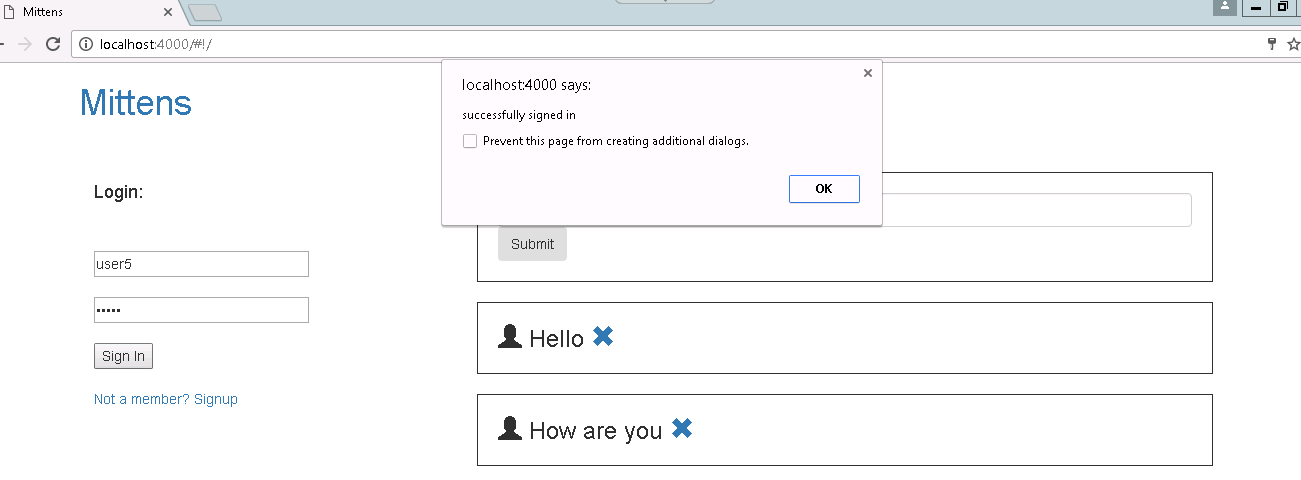
Click on Signup button.

Open your database and check users in it.



User is added in database with encrypted password.

Now try to login with your credentials.



**Recipe7:** **Login and Save a JSON Web Token in a Cookie.**

Why token, if you signed in a website and closed your browser or restarted your computer, comeback to the website and you still want to be logged in. So for that you need to create a token on the server upon login and Send the token to the browser and save it in a cookie.

So let’s start with the first step that is Generate a token on the server upon login.

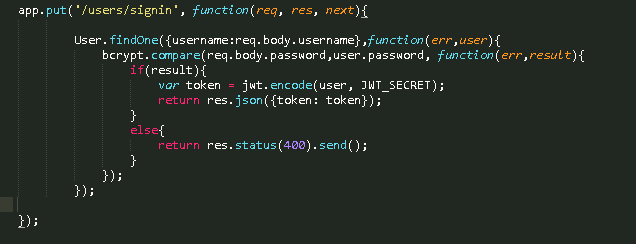
Add JWT variable in server.js file

var jwt = require('jwt-simple');

Now add Secret key for JWT. This is constant variable and define in server.js file.

var JWT\_SECRET='catsmeow';

Now change your Sign in function to generate token.



Token is generated now and we use this token at the time of adding new meow, deleting meow to authenticate user.

Now add one more dependency in your angular app i.e. cookies

var app = angular.module('mittens',['ngRoute','ngCookies']);

Add one more variable in home controller in index.html file.

C:\Users\462439\Desktop\22.PNG

Change sign in function as below in index.html file.

