CS602 Module3 Assignment

General Rules for Homework Assignments

- You are strongly encouraged to add comments throughout the program. Doing so will help your instructor to understand your programming logic and grade you more accurately.
- You must work on your assignments individually. You are not allowed to copy the answers from the others.
- Each assignment has a strict deadline. Assignments submitted after the deadline will carry a penalty.
- When the term *lastName* is referenced in an assignment, please replace it with your last name.

Download and extract the starter template zip file, CS602_HW3_*lastName. Rename the folder with your last name.* Complete the corresponding assignment files in this folder.

Part 1 – Express & MongoDB (25 Points)

This part is a redo of Module2 Assignment Part2. The web application remains the same. Redo the ZipCodeModule implementation using the file *mongo_zipCodeModule_v2.js* so that the three module functions get the data from the pre-populated MongoDB database as given in the assignment template files. The client test code is provided in *client.js* file and should produce the output as shown below after implementing the module functionality.

```
> node client.js

Lookup by zip code (02215)
{ _id: '02215', city: 'BOSTON', pop: 17769, state: 'MA' }

Lookup by zip code (99999)
undefined

Lookup by city (BOSTON, MA)
{
  city: 'BOSTON',
  state: 'MA',
  data: [
    { zip: '02108', pop: 3697 },
    { zip: '02109', pop: 3926 },
    { zip: '02110', pop: 957 },
    { zip: '02111', pop: 3759 },
    { zip: '02114', pop: 10246 },
    { zip: '02113', pop: 6698 },
    { zip: '02115', pop: 25597 },
    { zip: '02116', pop: 17459 },
    { zip: '02210', pop: 308 },
    { zip: '02199', pop: 886 },
    { zip: '02215', pop: 17769 }
  ]
}

Get Population by State (MA)
{ state: 'MA', pop: 6016425 }
```

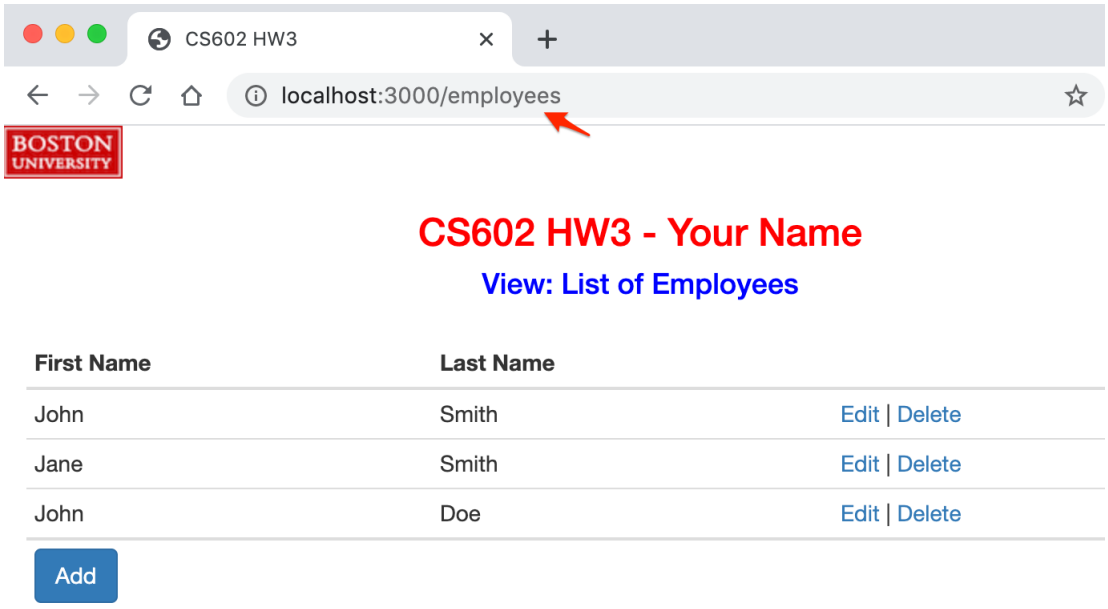
Now, test the web application similar to Module2 Assignment Part2 by copying the corresponding code from your Module2 submission.

Part 2 – Express Routing & Mongoose (75 Points)

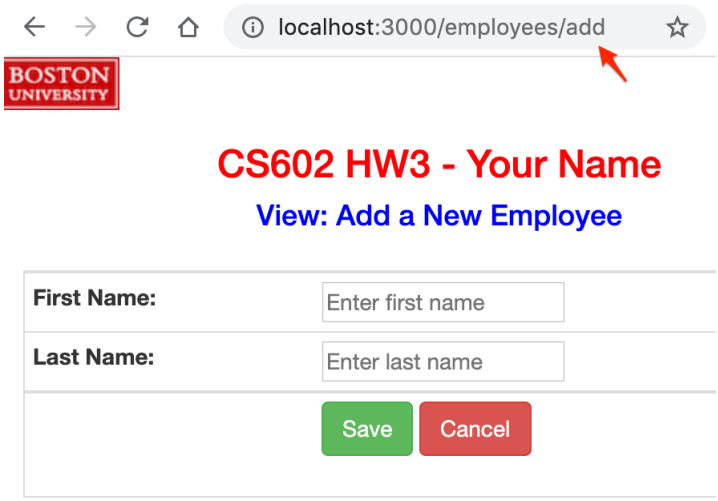
Complete the Employee Mongoose schema (*employeeDB*.js) for the employee data to be stored in MongoDB with only the firstName and lastName properties for each employee. If you are using your own MongoDB database, run the initDB.js using node to initialize the database collection with at least three employees.

