CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V1.1.0

JavaScript Full Stack Development

- MongoDB
  - NoSQL database (document store)
  - Stores JSON documents
- Express
  - JavaScript web framework
  - On top of Node
- Angular
  - JavaScript UI framework
  - Single Page Applications
- Node
  - JavaScript server-side platform
  - Single threaded, fast and scalable

Roadmap and Outcomes

- Node.js: write asynchronous (non-blocking) code. Understand node platform to start a project.
- Express: setup express and get requests and send back responses. REST API.
- MongoDB: what NoSQL DB looks like. Full API interacting with DB.
- AngularJS: Investigate AngularJS and architect it. A single page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

Integrating MEAN

Setup

- Check endpoints working properly using REST browser plugin.
- Create angular-app folder in the application public folder.
- Add public/angular-app/app.js file (empty for now). This is angular app.
- Install AngularJS using npm (or any other way)
  - npm i angular angular-route
- Add the angular files as dependencies to project
  - <script src="node_modules/angular/angular.js"></script>
    <script src="node_modules/angular-route/angular-route.js"></script>
- Include the angular application
  - <script src="angular-app/app.js"></script>
- Enable our node application to reach Angular (add app.use)
  - app.use("/node_modules", express.static(path.join(__dirname, "node_modules")));

MEAN

Title

Get List

Get One

Rating

Get the home page from Angular

Update index.html

```
…
<html ng-app="meanGames">
…
<body>
<div ng-view></div>
…
<script src="angular-app/game-list/game-list-
controller.js"></script>
</body>
}
```

MEAN

Title

Get List

Get One

Rating

Update angular-app/app.js

```
angular.module("meanGames", ["ngRoute"]).config(config);
function config($routeProvider) {
   $routeProvider.when("/", {
      templateURL: "angular-app/game-list/games.html",
      controller:" GamesController",
      controllerAs: "vm"
   });
}
```

Add the controller angular-app/game-list/game-list-controller.js

```
angular.module("meanGames", ["ngRoute"])
.controller("GamesController", GamesController);
function GamesController() {
 const vm= this;
 vm.title= "Mean Games App";
}
```

Add the template angular-app/game-list/gmaes.html

```
<H1>{{vm.title}}</H1>
```

MEAN

Title

Get List

Get One

Rating

Get the list of games from API

Update controller to make the request, public/angular-app/game-list/game-list-controller.js

```
function GamesController($http) {
  const vm= this;
  vm.title= "Mean Games App";
  $http.get("/api/games").then(function(response) {
    vm.games= response.data;
  }
}
```

Update the template angular-app/game-list/games.html

```
<H1>{{vm.title}}</H1>
<ul>
<li ng-repeat="game in vm.games">{{game.title}}</li>
</ul>
```

MEAN

Title

Get List

GetOne

Rating

Date routing to display a game

Update public/angular-app/app.js

```
…
function config($routeProvider, $locationProvier) {
$locationProvier.hashPrefix("");
…
.when("/game/:id", {
templateUrl: "angular-app/game-display/game.html",
controller: "GameController",
controllerAs: "vm"
});
}
```

Add controller to html page public/index.html

```
…
<script src="angular-app/game-data-factory/game-data-factory.js"></script>
<script src="angular-app/game-display/game-display-controller.js"></script>
```

MEAN

Title

Get List

GetOne

Rating

Create the data factory that calls the endpoints, and it used in our app.

Create public/game-data-factory/game-data-factory.js

```
angular.module("meanGames").factory("GameDataFactory", GameDataFactory);
function GameDataFactory($http) {
  return {
    getAllGames: getAllGames,
    getOneGame: getOneGame
  };
  function getAllGames() {
    return $http.get("/api/games").then(complete).catch(failed);
  }
  function getOneGame(id) {
    return $http.get("/api/games/"+id).then(complete).catch(failed);
  }
  function complete(response){
    console.log(response.data);
    return response.data;
  }
  function failed(error) {
    return error.status.statusText;
  }
}
```

Update game-list-controller.js to use the factory

```
function GamesController(GameDataFactory) {
 const vm= this;
 vm.title= "Mean Games App";
 GameDataFactory.getAllGames().then(function(response) {
  vm.games= response;
 });
}
```

MEAN

Title

Get List

GetOne

Rating

Get data about one game, add controller and template

Add controller public/angular-app/game-display/game-display-controller.js

```
angular.module("meanGames").controller("GameController", GameController);
function GameController(GameDataFactory, $routeParams) {
    const vm= this;
    const id= $routeParams.id;
    GameDataFactory.getOneGame(id).then(function(response) {
        vm.game= response;
    });
}
```