$scope.signin=function(){

$http.put('/users/signin',{ username: $scope.username,password: $scope.password})

.then(function(res){

$cookies.put('token',res.data.token);

$scope.currentUser=$scope.username;

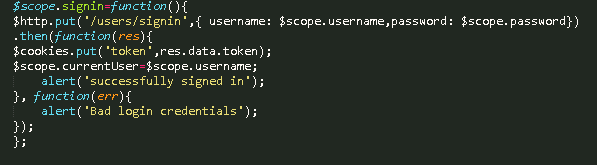
alert('successfully signed in');

}, function(err){

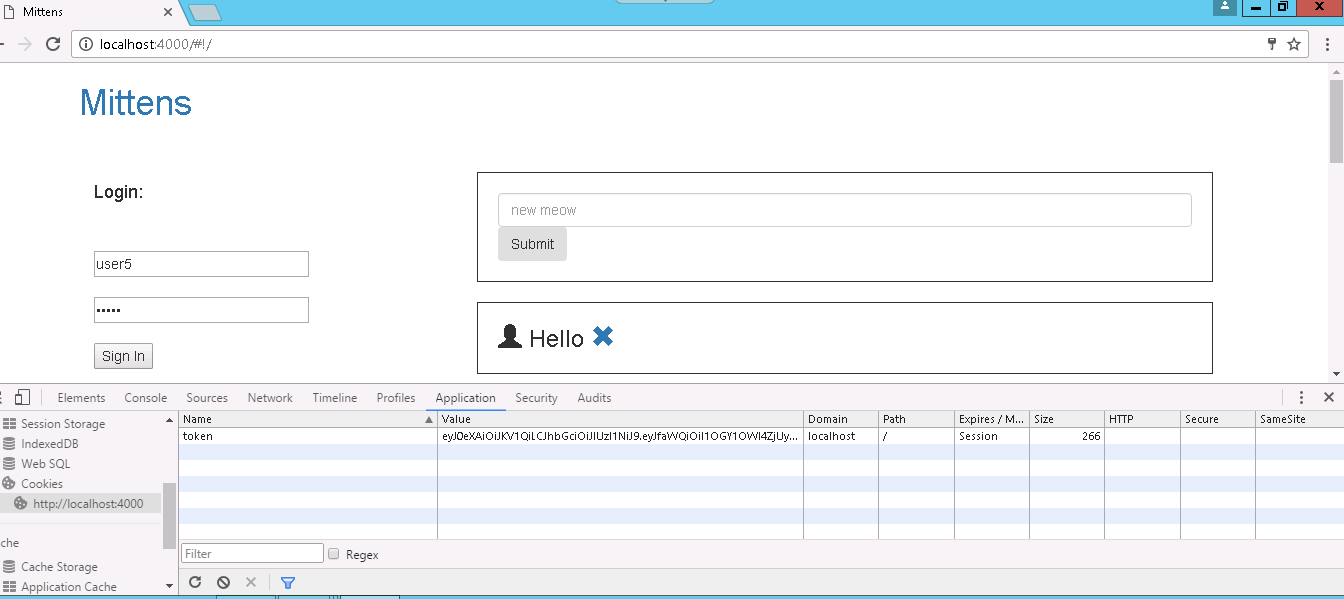
alert('Bad login credentials');

});

};



Login with credential and check in browser cookies



Add $rootScope in home controller. Here is $scope is to interact with your controller in your web page, $rootScope is similar but it’s like a step above $scope, It is used to share same stuff to all the controllers.

C:\Users\462439\Desktop\26.PNG

Change your Signin function again as below

$scope.signin=function(){

$http.put('/users/signin',{ username: $scope.username,password: $scope.password})

.then(function(res){

$cookies.put('token',res.data.token);

$cookies.put('currentUser',$scope.username);

$rootScope.token=res.data.token;

$rootScope.currentUser=$scope.username;

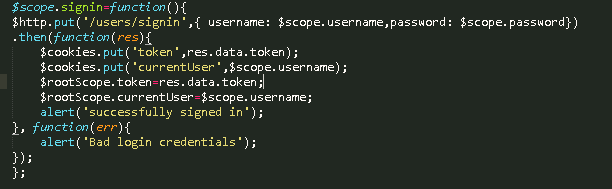
alert('successfully signed in');

}, function(err){

alert('Bad login credentials');

});

};



Here cookies are to save data for the browser and rootScope is for application.

