Tutorial 07: Node.js and Express Framework

NOTE: You may refer to sample codes shared on lecture notes for more details to create a new express project.

Task 1: Create a new Express project called "tutorial7express" from scratch

1. Create a project directory:

```
mkdir tutorial7express
```

2. Change to the new directory:

```
cd tutorial7express
```

3. Create new package.json file:

```
npm init
```

```
PS C:\Users\Chiam\Desktop\tutorial7express> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (tutorial7express)
version: (1.0.0)
description: Tutorial 7
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: Chiam
license: (ISC)
About to write to C:\Users\Chiam\Desktop\tutorial7express\package.json:

{
  "name": "tutorial7express",
  "version": "1.0.0",
  "description": "Tutorial 7",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Chiam",
  "license": "ISC"
}


Is this OK? (yes) yes
PS C:\Users\Chiam\Desktop\tutorial7express>
```

4. Create a new main app.js file:

```
fsutil file createnew app.js 0
```

(**NOTE:** `app.js` file contains the gateway to your project, it contains the global project and express settings, it also has the code to connect to your database.)

5. Install express:

```
npm install express
```

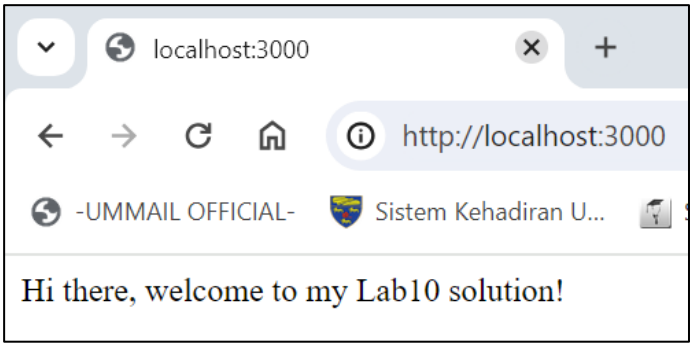
6. Edit main app.js file to:
   a. Require express and define var app to execute express as a function
   b. Define app.listen to start the server at a particular port
7. Start the web server to test the routes on browser:

```
node app.js
```

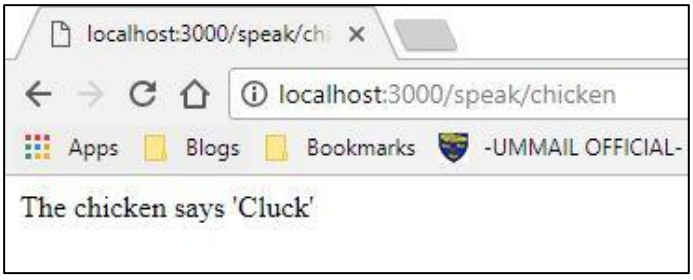
Task 2: Express Routing - Define routes to handle HTTP requests and responses

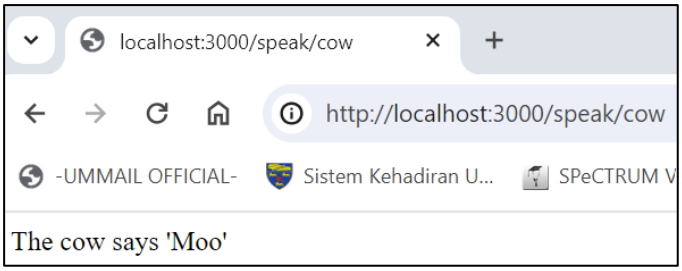
1. In your main `app.js` file, define 4 different routes using Get method:
   (NOTE: Try to apply *route matcher*, *route parameters*, and *request parameters* in your solutions.)

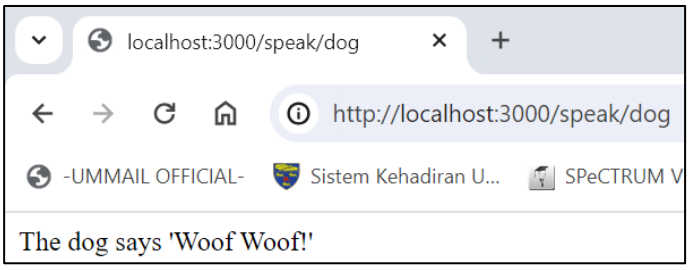
Route 1: Root "/"
Visiting "/" should print "Hi there, welcome to my Tutorial 7 solution!"



