Task 8 Specification

The response format is in JSON. You'll need to structure the responses exactly as specified or the test cases will fail. In addition to the specification an Artillery sample test script has been provided to help you generate a more robust test script. Keep in mind the provided test script only partially tests your system (and API). It would be prudent to create a more robust test script prior to submission.

Email the instructor to get your AWS coupon code worth $50 of AWS credits.

Note: You need to make sure that your system is pre-loaded with ****only**** the following employee in order to facilitate testing. No other user information can be in your database, otherwise your test cases might fail.

First Name: Jane

Last Name: Admin

Username: jadmin

Password: admin

Address: 123 Main street

City: Pittsburgh

State: Pa

Zip: 15143

Role: Employee

Number Formats: All shares should be in whole numbers. So 123 is acceptable. 234.00 or 234.5 is ***not*** acceptable. All cash values should not contain any commas, dollar sign, or other formatting characters. It can contain up to two decimal places, however. This means that a valid cash value could be 2889.50, 2889.5, or 2889. The following are ***not*** acceptable: $3,456, 4,509.50, and so forth. Both will be passed as a string in JSON {"cash": "123.45"}, ***not*** {"cash": 123.45}.

HTTP Status Codes: We will be using the following HTTP Status Codes (and no others):

- 200: Success
- 400: Malformed request (couldn't be understood by the server). This includes missing required content.
- 403: Forbidden request. Understood by the server, but the server refuses to fulfill the request
- 404: Not Found: The server was not able to find an endpoint matching the URI
- 500: Internal server error: there was an unexpected condition that prevents the server from fulfilling the request

Employee Use Cases:

- **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 that already has 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 (all required):**
    - { "**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"}
- Create Customer Account
  - **Description:** This method creates a new customer account. The system should check to ensure that all of the parameters have values and the format is appropriate (cash is optional. If there is no value for cash the default is 0). The system should not allow two accounts with the same username.
  - **Interface:** {BASEURI}/createCustomerAccount
  - **HTTP Method:** Post
  - **Input parameters(all required):**
    - {"**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*", "**cash**": "*value*", "**username**":"*Username – must be unique",* "**password**": "*Password"*}
  - **Return Values**:
    - Success Case
      - {"**message**":"*fname* was registered successfully"}
    - Duplicate username
      - {"**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
            - Not Logged In
                - {"**message**":"You are not currently logged in"}
            - Not employee
                - {"**message**":"You must be an employee to perform this action"}
- Deposit Check
    - **Description:** This method adds money to the customer's account. To simplify matters it requires the username as the unique identifier for the account (rather than having to fetch the customer id). The system should ensure that the format of the input is correct. The detonation is assumed to be US dollars.
    - **Interface:** {BASEURI}/depositCheck
    - **HTTP Method:** Post
    - **Prerequisites:** The user needs to be an employee and be actively logged into the system. If they are not logged in they should get a message.
    - **Input Parameters (all required):**
        - **{"username": "***username***", "cash": "***value***"}**
    - **Return Values:**
        - Success Case
            - {"**message**":"The check was successfully deposited"}
        - Not Logged In
            - {"**message**":"You are not currently logged in"}
        - Not admin
            - {"**message**":"You must be an employee to perform this action"}

- Create Fund
    - **Description:** This method allows the employee to add a new fund to the system. The system should also record the date the fund was created. The price for the fund should be recalculated every time the "Transition Day" function is called.
    - **Interface:** {BASEURI}/createFund
    - **HTTP Method:** Post
    - **Prerequisites:** The user needs to be an employee and be actively logged into the system. If they are not logged in they should get a message.
    - **Input Parameters (all required):**
        - **{"name"**: *"name"*, **"symbol"**: *"symbol"*, **"initial_value"**: *"initial_value"*}
    - **Return Values:**
        - Success Case
            - {"**message**":"The fund was successfully created"}
        - Not Logged In
            - {"**message**":"You are not currently logged in"}
        - Not admin
            - {"**message**":"You must be an employee to perform this action"}
- Transition Day

