# CSC570 NoSQL Databases

# Programming Assignment #1

Using the same Node.js application from Lab 3 Part 3, implement the MongoDB blogs collection web and API front-end routes accordingly. The URLs will have the same format that we had for the users model but they will be blogs instead. For example, <https://csc570e.uis.edu:9443/blogs/>

You will need to implement the blogs model (models/blogs.js) as well as their respective views/ files and add the routes to server.js. Keep in mind that the blogs model includes many more fields than the users model from Lab 3 Part 3. For this assignment, you will need to implement the ability to add/edit/delete every field for blogs, including comments.

You will not need to submit anything to GitHub. I will grade your assignment by checking the URLs.

Video Tutorials

The video lectures from Lab 3 Part 3 will be helpful for this assignment as well:

**Learning Node.js**

<https://www.lynda.com/Node-js-tutorials/Welcome/612195/677534-4.html?org=uis.edu> (You may need to click the Sign In button in the upper right corner, and then log in with your UIS NetID and password.)

**Access from Programming Language (Node.js Mongoose ORM**)

[68 min 05 sec]:

https://cdnapisec.kaltura.com/index.php/extwidget/preview/partner\_id/1371761/uiconf\_id/13362791/entry\_id/1\_b88z1mj2/embed/dynamic

Use this table to determine which container is yours. You will log into the share with .\NetID for the username (.\tllos1 for example) and your UIN for the password.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Netid** | **Windows share** | | **Url of the PHP application** | |
| aa35 | \\10.64.3.56\aa35 | | <https://csc570e.uis.edu:9444> | |
| chun1 | \\10.64.3.56\chun1 | | <https://csc570e.uis.edu:9445> | |
| csoto24 | \\10.64.3.56\csoto24 | | <https://csc570e.uis.edu:9446> | |
| dlo3 | \\10.64.3.56\dlo3 | | <https://csc570e.uis.edu:9447> | |
| dpine4 | \\10.64.3.56\dpine4 | | <https://csc570e.uis.edu:9448> | |
| ebarr5 | \\10.64.3.56\ebarr5 | | <https://csc570e.uis.edu:9449> | |
| esuzu2 | \\10.64.3.56\esuzu2 | | <https://csc570e.uis.edu:9450> | |
| fbata2 | \\10.64.3.56\fbata2 | | <https://csc570e.uis.edu:9451> | |
| gmoldo2 | \\10.64.3.56\gmoldo2 | | <https://csc570e.uis.edu:9452> | |
| jruvu2 | \\10.64.3.56\jruvu2 | | <https://csc570e.uis.edu:9453> | |
| jryan37 | \\10.64.3.56\jryan37 | | <https://csc570e.uis.edu:9454> | |
| momea2 | \\10.64.3.56\momea2 | | <https://csc570e.uis.edu:9455> | |
| mturo2 | \\10.64.3.56\mturo2 | | <https://csc570e.uis.edu:9456> | |
| mwill34 | \\10.64.3.56\mwill34 | | <https://csc570e.uis.edu:9457> | |
| svu3 | \\10.64.3.56\svu3 | | <https://csc570e.uis.edu:9458> | |
| vpate54 | \\10.64.3.56\vpate54 | | <https://csc570e.uis.edu:9459> | |
|  | |  | |  | |

You will need to update the server.js file with the IP address of your MongoDB VM. Ex:

mongoose.connect('mongodb://10.92.130.109:27017/blogger', { useNewUrlParser: true });

In order to implement the user.js model you will need to look up the Mongoose documentation (<https://mongoosejs.com/docs/api.html>) and figure out what parameters each method takes and what it returns.