**Assignment 6 – NodeSecurity**

**Setup**

* ~~Create a new collection and name it users.~~
* ~~Using the CLI, add the bcryptjs package to your GitHub repository.~~ 
  + ~~CLI command:~~
    - **~~npm install bcryptjs~~**
* A~~dd a new file under the~~ **~~model’s~~** ~~directory and name it~~ **~~<yourLastName>-user.js~~**~~.~~
* ~~Add a new file under the~~ **~~route’s~~** ~~directory and name it~~ **~~<yourLastName>-session-routes.js~~**~~.~~

**<yourLastName>-user.js**

* ~~Add a require statement for mongoose and assign it to a variable named~~ **~~mongoose~~**~~.~~
* ~~Add a new variable named~~ **~~Schema~~** ~~and assign it the~~ **~~mongoose.Schema~~** ~~object.~~
* C~~reate a schema named userSchema with the following fields:~~

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Schema** |
| userName | String | n/a |
| Password | String | n/a |
| emailAddress | Array |  |

* ~~Name the model “User” and export it using~~ **~~module.exports~~**

**<yourLastName>-session-routes.js**

* ~~Add the appropriate requirement statements (express, router, User, and bcrypt)~~
* ~~Create a variable named router and assign it the express.Router() function.~~
* ~~Add a variable named saltRounds with an integer value of 10~~
* Create two operations: signup and login

# **Signup**

**Operation: POST**

**Path: /api/signup**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestBody | userName | String |
| RequestBody | Password | String |
| RequestBody | emailAddress | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Registered user |
| 401 | Username is already in use |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Wrap the code in a try/catch block.
  + Query the users collection using the findOne() function and the username form the RequestBody.
  + Add and if…else block that checked the returned value from the query.
    - if(!user)
      * Create a new object literal named newRegisteredUser and map the RequestBody values to the object’s properties.
      * For the password property, use the bcrypt package to hashSync() the password.
        + Code Snippet: 
      * Call the create() function on the User model and save the record to MongoDB.
    - if(user)
      * Return the message identified under status 401.
  + Describe the operation using the OpenAPI Specification above the request as developer comments.

# **Login**

**Operation: POST**

**Path: /api/login**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestBody | userName | String |
| RequestBody | Password | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | User logged in |
| 401 | Invalid username and/or password |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Wrap the code in a try/catch block.
  + Query the users collection using the findOne() function and the username from the RequestBody.
  + Add and if…else block that checks the returned value from the query.
    - if(user)
      * Compare the RequestBody password against the saved user’s password using the bcrypt.compareSync() function.
        + Code snippet: 
      * Add and if…else block that checks whether the password is valid.
        + For valid passwords, return the message identified for status 200.
        + For invalid passwords, return the message identified under status 401.
    - if(!user)
      * Return the message identified under status 401
  + Describe the operation using the OpenAPI Specification above the request as developer comments.
* Export the router using module.exports.
* Test one of the API’s using the generated Swagger documentation at <https://localhost:3000/api-docs> and test the second API’s using SoapUI. Take screenshots of the testing you did in SoapUI and from the Swagger documentation link.
* The screenshots you take of SoapUI and Swagger must be added to your personal portfolio website under the “API Unit Tests Page.” This is a gradable item.
* If you run into issues, refer to the courses GitHub repository and the work you completed in WEB 340.