- **Description:** This method will cause the values of the funds to fluctuate. When the method is called the price of the funds will be recalculated by the system. They will go up or down some random amount within +-10% of their current value. The price should always be greater than 0.
- **Interface:** {BASEURI}/transitionDay
- **HTTP Method:** Post
- **Prerequisites:** The user needs to be an employee and be actively logged into the system. If they are not logged in they should get a message.
- **Input Parameters: None**
- **Return Values:**
  - Success Case
    - {"**message**":"The fund prices have been successfully recalculated"}
  - Not Logged In
    - {"**message**":"You are not currently logged in"}
  - Not admin
    - {"**message**":"You must be an employee to perform this action"}

Customer Use Cases:

- **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 that already has 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 (all required):**
    - { "**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"}

- View Portfolio
    - **Description:** This method allows users to see a list of funds that they currently own.
    - **Interface:** {BASEURI}/viewPortfolio
    - **HTTP Method:** Get
    - **Prerequisites:** The user must be logged in and be a customer in order to view their portfolio.
    - **Input Parameters:**
        - **None needed. The user specific information should be available in the active session**
    - **Return Values:**
        - Success Case
            - **{"message":** *"*The action was successful*",* "**cash**": "*current balance in account*", "**funds**":[{"**name**":*"fund1*", "**shares**": "**numShares**", "**price**":*"currentFundPrice*"}, {"**name**":*"fund2*", "**shares**": "*numShares*", "**price**":*"currentFundPrice*"}, …]}
        - No Funds
            - {"**message**": "You don't have any funds in your Portfolio"}
        - Not Logged In
            - {"**message**":"You are not currently logged in"}
        - Not customer
            - {"**message**":"You must be a customer to perform this action"}

- Buy Fund
    - **Description:** This method allows the users to spend a specified amount on an existing fund at the current price. The system will determine how many shares they can buy (cannot buy partial shares). If they have sufficient funds to cover the purchase of at least one share their account will be debited accordingly and they will now have these funds in their portfolio.
    - **Interface:** {BASEURI}/buyFund
    - **HTTP Method:** Post
    - **Prerequisites:** The user must be logged in, be a customer, and have sufficient funds in their account in order to buy the requested funds.
    - **Input Parameters (all required):**
        - **{"symbol":** *"fundSymbol"***, "cashValue",** *"cashValue"***}**
    - **Return Values:**
        - Success Case
            - {"**message**":"The fund has been successfully purchased"}
        - Not Logged In
            - {"**message**":"You are not currently logged in"}
        - Not enough cash in account
            - {"**message**":"You don't have enough cash in your account to make this purchase"}
        - Not enough cash provided
            - {"**message**":"You didn't provide enough cash to make this purchase"}
        - Fund doesn't exist

        - {“**message**”:”The fund you provided does not exist”}
    - Not customer
        - {“**message**”:“You must be a customer to perform this action”}

- Sell Fund
    - **Description:** This method allows the user to sell one or more shares of a fund that they own. When they sell a fund the current value of that fund will be deposited into their “cash” account.
    - **Interface:** {BASEURI}/sellFund
    - **HTTP Method:** Post
    - **Prerequisites:** The user must be logged in, be a customer, and own funds in order to sell funds in their portfolio.
    - **Input Parameters (all required):**
        - **{“symbol”:** *“fundSymbol”***, “numShares”,** *“numShares”***}**
    - **Return Values:**
        - Success Case
            - {“**message**”:“The shares have been successfully sold”}
        - Not Logged In
            - {“**message**”:“You are not currently logged in”}
        - Not enough shares
            - {“**message**”:“You don’t have that many shares in your portfolio”}
        - Fund doesn’t exist
            - {“**message**”:”The fund you provided does not exist”}
        - Not customer
            - {“**message**”:“You must be a customer to perform this action”}
- Request Check
    - **Description:** This method allows users to withdraw money from their account. If they have sufficient balance the balance will be deducted. If not, they will get an error message.
    - **Interface:** {BASEURI}/requestCheck
    - **HTTP Method:** Post
    - **Prerequisites:** The user must be logged in, be a customer, and have a positive balance greater than the value of the check in order to withdraw money
    - **Input Parameters (all required):**
        - **{“cashValue”:** *“cashValue”***}**
    - **Return Values:**
        - Success Case
            - {“**message**”:“The check has been successfully requested”}
        - Not Logged In
            - {“**message**”:“You are not currently logged in”}
        - Not enough cash
            - {“**message**”:“You don’t have sufficient funds in your account to cover the requested check”}
        - Not customer
            - {“**message**”:“You must be a customer to perform this action”}