Now add app.run after app. Config to check for cookies, when application start.

app.run(function($rootScope,$cookies){

if($cookies.get('token')&& $cookies.get('currentUser')){

$rootScope.token=$cookies.get('token');

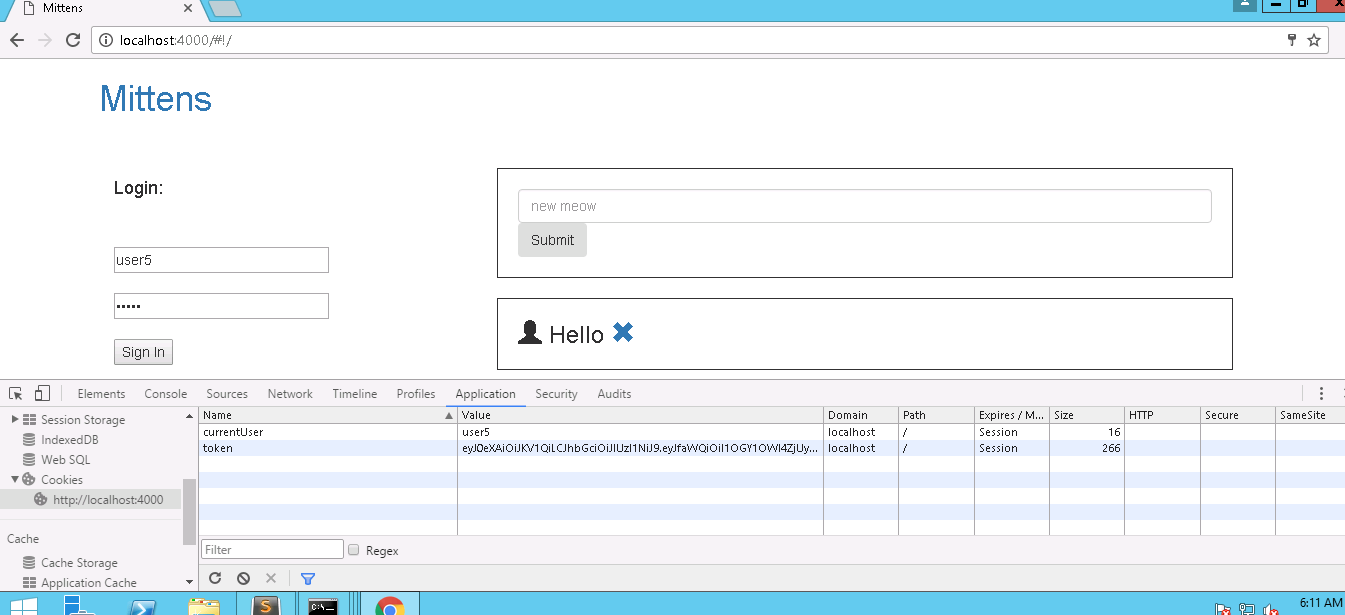
$rootScope.currentUser=$cookies.get('currentUser');

}

});



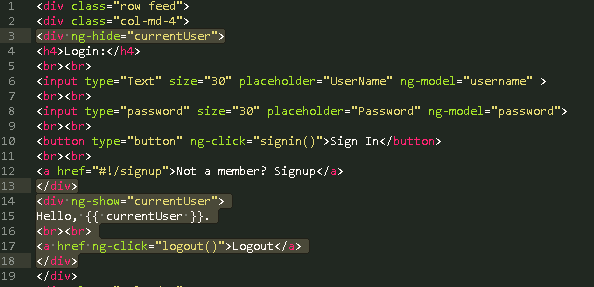
Login with credential and check in browser cookies



Here current user and token are saved in cookies.

Now we have to hide login box with “Hello User”

Open Home.html file and change it as below



Put ng-hide for current user on login box, and add new div with ng-show for current user with Logout button.

Implement logout() inside home controller in index.html file and clear all cookies and rootScope data.

