Engineering Data Intensive Scalable Systems

Project Task 1: Technology Familiarization

***********Note: You'll need to complete the setup in "AWS Account Setup Document" before starting this project******************

Problem Description

The goal of this project is to familiarize yourself with the technologies that you will be using on upcoming projects. Throughout the course, you will be incrementally developing a large robust e-commerce system. The requirements and context in which this system will be used will evolve as the course progresses. This first increment is essentially a "Hello World" application with minimal capability.

Learning Objectives

This project is designed to introduce you to the core technologies that you will be using throughout the course. By the end of this project, you should be able to install and configure the needed software, develop a simple application, deploy your application, and execute your system. These technologies include:

- Javascript
- Node JS
- MySQL
- Amazon Web Services
- Artillery

Tasks

In this project, you will build a simple system and deploy it on AWS. In addition to building a system to support the requisite functionality, you'll need to manage the compute costs for your system. You will build a web service that accepts and returns arguments in JSON format. The system will require the user to login and allow them to perform some simple calculations (e.g. 2 + 2).

The interfaces specifications for the system are:

- **Name:** Login
  - **Description:** Allows users to log into the system. This should create a session that will remain active until the user logs out or remains idle for 15 minutes. The user should be able to access any functionality that requires the user is authorized to use. You need to allow for someone to log in from a system with an active session. If there's an active session and there is a new log in (from the same system) you should end the previous session and initiate a new one. There could be multiple active sessions from multiple systems.
  - **Interface:** {BASEURI}/login
  - **HTTP Method:** Post
  - **Input parameters:**
    - { "**username**":"*Username of the person attempting to login", "**password**": "*Password of the person attempting to login"* }
  - **Return Values:**
    - Successful login – *first name* = name of the person that has logged in
      - { "**message**":"Welcome *first name*"}
    - Failure case

            - {“**message**”:“There seems to be an issue with the username/password combination that you entered”}

- **Name:** Logout
    - **Description:** If the user has an active session this method will end the session. If there is no session then there is no change of state.
    - **Interface:** {BASEURI}/logout
    - **HTTP Method:** Post
    - **Return values**
        - Successful logout
            - {“**message**”:“You have been successfully logged out”}
        - Failure case
            - {“**message**”:“You are not currently logged in”}
- **Name:** Add
    - **Description:** This function will return the sum of the two integers provided. The user must be logged in prior to accessing this function.
    - **Interface:** {BASEURI}/add
    - **HTTP Method:** Post
    - **Precondition:** must be logged in as a user to access this function
    - **Input parameters:**
        - {“**num1**”:“*first inteteger argument*”, “**num2**”: “*second integer argument”*}
    - **Return values**
        - Success Case
            - {“**message**”:“The action was successful”, “**result**”: “*num1 + num2”*}
        - Not Logged In
            - {“**message**”:“You are not currently logged in”}
        - Illegal Input
            - {“**message**”:“The numbers you entered are not valid”}
- **Name:** Divide
    - **Description:** This function will return the result of dividing num 1 by num 2. The user must be logged in prior to accessing this function.
    - **Interface:** {BASEURI}/divide
    - **HTTP Method:** Post
    - **Precondition:** must be logged in as a user to access this function
    - **Input parameters:**
        - {“**num1**”:“*first inteteger argument*”, “**num2**”: “*second non-zero integer argument”*}
    - **Return values**
        - Success Case
            - {“**message**”:“The action was successful”, “**result**”: “*num1/num2”*}
        - Not Logged In
            - {“**message**”:“You are not currently logged in”}
        - Illegal Input
            - {“**message**”:“The numbers you entered are not valid”}

Name: Multiply
    - **Description:** This function will return the product of the two integers provided. The user must be logged in prior to accessing this function.
    - **Interface:** {BASEURI}/multiply
    - **HTTP Method:** Post
    - **Precondition:** must be logged in as a user to access this function
    - **Input parameters:**
        - {“**num1**”:“*first inteteger argument*”, “**num2**”: “*second integer argument”*}
    - **Return values**
        - Success Case
            - {“**message**”:“The action was successful”, “**result**”: “*num1 * num2”*}
        - Not Logged In

        - {“**message**”:“You are not currently logged in”}
    - Illegal Input
        - {“**message**”:“The numbers you entered are not valid”}

Default users:

- Henry Smith
    - Username: hsmith
    - Password: smith
- Tim Bucktoo
    - Username: tbucktoo
    - Password: bucktoo

You should assume that multiple people will be accessing your system simultaneously and be sure that your system operates correctly when you have concurrent users. You are expected to use good programming practices with proper error handling. You will need to be sure to have the users above loaded in your system prior to submission.