Add the template angular-app/game-display/game.html

```
<H1>Information about game: <p>{{vm.game.title}}</p></H1>
<p>
    Price: {{vm.game.price | currency}}<BR/>
    Minimum Players: {{vm.game.minPlayers}}<BR/>
    Maximum Players: {{vm.game.maxPlayers}}<BR/>
    Minimum Age: {{vm.game.minAge}}</BR>
    Publisher: {{vm.game.publisher.name}}
</p>
```

MEAN

Title

Get List

GetOne

Rating

Selecting a game from the list

Update public/angular-app/game-list/games.html

```
…
<li ng-repeat="game in vm.games"><a ng-
href="#/game/{{game._id}}">{{game.title}}</a></li>
}
```

Display Ratings

- What is the best way to display ratings?
- Number :(
- Images :/
- Stars :)
- Custom directive

Custom Directives

MEAN

Title

Get List

GetOne

Rating

Update template public/game-display/game-display-controller.js

…

vm.rating= response.rate;

…

Update template public/game-display/game.html

<H1>Information about game: {{vm.game.title}} - {{vm.rating}} </H1>

…

We would prefer to see stars according to this number

MEAN

Title

Get List

GetOne

Rating

Update template public/game-display/game.html

```
<H1>Information about game: <p>{{vm.game.title}} <game-rating
stars={{vm.rating}}></game-rating> </p></H1>
```

Add to html file index.html

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
...
<script src="angular-app/game-rating/game-rating-directive.js></script>
```

Update controller to send an array instead of a number game-display-controller.js

```
…
    vm.rating= _getStarRating(response.rate);
  });
}
function _getStarRating(stars) {
    return new Array(rate);
}
...
```

MEAN

Title

Get List

GetOne

Rating

Create directive public/angular-app/game-rating/game-rating-directive.js

```
angular.module("meanGames").directive("gameRating".
GameRating);
function GameRating() {
  return {
    restrict: "E",
    templateUrl: "angular-app/game-rating/rating.html",
    bindToController: true,
    controller: "GameController",
    controllerAs: "vm",
    scope: {
      stars: "@"
    }
  }
}
```

Create template public/angular-app/game-rating/rating.html

```
<span ng-repeat="star  in vm.rating track by $index"
class="glyphicon glyphicon-star"></span>
```

MEAN

Title

Get List

GetOne

Rating

Use component instead public/angular-app/game-rating/game-rating-directive.js

```
angular.module("meanGames").component("gameRating"
, {
    bindings: {
      stars: "*"
    },
    templateUrl: "angular-app/game-rating/rating.html",
    controller: "GameController",
    controllerAs: "vm",
});
```

Form Validation

Forms

Field Checking

Pattern Check

Check on submit

Add Game

We will use JSBin for this part

WE can use HTML 5 form validation attributes (required, email number, url) alos AngularJS form validation ng-minlength, ng-maxlength, ng-pattern

```
<form name="myForm">
<input type="text" name="name" required ng-minlength="3"
ng-maxlength="10" ng-model="name"></input>
</form>
<p>{{myForm.$pristine}}</p>
<p>{{myForm.$dirty}}</p>
<p>{{myForm.name.$pristine}}</p>
<p>{{myForm.name.$dirty}}</p>
<p>{{myForm.name.$valid}}</p>
<p>{{myForm.name.$invalid}}</p>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<form name="myForm">
<input type="text" name="name" required ng-minlength="3" ng-maxlength="10" ng-model="name"></input>
<span ng-show="myForm.name.$dirty && myForm.name.$invalid">
This feild requires 3-10 characters.
</span>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<form name="myForm">
<input type="text" name="name" required ng-pattern="/^[0-9]{2,3}$/" ng-model="name"></input>
<span ng-show="myForm.name.$dirty && myForm.name.$invalid">
This feild requires 2 or 3 digits.
</span>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<script>
 angular.module("myApp", []).controller("MyController", MyController);
 function MyController() {
  const vm= this;
  vm.message= "hello";
  vm.isSubmitted= false;
  vm.add= function() {
   if (vm.myForm.$valid) {
    console.log("Add to database...");
   } else {
    vm.isSubmitted= true;
   }
  }
 }
</script>
...
 <form name="vm.myForm" ng-submit="vm.add()">
 Please enter age greater than 9: <input type="text" name="name"
required ng-pattern="/^[0-9]{2,3}$/" ng-model="name"></input>
 <span ng-show="vm.myForm.name.$dirty &&
vm.myForm.name.$invalid && vm.isSubmitted">
   This feild requires 2 or 3 digits.
 </span>
 <button type="submit">Add data</button>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Add Game form to public/angular-app/game-list/games.html

```
<form name="vm.gameForm" ng-submit="vm.addGame()" >
    To add a new game please fill in all the fields below:<BR/>
    Title: <input type="text" name="title" required ng-model="vm.newGameTitle"
style="color:black"/><BR/>
    Price: <input type="text" name="price" required ng-
model="vm.newGamePrice" style="color:black"/><BR/>
    Year of Publication: <input type="text" name="year" required ng-
model="vm.newGameYear" style="color:black"/><BR/>
    Rating: <input type="text" name="rating" required ng-
model="vm.newGameRating" style="color:black"/><BR/>
    Minimum Number of Players: <input type="text" name="minPlayers" required
ng-model="vm.newGameMinPlayers" style="color:black"/>
    <span ng-show="vm.gameForm.minPlayers.$dirty &&
vm.gameForm.minPlayers.$invalid && vm.gameForm.isSubmitted"
style="color:black">Minimum players must be at least 1.</span>
    <BR/>
    Maximum Number of Players: <input type="text" name="maxPlayers" required
ng-model="vm.newGameMaxPlayers" style="color:black"/><BR/>
    Minimum Recommended Player Age: <input type="text" name="minAge"
required ng-model="vm.newGameMinAge" style="color:black"/><BR/>
    Designer name: <input type="text" name="designer" required ng-
model="vm.newGameDesigner" style="color:black"/><BR/>
    <button type="submit" class="btn-success">Add Game</button><BR/>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Add controller functionality for submitting. Update public/angular-app/game-list/game-list-controller.js

