CMU 17-648 Engineering Data Intensive Scalable Systems

Project Task 2: E-Commerce System

Matthew Bass

Problem Description

In this project, we will build a basic e-commerce system. We will increase the complexity and concerns related to this system throughout the course. At first we will focus on building basic functionality, later on we will focus on systemic properties such as availability, scalability, and response time.

Learning Objectives

This project will allow you to become more familiar with the core technologies that you'll be using throughout the course.

These technologies include:

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

You might evolve your thinking about the design of your system. You may find that some of the previous decisions limit the options available to you or adversely affect one or more desirable properties. When this occurs, you should make mental note of the decision and its impact. This is a learning opportunity that you face repeatedly in the course and will want to explicitly recognize, as you'll surely have similar choices in your professional life.

Tasks

In this project, you will build a simple e-commerce 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**. The functionality will be available programmatically; you do not need to develop a user interface. The functionality should be consistent with the specifications provided.
You can assume that the user will reside in the USA (you don't need to worry about international addresses).

The requirements for the system are:

- **Name**: Register Users (does not require authentication):
    - **Description:** This allows new users to create an account. The result of this will be that customer information is in the system and the user will now have an account that will allow them to log on as a customer.
    - **Interface:** {BASEURI}/registerUser
    - **HTTP Method**: Post
    - **Input parameters**:
        - {"**fname**": "*first name", "***lname**": "*last name", "***address**": "*street address", "***city**": "*city"*, "**state**": "*2 letter state code or complete state name*", "**zip**": "*can be numbers or letters*", "**email**": "*email*", "**username**":"*Username – must be unique", "***password**": "*Password"*}
    - **Return Values**:
        - ***message***:

        - Success Case
            - {“**message**”:“*first name* was registered successfully”}
        - Illegal Input
            - {“**message**”:“The input you provided is not valid”}
            - The system should not allow duplicate registrations – this is defined as a non-unique username. For this system the username needs to be unique
            - All of the fields are required. The registration should fail if any of the fields are left blank

- **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. 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 login – *first name* = name of the person that has logged in
            - {“**message**”:“You have been successfully logged out”}
        - Failure case
            - {“**message**”:“You are not currently logged in”}
- **Name:** Update Contact Information
    - **Description:** This method allows the user to update their own contact information
    - **Preconditions:** User is a registered user and is logged into system. The parameters are optional and only updated if provided.
    - **Interface:** {BASEURI}/updateInfo
    - **HTTP Method:** Post
    - **Input parameters:**
        - {“**fname**”: “*first name”,* **“lname”**: “*last name”,* **“address”**: “*street address”,* **“city”**: “*city”*, “**state**”: “*2 letter state code or complete state name*”, “**zip**”: “*can be numbers or letters*”, “**email**”: “*email*”, “**username**”:“*Username – must be unique”,* “**password**”: “*Password”*}
    - **Return values:**
        - Success Case
            - {“**message**”:“*first name* your information was successfully updated”}
        - Not Logged In
            - {“**message**”:“You are not currently logged in”}
        - Illegal Input
            - {“**message**”:“The input you provided is not valid”}
- **Name**: Add Products (must be logged in as an admin):
    - **Description:** This allows admin to add a product to the system. The result of this will be that the product is now in the system. All parameters are required.

  - **Interface:** {BASEURI}/addProducts
  - **HTTP Method**: Post
  - **Input parameters**:
    - {"**asin**": "*a unique id that can be used to access this product and associate with related information (such as reviews)*", "**productName**": "*the name of the product*", "**productDescription**": "*a description of the product*", "**group**", "*the group(s) the product belongs to, examples below*"}
    - Example groups are (you should allow any group provided):
      - Book
      - DVD
      - Music
      - Electronics
      - Home
      - Beauty
      - Toys
      - Clothing
      - Sports
      - Automotive
      - Handmade
  - **Return Values**:
    - Success Case
      - {"**message**":"*productName* was successfully added to the system"}
    - Not Logged In
      - {"**message**":"You are not currently logged in"}
    - Not admin
      - {"**message**":"You must be an admin to perform this action"}
    - Illegal Input
      - {"**message**":"The input you provided is not valid"}
        - The ASIN must be unique/or required fields not included