- Now, complete the Express web application (*server*.js) similar to ex13_caseStudy and use the Express Router to configure the paths for adding a new employee, deleting an existing employee, displaying all employees, and editing an existing employee. The application should have functionality similar to the ex13_caseStudy covered in the lecture. The data is persisted to the MongoDB database using Mongoose.
- Add, Delete, and Edit views should have the Cancel feature.

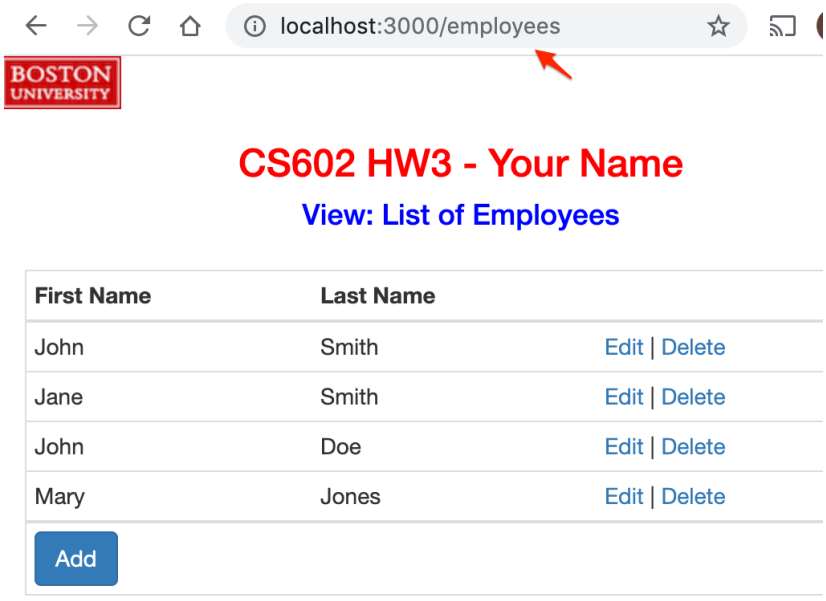
The initial display of the home page of the application (*displayEmployeesView*) is shown below:



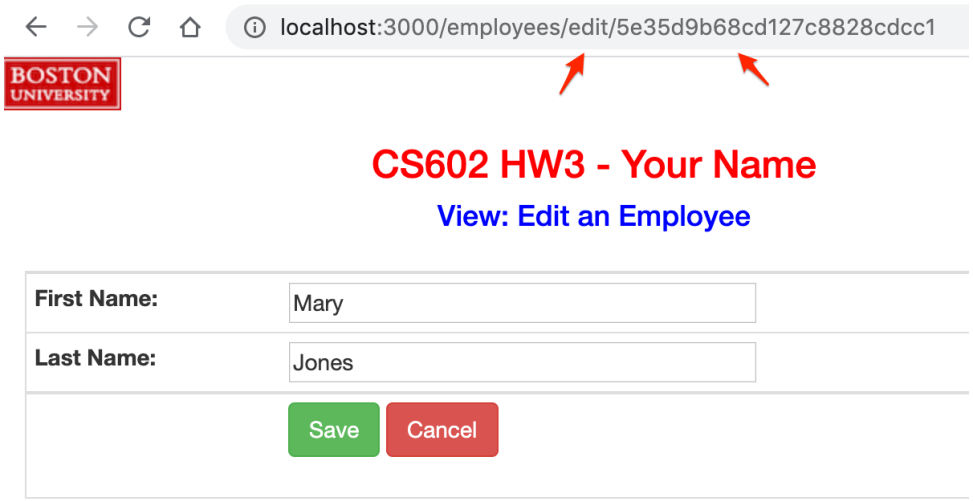
If the user clicks the **Add** button, the *addEmployeeView* is rendered as below:



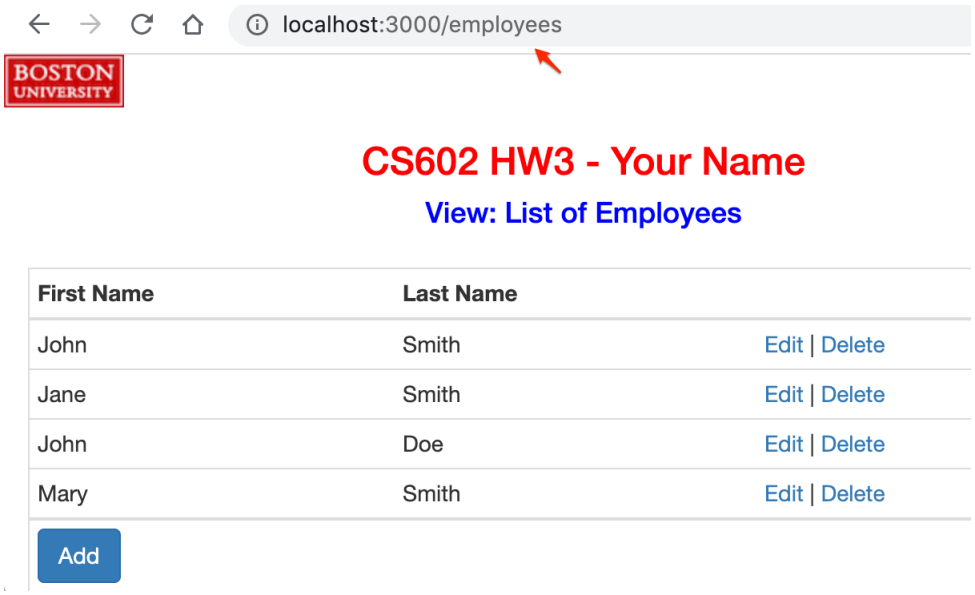
When the users submits the form with the new employee data, say, *Mary Jones*, the new employee data is persisted and the list of the employees is rendered as below:



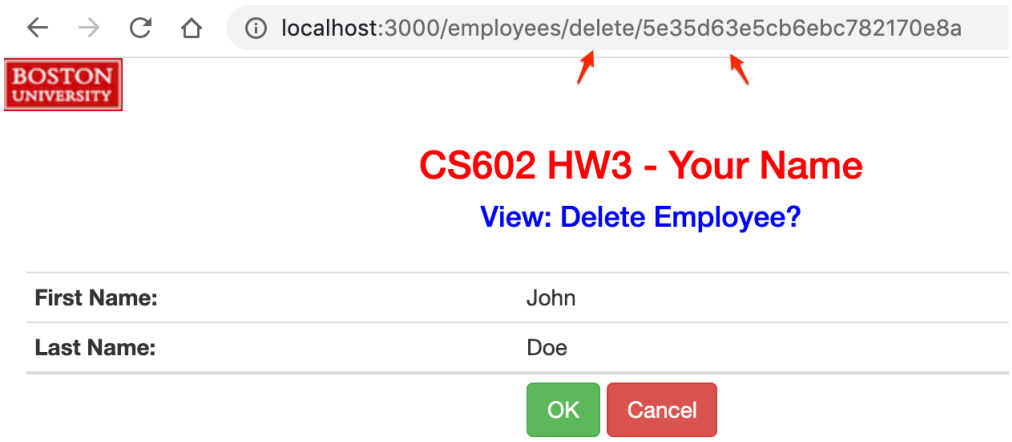
When the user clicks on *Edit*, the corresponding employee data is rendered by *editEmployeeView* as shown below:



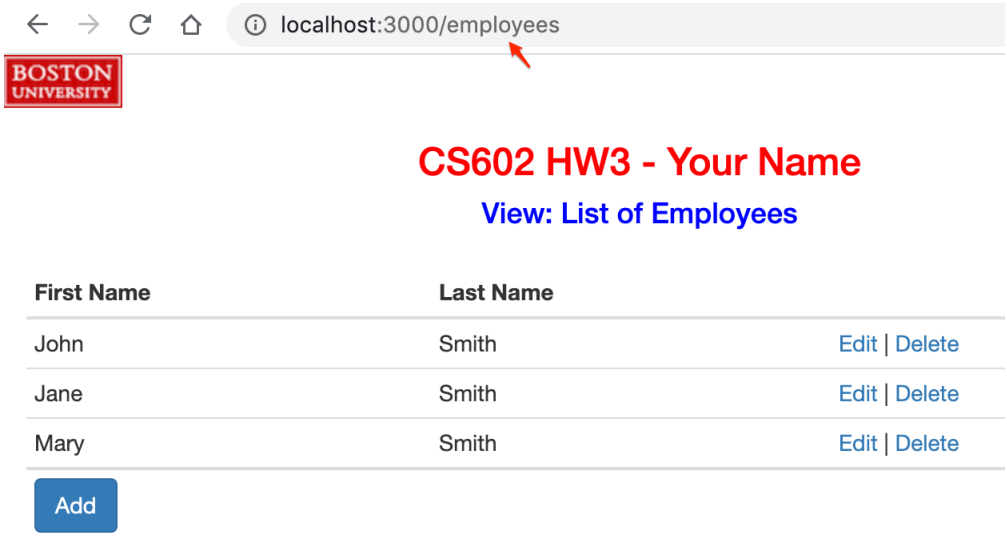
If the user changes Mary Jones to Mary Smith and clicks Save, the changes are persisted to the database and the list of the employees is rendered as below:



If the user clicks on *Delete*, a confirmation page rendered by *deleteEmployeeView* is shown:



If the user clicks OK, the corresponding data is deleted from the database and the list of the employees is rendered as below.



Submission: Export your CS602_HW3_*lastName* folder containing all the relevant files as a zip file, and upload the zip file to the Assignment section.