```
function GamesController(GameDataFactory) {
    const vm= this;
    vm.title= "Mean Games App";
    vm.isSubmitted= false;
    GameDataFactory.getAllGames().then(function(response) {
        // console.log(response);
        vm.games=  response;
    });
    vm.addGame= function(){
        const postData= {
            title: vm.newGameTitle,
            price: vm.newGamePrice,
            rate: vm.newGameRating,
            year: vm.newGameYear,
            rating: vm.newGameRating,
            minPlayers: vm.newGameMinPlayers,
            maxPlayers: vm.newGameMaxPlayers,
            minAge: vm.newGameMinAge,
            designers: vm.newGameDesigner,
        };
        if (vm.gameForm.$valid) {
            GameDataFactory.postGame(postData).then(function(response){
                console.log("Game saved");
                //
            }).catch(function(error) {
                console.log(error);
            });
        } else {
            vm.isSubmitted= true;
        }
    };
}
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Update the Factory public/angular-app/game-data-factory/game-data-factory.js

```
function GameDataFactory($http) {
    return {
        getAllGames: getAllGames,
        getOneGame: getOneGame,
        postGame: postGame
    };

...
    function postGame(game) {
        return $http.post("/api/games/",
game).then(complete).catch(failed);
    }
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Enable JSON processing. Update app05,js

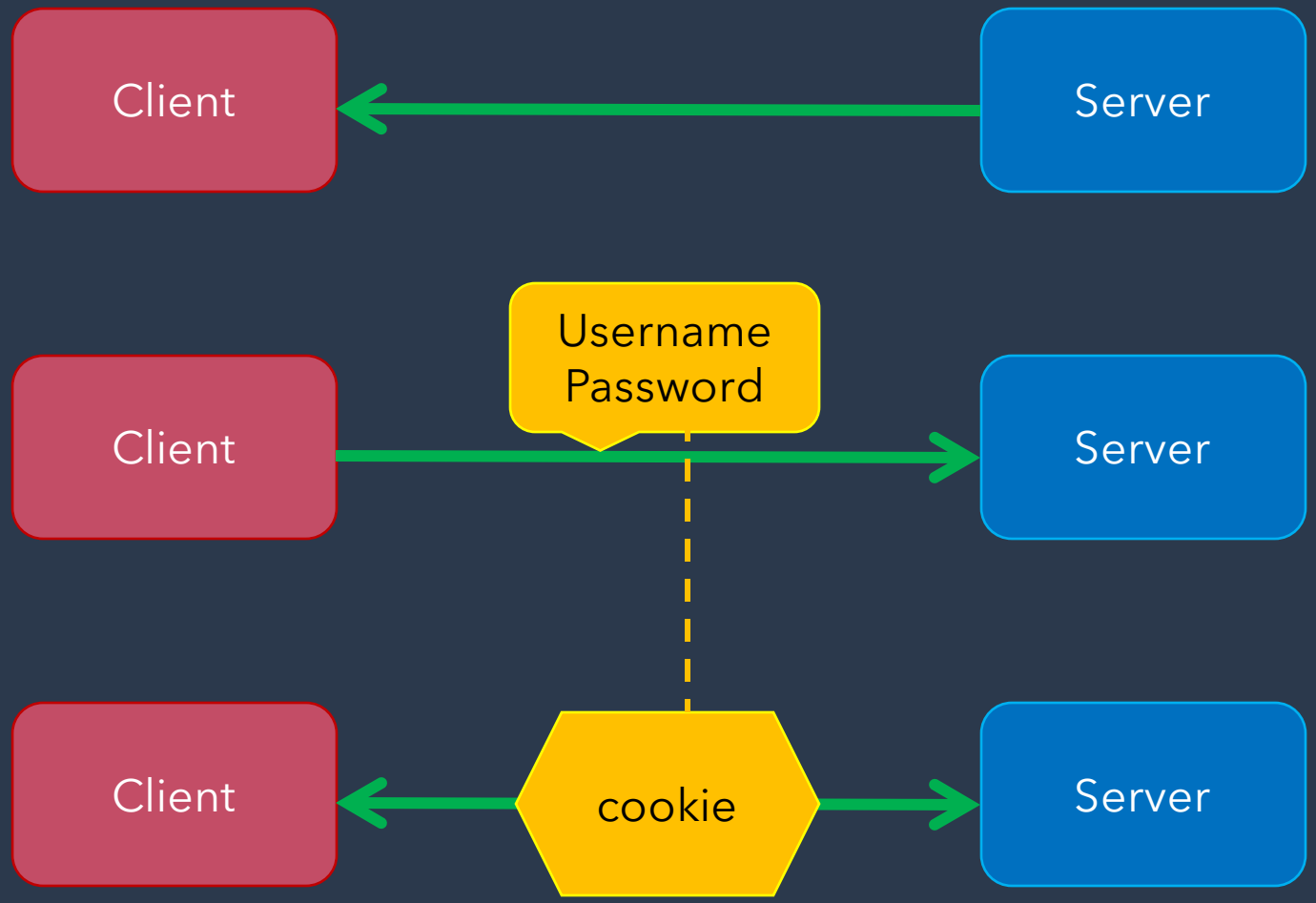
```
...
app.use(express.urlencoded({extended  : false}));
app.use(express.json());
...
```

Authentication

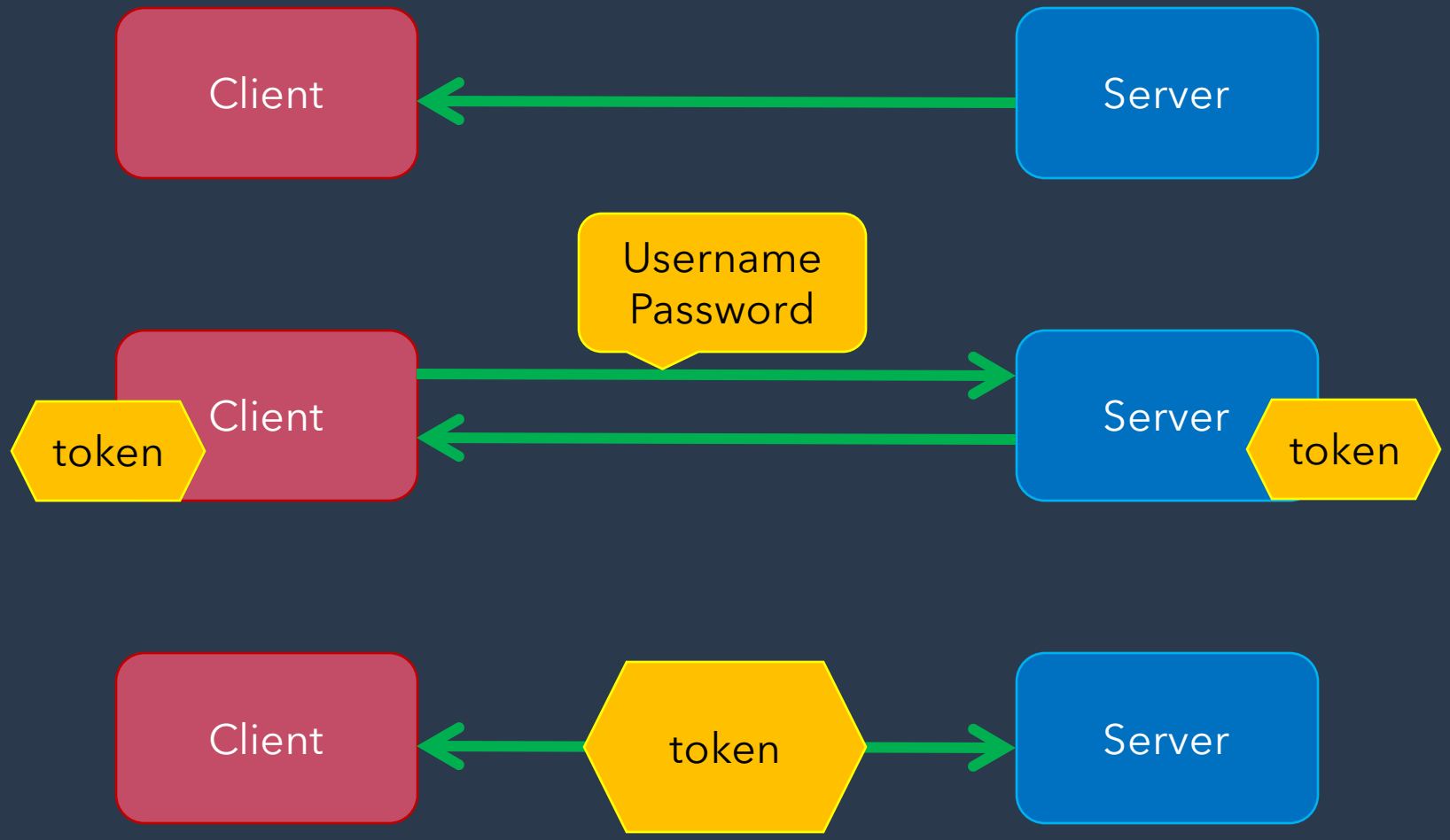
What is Authentication and Authorization

- Are you who you say you are?
- Do you have the authority (privilige) to access this?
- Authentic: Original.
- Authroized: Allowed to do this.

Classic Server Based Authentication



Token Based Authentication



JWT

- JSON Web Token
- "Header", "Payload", "Signature"
- Payload:
  - Any data, username, roles, ...
- Authentication, and encryption methods.

Authentication

Users DB

Encryption

Token Authentication

Before we can authenticate, we need to have a credentials DB. A DB of users with a first names, usernames, and passwords.

Create api/data/users-model.js