$scope.logout=function(){

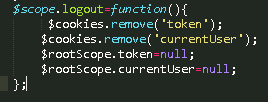
$cookies.remove('token');

$cookies.remove('currentUser');

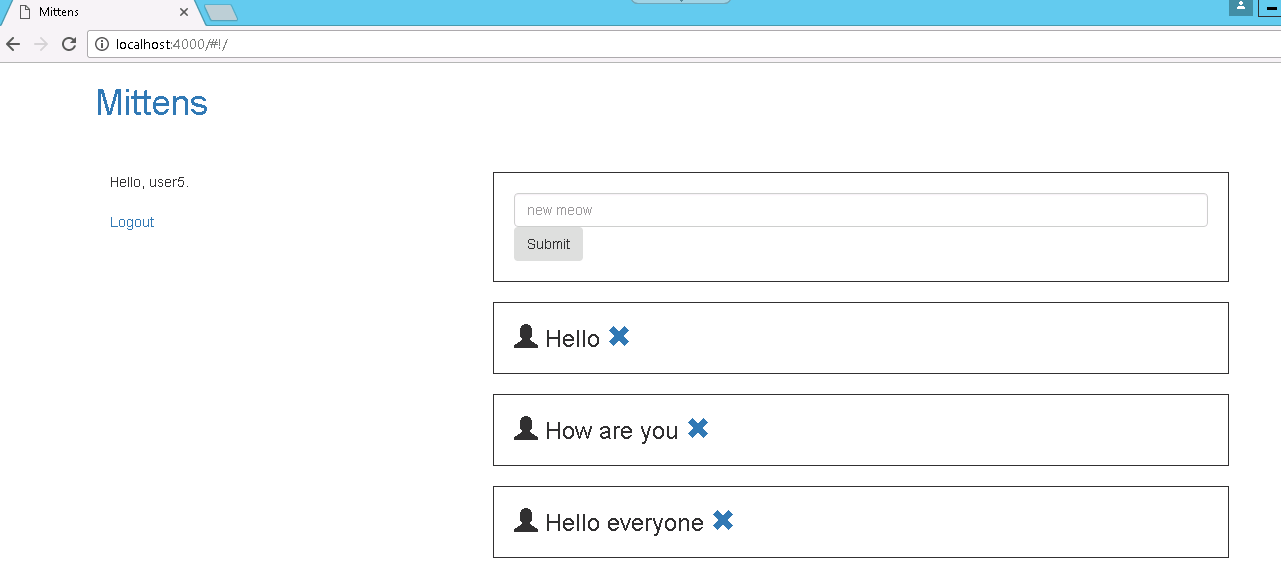
$rootScope.token=null;

$rootScope.currentUser=null;

};



