**Assignment 7 – NodeShopper**

**Setup**

* Create a new collection named customers.
* Add a new file under the **model’s** directory and name it **<yourLastName>-customer.js**.
* Add a new file under the **route’s** directory and name it **<yourLastName>-node-shopper-routes.js**.

**<yourLastName>-customer.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.
* Create a schema named lineItemSchema with the following fields:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Schema** |
| name | String | n/a |
| price | Number | n/a |
| quantity | Number | n/a |

* Create a schema named invoice with the following fields:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Schema** |
| subtotal | Number | n/a |
| tax | Number | n/a |
| dateCreated | String | n/a |
| dateShipped | String | n/a |
| lineItems | Array | lineItemSchema |

* Create a schema named customerSchema with the following fields:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Schema** |
| firstName | String | n/a |
| lastName | String | n/a |
| userName | String | n/a |
| invoices | Array | invoiceSchema |

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

**<yourLastName>-node-shopper-routes.js**

* Create three operations: createCustomer, creatInvoiceByUserName, and findAllInvoicesByUserName

# **createCustomer**

**Operation: POST**

**Path: /api/customers**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestBody | firstName | String |
| RequestBody | lastName | String |
| RequestBody | userName | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Customer added to MongoDB |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Wrap the code in a try/catch block.
  + Create an object literal named newCustomer and bind the RequestBody to the object literal.
  + Call the Customer.create() function and use the newCustomer object literal as the argument.
  + Describe the operation using the OpenAPI Specification above the request as developer comments.

# **createInvoiceByUserName**

**Operation: POST**

**Path: /api/customers/:username/invoices**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequestParams | username | String |
| RequestBody | subtotal | String |
| RequestBody | tax | String |
| RequestBody | dateCreated | String |
| RequestBody | dateShipped | String |
| RequestBody | lineItems | Array of lineItem objects |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Customer added to MongoDB |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Query the customers collection using the findOne() function and the username from the RequestParams object.
  + Create an object literal named newInvoice and map the values from the RequestBody to its properties.
  + Call the push() function off of the invoices array and pass-in the newInvoice object literal.
  + Call the save() function on the Customer model and save the results to MongoDB.
  + Describe the operation using the OpenAPI Specification above the request as developer comments.

# **findAllInvoicesByUserName**

**Operation: GET**

**Path: /api/customers/:username/invoices**

|  |  |  |
| --- | --- | --- |
| **Param Type** | **Field** | **Data Type** |
| RequesParams | username | String |

|  |  |
| --- | --- |
| **Response Code** | **Message** |
| 200 | Customer added to MongoDB |
| 500 | Server Exception |
| 501 | MongoDB Exception |

**Additional Programming Requirements**

* + Query the customers collection using the findOne() function and the username from the RequestParams object.
  + Return the results using the res.json() function.
* 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 two 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.