- **Name:** Modify Products (only an admin can modify a product):
  - **Description:** This method allows you to modify the description and name of the product. All parameters are required.
  - **Preconditions:** The user is an admin and is logged into the system
  - **Interface:** {BASEURI}/modifyProduct
  - **HTTP Method:** Post
  - **Input parameters:**
    - {"**asin**": "*cannot be modified, only used to reference the product",* "**productName**": "*the name of the product*", "**productDescription**": "*a description of the product*", "**group**", "*the group(s) the product belongs to, examples below*"}
  - **Return values:**
    - Success Case
      - {"**message**":" *productName* was successfully updated"}
    - Not Logged In
      - {"**message**":"You are not currently logged in"}
    - Not admin
      - {"**message**":"You must be an admin to perform this action"}
    - Illegal Input
      - {"**message**":"The input you provided is not valid"}
        - The ASIN must be unique/or required fields not included

- **Name:** View Users
  - **Description:** This method allows the user to view all of the users registered in the system. . If the search criteria is blank the system will return all users.
  - **Preconditions:** The user is an admin and is logged into the system
  - **Interface:** {BASEURL}/viewUsers

        - **HTTP Method:** Post
        - **Input parameters (optional parameter to filter results):**
            - **{“fname”: “***some portion (or all) of the first name of the users you’d like to view*”, “**lname**”: “*some portion (or all) of the last name of the users you’d like to view*”}
        - **Return values:**
            - Success
                - {“**message**”:“The action was successful”, “**user**”:[{“**fname**”: “*first name*”, “**lname**”: “*last name*”, “**userId**”: “*a unique id for this user*”}, {“**fname**”: “*first name*”, “**lname**”: “*last name*”, “**userId**”: “*a unique id for this user*”}, …]}
            - No Users
                - {**“message”:** “There are no users that match that criteria”}
            - Not Logged In
                - {“**message**”:“You are not currently logged in”}
            - Not admin
                - {“**message**”:“You must be an admin to perform this action”}
- **Name:** View Products (does not require log in):
    - **Description:** This method returns products that meet the search criteria. If the search criteria is blank the system will return all products.
    - **Interface:** {BASEURI}/viewProducts
    - **HTTP Method:** Post
    - **Input parameters (optional)**
        - {“**asin**”: “*id of the product that you’re interested in viewing (if you have this),* “**keyword**”: “*if you’d like you can search for products via a keyword. The product should be returned if the keyword is contained in the name or description of the product*”: “**group**”, “*only products in this category should be returned*”}
    - **Return values**
        - Success
            - **{“product”:[** {“**asin**”: “*The id of the product retrieved”,* “**productName**”: “*the name of the product*”}, {“**asin**”: “*The id of the product retrieved”,* “**productName**”: “*the name of the product*”}, …]}
        - No Products
            - {**“message”:** “There are no products that match that criteria”}

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.

Default administrator to have in your system at the time of testing:
- Jenny Admin
    - Username: jadmin
    - Password: admin

*** Your database should not have any products or users other than the admin listed above ***

Technologies and Resources

You will be using Javascript, Node js, Express, MySQL (or the data store of your choice), 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: http://artillery.io
- JSON: www.json.org

Submission

For this project you'll submit:

- *The code to the course site*
- *Submit your system to the auto grader. The link for the autograder can be found in the course resources block on the course website.*

**** You will only able to submit your system to the autograder one time. You'll need to do any testing prior to submission ****

You will submit your code and the billing report in a zip file uploaded to the course site. The default name for your file should be ***yourID****Project2.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

You will have a budget of $1.00 for this project. This does not include development and testing costs. Costs will be evaluated based on the services used when you submit your system to the autograder.