Check in the browser

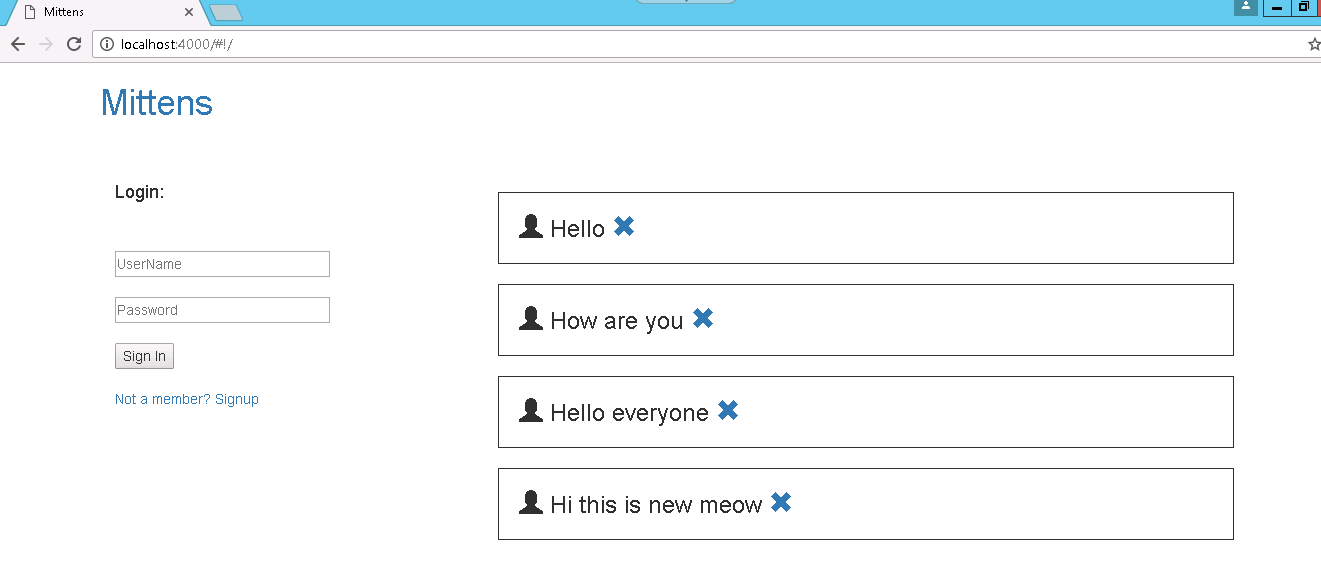


**Recipe8:** **Showing the Author of the Meows + Only Letting You Delete Your Own.**

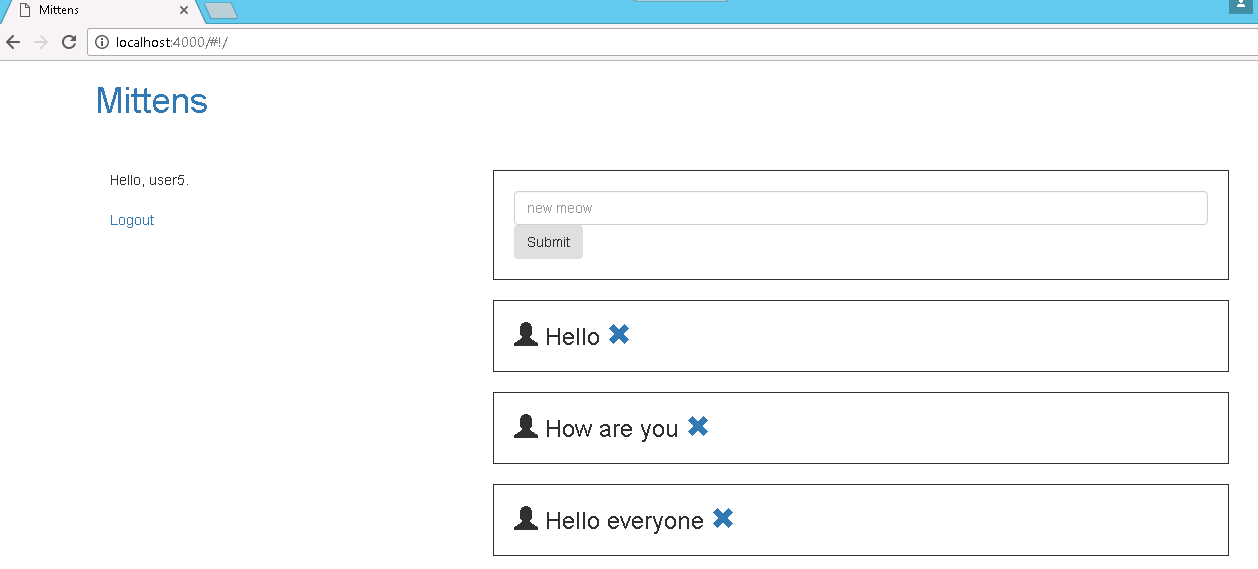
Now we have to show New Meow box for logged in user only. So for that change home.html file as below

C:\Users\462439\Desktop\32.PNG

Add ng-show for current user only. Let’s check in browser. It’s hide.



Logged in and check. Now it’s there.