You will build the web application using Node JS and Express with MySQL as the database. The system will be deployed and tested on AWS. You will have a budget of $1.00 for this project including development, testing, and instructor verification.

Technologies and Resources

You will be using Javascript, Node js, Express, MySQL, and Amazon Web Services (AWS) for this project. Links for general descriptions and resources are listed below. In the next section we will give detailed instructions on installing the technologies, getting started with AWS, and deploying an application to AWS.

General links:

- Javascript: http://www.w3schools.com/js/
- NodeJS: https://nodejs.org/
- MySQL: https://www.mysql.com/
- AWS: http://aws.amazon.com/
- Artillery: https://artillery.io/
- JSON: www.json.org

Tutorials to help you get started:

- https://www.airpair.com/javascript/node-js-tutorial
- http://nodeguide.com/beginner.html

Getting Started

Setting up an AWS Account: You’ll need to create an AWS account that will be used only for EDISS. Follow directions on the document “AWS Account Setup Instructions”. **Be sure to complete these steps before you continue with the project.**

Installing required software on your local machine:

- **Node js:** Installing node js is pretty straightforward, but differs a bit depending on your platform. You can download the software from www.nodejs.org, be sure to test that you have both node and the package manager (npm) installed by running node –v and npm –v from the command line to get the installed version.
  You can test the installation by typing: ‘express --version' to get the version of express that’s installed.
- **MySQL:** Go to http://www.mysql.com/downloads/ and download the community edition. The installation and setup is straight forward.

Setting up AWS:

Once you have an AWS account and have your developed and tested your application locally you'll need to deploy your application to the cloud. This is a two-step process. You'll need to set up an instance to run your node application and you'll need to set up your database.

- Node instance: Starting an instance on AWS is pretty easy. The directions for setting up an instance can be found here http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance_linux.html. Follow the instructions to launch the instance.
    - When you create an instance it is deployed into a Virtual Private Cloud (VPC). This is essentially a virtual firewall. As with other firewalls you'll need to create rules that will allow specific kinds of access in order to be able to connect to your instance.
    - You'll want to be sure that you can connect to your instance via port 22 in order to connect with an SSH client. You can specify a specific IP address where you'll be accessing your system from or a range of IP addresses (e.g. from your school's network). You don't want to allow unrestricted access, however, as this increases the likelihood that your account can be compromised. You can read more about setting up access rules at: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html#vpc-security-groups. In addition to opening access for an SSH connection, you will want to grant access from anywhere for the port for your application (port 3000 is the default port that node js uses). Otherwise, you won't be able to access your node application.
    - Once you create an instance on AWS you'll need to install node & express as you did on your local machine. You can connect to your instance with SSH. You can see directions here http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-connect-to-instance-linux.html.
- Database Instance: You'll now need to create a database instance on AWS. You can find directions here http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateInstance.html.
    - Once you create an instance you'll need to once again create rules to be able to connect to the database. You'll want to allow access from your machine in order to be able to configure the database. Additionally, you'll need to allow access from the EC2 instance that you created, otherwise your node application will not be able to access the database.
- Once you have the instances set up deploying your application is a matter of moving your node project to the EC2 instance and uploading your database to your db instance. Be sure to change your code so that your application is accessing the database running on AWS and not on your local machine.

Submission

For this project you'll submit:

- *The code*
- *Submit your system to the auto grader. The IP address for the autograder can be found from the course information block of the course site.*

You will submit your code and the billing report in a zip file uploaded to moodle. The default name for your file should be ***yourFullname****Project1.zip.*

Grading

The grading for this project will be:

- Correctly implemented the requirements (should gracefully handle incorrect input)
- Within budget
    - If you exceed the budget by less than 100% you will be penalized 20%
    - If you exceed the budget by 100% or more you will not get any credit for the project

Budget

The budget for this project is $1.00 (or resources that would total $1 without the free tier).