```
var mongoose= require("mongoose");
var userSchema= new mongoose.Schema({
username: {
type: String,
unique: true,
required: true
},
name: {type: String},
password: {type: String, required: true}
});
mongoose.model("User",  userSchema);
```

Authentication

Users DB

Encryption

Token Authentication

Make sure we bring in the Schema and module for the application. Update users/api/data/db.js

```
…
require("./users.model");
```

Add the authentication routes to api/routes/index.js

```
…
var controllerUsers=
require("../controllers/users.controller.js");
…
router.route("/users/register")
        .post(controllerUsers.register);
router.route("/users/login").post(controllerUsers.login);
module.exports= router;
```

Authentication

Users DB

Encryption

Token

Authentication

Add a new controller. Create api/controllers/users.controller.js

```
var mongoose= require("mongoose");
var User= mongoose.model("User");
module.exports.register= function(req, res) {
  console.log("Registering user");
  var username= req.body.username;
  var name= req.body.name || null;
  var password= req.body.password;
  User.create({username: username, name: name, password: password},
function(err, user) {
    if (err) { console.log(err); res.status(400).json(err);}
    else {console.log("user created", user); res.status(200).json(user);}
  });
};
module.exports.login= function(req, res) {
  console.log("Logging in user");
  var username= req.body.username;
  var password= req.body.password;
  User.findOne({username: username}).exec(function(err, user) {
    if (err) { console.log(err); res.status(400).json(err);}
    if (user) { console.log("user found", user); res.status(200).json(user);}
    else {console.log("user not found", user); res.status(400).json("Unauthorized");}
  });
};
```

Authentication

Users DB

Encryption

Token Authentication

Install the encryption package bcrypt-nodejs

```
npm I bcrypt-nodejs
```

Modify controller to use encryption. Update api/controllers/users-controller.js

```
var mongoose= require("mongoose");
var User= mongoose.model("User");
var bcrypt= require("bcrypt-nodejs");
module.exports.register= function(req, res){
  console.log("Registering user");
  var username= req.body.username;
  var name= req.body.name || null;
  var password= bcrypt.hashSync(req.body.password, bcrypt.genSaltSync(10));
  User.create({username: username, name: name, password: password}, function(err, user){
    if (err) { console.log(err); res.status(400).json(err);}
    else {console.log("user created", user); res.status(200).json(user);}
  });
};
module.exports.login= function(req, res) {
  console.log("Logging in user");
  var username= req.body.username;
  var password= req.body.password;
  User.findOne({username: username}).exec(function(err, user) {
    if (err) { console.log(err); res.status(400).json(err);}
    if (user) {
      if (bcrypt.compareSync(password, user.password)) {
        console.log("user found", user); res.status(200).json(user);
      } else { res.status(401).json("Unauthorized");}
    } else {console.log("user not found", user); res.status(400).json("Unauthorized");}});
};
```

Authentication

Users DB

Encryption

Token

Authentication

Install the token generation package

```
npm i jsonwebtoken
```

Use the token generator in the controller. Update api/controllers/users-controller.js