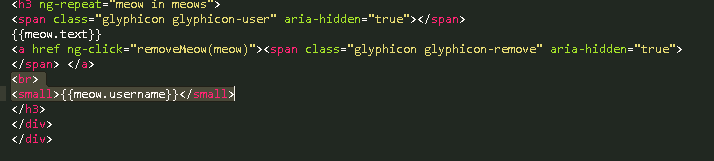
Here we can add new meow and also delete any meow. But there is a problem user can delete other user meow also. So to fix it

1) Put the JWT token we've been saving into the header of the POST meow request

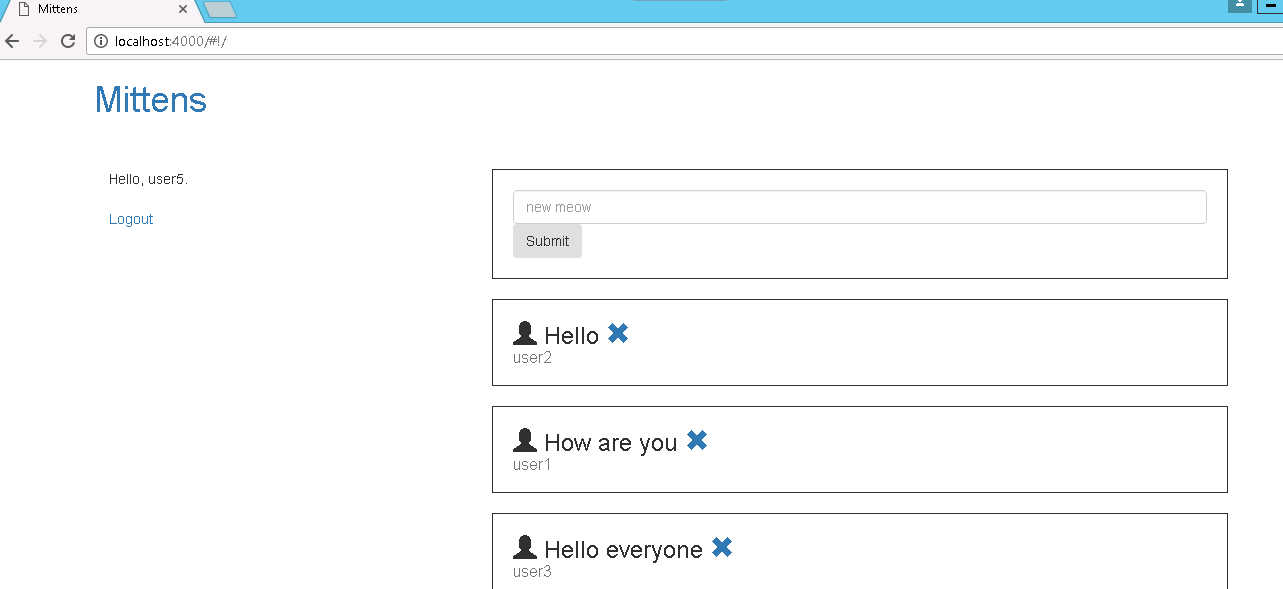
2) Take it and decode it on the server, and save the user's ID and username as part of the new meow object

3) Upon delete request, check to make sure the person deleting it owns that meow.

First put username below the meow, so change home.html as below



Check in browser

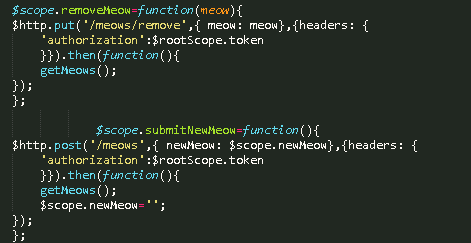


Add header variable in submitNewMeow and removeMeow function as below In Index.html file

{headers: {

'authorization':$rootScope.token

}}



Open server.js file

Change Meow model as below

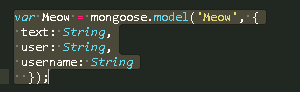
var Meow = mongoose.model('Meow', {

text: String,

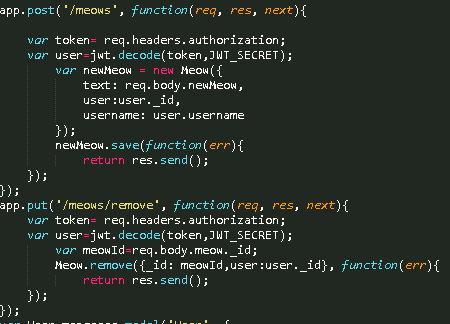
user:String,

username: String

});



Change POST and PUT meows function as below



Restart your server and check in browser. Now you can delete your own meows only not the other one.

**Happy Coding ☺**