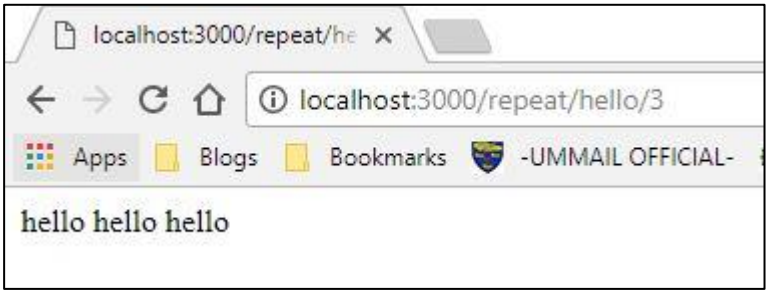
Route 2:
Visiting "/speak/chicken" should print "The chicken says 'Cluck'"
Visiting "/speak/cow" should print "The cow says 'Moo'"
Visiting "/speak/dog" should print "The dog says 'Woof Woof!'"

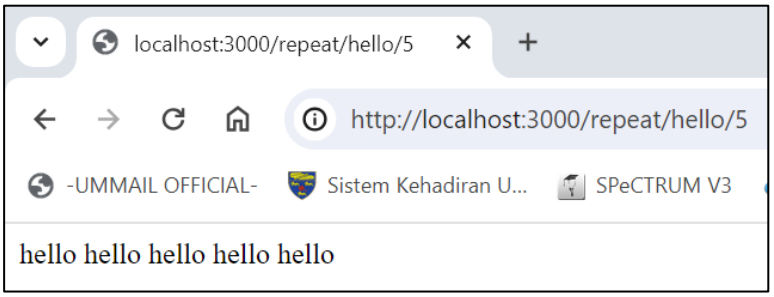


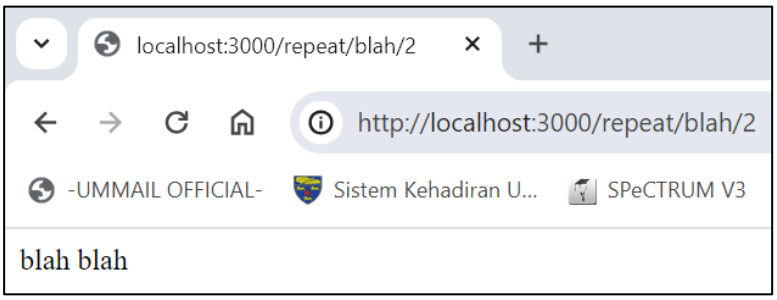




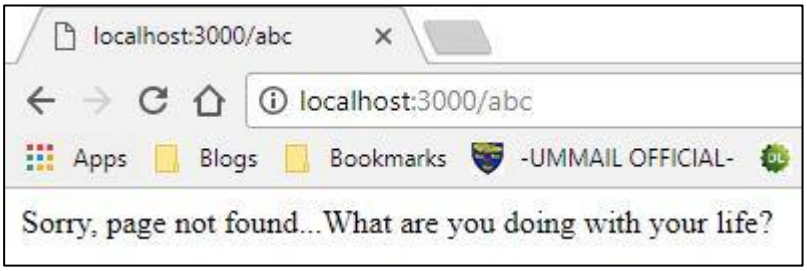
Route 3:
Visiting "/repeat/hello/3" should print "hello hello hello"
Visiting "/repeat/hello/5" should print "hello hello hello hello hello"
Visiting "/repeat/blah/2" should print "blah blah"







Route 4: If a user visits any other route, print:
"Sorry, page not found...What are you doing with your life?"



2. Start and restart your web server using `$node app.js` command to test the routes that you defined.

Task 3: Define a function

In the app.js file:

1. Write a new function named “calculateAverage” that:
    a. takes a single parameter: an array of test scores (all numbers)
    b. it should return the average score

    Example:

    - *an array of test scores = [10, 20, 30, 40, 50]*
    - *average score = 30*

2. Define a new route using GET method:
    a. Call the “calculateAverage” function by passing an array of test scores.
    b. Visiting "/average” should:
        i. Print a message on console with the average score.

```
PS C:\Users\Chiam\Desktop\tutorial7express> node app.js
Server has started!!!
Average: 30
```

        ii. Print a message on browser: “This is a request to calculate average”.