```
…
var jwt= require("jsonwebtoken);
…
module.exports.login= function(req, res) {
  console.log("Logging in user");
  var username= req.body.username;
  var password= req.body.password;
  User.findOne({username:  username}).exec(function(err, user) {
    if (err) { console.log(err); res.status(400).json(err);}
    if (user) {
      if (bcrypt.compareSync(password, user.password)) {
        console.log("user found", user);
        var token= jwt.sign({username:  user.username}, "cs572", {expiresIn: 3600});
        res.status(200).json({success:  true, token: token});
      } else { res.status(401).json("Unauthorized");}
    } else {console.log("user not found", user);
res.status(400).json("Unauthorized");}});
};
```

Authentication

Users DB

Encryption

Token

Authentication

Create a middleware function to check the existance of a token, and the token is valid. If successful it will call the next middelware function.

Update api/controllers/users-controller.js

```
...
module.exports.authenticate= function(req, res, next) {
 var headerExists= req.headers.authorization;
 if (headerExists) {
  var token= req.headers.authorization.split(" ")[1];
  jwt.verify(token, "cs572", function(err, decoded){
   if (err) { console.log(err); res.status(401).json("Unauthorized");
   } else {
    req.user= decoded.username;
    next();
   }});
  } else {res.status(403).json("No token provided");}
};
```

Validate JWT Tokens

- Use jwt.io to learn about JWT and validate tokens.
- Paste your token
  - You see the infomration in the token.
  - The signiture is not validated, because it is not in the token.
  - Type your signiture and the token signiture is validated.

Registration UI

UI

Registration

Login

Authentication

Login code

Token Management

Create a registration route, controller, and template.

Update public/angular-app/app.js

```
…
.when("/register",  {
  templateUrl: "angular-app/register/register.html",
  controller: "RegisterController",
  controllerAs: "vm"
});
```

Create registration form public/angular-app/register/register.html

```
<H1>Register</H1>
<DIV ng-if="vm.message" class="alert alert-success" role="alert">
  <P>{{ vm.message}}</P>
</DIV>
<DIV ng-if="vm.err" class="alert alert-danger" role="alert">
  {{vm.err}}
</DIV>
<FORM ng-hide="vm.message" name="register" ng-submit="vm.register()">
  <DIV class="form-group">
    <label for="username">Username</label>
    <input type="username" class="form-control" id="username" placeholder="Username" ng-model="vm.username"
autocapitalize="none"/>
  </DIV>
  <DIV class="form-group">
    <label for="password">Password</label>
    <input type="password" class="form-control" id="password" placeholder="Password" ng-model="vm.password" autocapitalize="none"/>
  </DIV>
  <DIV class="form-group">
    <label for="password-repeat">Repeat  Password</label>
    <input type="password" class="form-control" id="password-repeat" placeholder="Password" ng-model="vm.passwordRepeat"
autocapitalize="none"/>
  </DIV>
<BUTTON type="submit" class="btn btn-success">Register</BUTTON>
</FORM>
```

UI

Registration

Login

Authentication

Login code

Token Management

Create a registration route, controller, and template.

Create register controller public/angular-app/register/register-controller.js

```
angular.module("meanGames").controller("RegisterController",  RegisterController);
function RegisterController() {
  var vm= this;
  vm.register= function($http){
    var user= {username: vm.username, pasword: vm.password};
    if (!vm.username || !vm.password) { vm.err=  "Please add a username and password.";}
    else {
      if (vm.password !== vm.passwordRepeate){
        vm.err=  "Please make sure the passwords match.";
      } else {
        $http.post("/api/users/register", user).then(function(result) {
          console.log(result);
          vm.message= "Successful registration, please login.";
          vm.err=  "";
        }).catch(function(err) {console.log(err);});
      }
    }
  }
};
```

Update public/index.html to include the controller script

```
…
<script src="angular-app/register/register-controller.js"></script>
…
```

UI

Registration

Login

Authentication

Login code

Token Management

Some application changes to support login.
Add welcome page as home page.
Create a welcome template. Create public/angular-app/welcome/welcome.html

```
<H1>Welcome to MEAN Games</H1>
```

Update public/index.html

```
…
  <games-Navigation></games-Navigation>
  <DIV class="container">
    <div ng-view></div>
  </DIV>
…
  <script src="angular-app/navigation-directive/navigation-directive.js"></script>
  <script src="angular-app/login/login-controller.js"></script>
…
```

Add style to our container. Update public/css/custom.css

```
…
.container {
  padding: 60px 0 0 0;
}
```

Update public/angular-app/app.js

```
…
.when("/",  {
  templateUrl:  "angular-app/welcome/welcome.html"
})
.when("/games",  {
  templateUrl:  "angular-app/game-list/games.html",
  controller: GamesController,
  controllerAs: "vm"
})
…
.otherwise({redirectTo:   "/"});
}
```

UI

Registration

Login

Authentication

Login code

Token Management

Add a navigation bar using a directive. Create public/angular-app/navigation-directive/navigation-directive.html

```
<nav ng-controller="LoginController as vm" class="navbar navbar-expand-lg navbar-fixed-top bg-dark" ng-
init="vm.init()">
 <div class="container-fluid">
  <div class="navbar-header">
   <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-
expanded="false" aria-controls="navbar">
    <span class="sr-only">Toggle navigation</span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
   </button>
   <a class="navbar-brand" href="#/">MeanGames</a>
  </div>
  <div id="navbar" class="collapse navbar-collapse">
   <ul class="nav navbar-nav">
    <li ng-class="vm.isActiveTab('')" class="active"><a href="#/">Home</a></li>
    <li ng-class="vm.isActiveTab('games')"><a href="#/games">Games</a></li>
    <li ng-class="vm.isActiveTab('register')"><a href="#/register">Register</a></li>
   </ul>
   <div class="col-xs-6">
    <form ng-if="!vm.isLoggedIn()" class="navbar-form navbar-right" role="login" ng-submit="vm.login()">
     <div class="form-group" ng-class="{'has-error': vm.err}">
      <input type="text" class="form-control input-sm" placeholder="Username" ng-model="vm.username"
autocapitalize="none">
     </div>
     <div class="form-group" ng-class="{'has-error': vm.err}">
      <input type="password" class="form-control input-sm" placeholder="Password" ng-
model="vm.password" autocapitalize="none">
     </div>
     <button type="submit" class="btn btn-sm btn-success">Login</button>
    </form>
   </div>
   <ul ng-if="vm.isLoggedIn()" class="new navbar-nav navbar-right">
    <li><p class="navbar-text">Welcome {{vm.loggedinUser}}</p></li>
    <li><button class="navbar-btn btn-sm btn-info" ng-click="vm.logout()">Logout</button></li>
   </ul>
  </div>
 </div>
</nav>
```

UI

Registration

Login

Authentication

Login code

Token Management

Create public/angular-app/navigation-directive/navigation-directive.js

```
angular.module("meanGames").directive("gamesNavigation",
GamesNavigation);
function GamesNavigation() {
  return {
    restrict: "E",
    templateUrl: "angular-app/navigation-directive/navigation-
directive.html"
  };
}
```

Call the directive in public/angular-app/login/login-controller.js

```
angular.module("meanGames").controller("LoginController",
LoginController);
function LoginController() {
  var vm= this;
}
```

UI

Registration

Login

Authentication

Login code

Token

Management

Create public/angular-app/authentication/auth-interceptor.js

```
angular.module("meanGames").factory("AuthInterceptor",  AuthInterceptor);
function AuthInterceptor($location, $q, $window, AuthFactory) {
  return { request: request, response: response, responseError:  responseError}
  function request(config) {
    config.headers= config.headers || {};
    if ($window.sessionStorage.token) {
      config.headers.Authorization= "Bearer "+ $window.sessionStorage.token;
    }
    return config;
  }
  function response(response){
    if (response.status === 200 && $window.sessionStorage.token && !AuthFactory.isLoggedIn)
{ AuthFactory.isLoggedIn= true;}
    if (response.status === 401) { AuthFactory.isLoggedIn= false;}
    return response || $q.when(response);
  }
  function responseError(rejection) {
    if (rejection.status ===  401 || rejection.status === 403) {
      delete $window.sessionStorage.token;
      AuthFactory.isLoggedIn= false;
      $location.path("/");
    }
    return $q.reject(rejection);
  }
}
```

Create public/angular-app/authentication/auth-factory.js

```
angular.module("meanGames").factory("AuthFactory",  AuthFactory);
function AuthFactory() {
  return {auth: auth};
  var auth= {ifLoggedId: false};
}
```

Add factories to public/index.html

```
<script src="angular-app/authentication/auth-factory.js"></script>
<script src="angular-app/authentication/auth-interceptor.js"></script>
```

UI

Registration

Login

Authentication

Login code

Token

Management

User authentication in controller, update public/angular-app/login/login-controller.js

```
function LoginController($http, $location, $window, AuthFactory) {
  var vm= this;
  vm.isLoggedIn= function() {
    if (AuthFactory.isLoggedIn) { return true}
    else {return false;}
  };
  vm.login= function() {
  }
  vm.logout= function() {
  }
  vm.isActiveTab= function(url) {
    var currentPath= $location.path().split("/")[1];
    return (url === currentPath ? "active" : "");
  }
}
```

UI

Registration

Login UI

Authentication

Login code

Token Management

Write Login code in controller, update public/angular-app/login/login-controller.js

```
....
  vm.login= function() {
    if (vm.username  && vm.password) {
      var user= {
        username:  vm.username,
        password: vm.password
      };
      $http.post("/api/users/login", user).then(function(response) {
        console.log(response);
      }).catch(function(err) {
        console.log(err);
      });
    }
  }
...
```

UI

Registration

Login UI

Authentication

Login code

Token Management

Add token management on client side. Update public/angular-app/login/login-controller.js

```
....
 vm.login= function() {
  if (vm.username && vm.password) {
   var user= {
    username: vm.username,
    password: vm.password
   };
   $http.post("/api/users/login", user).then(function(response) {
    if (response.data.success) {
     $window.sessionStorage.token= response.data.token;
     AuthFactory.isLoggedIn= true;
    }
   }).catch(function(er) {
    console.log(err);
   });
  }
 }
 vm.logout= function() {
  AuthFactory.isLoggedIn= false;
  delete $window.sessionStorage.token;
  $location.path("/");
 }
...
```

UI

Registration

Login UI

Authentication

Login code

Token Management

Dispaly menu items based on login status. Update public/angular-app/app.js

```
angular.module("meanGames", ["ngRoute"]).config(config).run(run);
function config($httpProvider, $routeProvider, $locationProvider) {
 $httpProvider.interceptors.push("AuthInterceptor");
 ....
 .when("/", {
  templateUrl: "angular-app/welcome/welcome.html",
  access: {restricted: false}
 })
 .when("/games", {
  ...
  access: {restricted: false}
 })
 .when("/game/:id", {
  ...
  access: {restricted: false}
 })
 .when("/register", {
  ...
  access: {restricted: false}
 })
 .when("/profile", {
  templateUrl: "angular-app/profile/profile.html",
  controllerAs: "vm",
  access: {restricted: true}
 })
...
}
function run ($rootScope, $location, $window, AuthFactory) {
 $rootScope.$on("$routeChangeStart", function(event, nextRoute, currentRoute) {
  if (nextRoute.access !== undefined && nextRoute.access.restricted && !$window.sessionStorage.token &&
!AuthFactory.isLoggedIn) {
    event.preventDefault(); // Do not go to that path
    $location.path("/");      // Instead go to the root
  }
 });
}
```

UI

Registration

Login UI

Authentication

Login code

Token Management

Update the navigation directive to display the menu according to the new managemnet rules. Update public/angular-app/navigation-directive/navigation-directive.html

```
...
      <li ng-class="vm.isActiveTab('register')" ng-if="!vm.isLoggedIn()"><a href="#/register">Register</a></li>
      <li ng-class="vm.isActiveTab('profile')" ng-if="vm.isLoggedIn()"><a href="#/profile">Profile</a></li>
...
```

Add the profile template, create public/angular-app/profile/profile.html

```
<H1>This is the profile Page.</H1>
```

UI

Registration
Login UI
Authentication
Login code
Token Management

Access JSON Web Token attributes from Angular.

Install package angular-jwt

```
npm i angular-jwt
```

Update LoginController to use angular-jwt to access token attributes. Update public/angular-app/login/login-controller.js

```
angular.module("meanGames").controller("LoginController", LoginController);
function LoginController($http, $location, $window, AuthFactory, jwtHelper) {
...
 vm.login = function() {
 ...
      AuthFactory.isLoggedIn= true;
      var token= $window.sessionStorage.token;
      var decodedToken= jwtHelper.decodeToken(token);
      vm.loggedInUser= decodedToken.username;
...
```

Add dependency on angular-jwt in /public/app.js

```
angular.module("meanGames", ["ngRoute", "angular-jwt"]).config(config).run(run);
...
```

Add angular-jwt to public/index.html

```
...
<script src="node_modules/angular-jwt/dist/angular-jwt.js"></script>
...
```

Authentication Based Usage

- UI
  - Only display pages when users are authenticated.
- BL
  - Only accept api calls from authenticated users.

Authentication

UI

BL

UI where only logged in users can add games.
Modify the public/angular-app/game-list/game-list-controller.js to support this

```
function GamesController($route, GameDataFactory, AuthFactory) {
  …
  vm.isLoggedIn= function() {
    if (AuthFactory.isLoggedIn) {return true;}
    else {return false;}
  };
  …
```

Update the template to use the funciton when displaying game-list.html

```
…
<div ng-if="vm.isLoggedIn()">
  <form name="vm.gameForm"  ng-submit="vm.addGame()"  >
  …
</div>
…
```

Authentication

UI

BL

BL where only logged in users can add games. Only accept a new game to be added if a user is authenticated. Update api/route/index.js

```
...
router.route("/games")
.get(controllerGames.gameGetAll)
.post(controllerUsers.authenticate,
controllerGames.gameAddOne);
...
```

Main Points

- MEAN is the ultimate separation of concerns (SoC). Not only do we have each responsibility separated in code, but each one is handled by a different framework.
- Token authentication integrates perfectly with HTTP and supports JSON which makes it perfect for MEAN applications. Both HTTP and JWT are stateless.
- We can use JWT to prevent the UI from displaying operations the user may not be allowed to perform. At the same time, we use JWT to authorize API calls.