Web Application

RESTful API Endpoints To Be Implemented

About the field data types in swagger docs:

1. If a field has readOnly set to true, the value in this field is populated by your application. Example fields are timestamp and id. readOnly properties are included in responses but not in requests.
2. If a field has writeOnly set to true, the value in this field is provided by the API caller in POST or PUT request but these fields are not part of GET request. Example is the password field. writeOnly properties may be sent in requests but not in responses.
3. multipleOf keyword is used to specify that a number must be the multiple of another number.
4. The minimum and maximum keywords are used to specify the range of possible values.

API Documentation: <https://app.swaggerhub.com/apis-docs/spring2022-csye6225/app/a02>

User Stories

1. All API request/response payloads should be in JSON.
2. No UI should be implemented for the application.
3. As a user, I expect all APIs calls to return with a proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
4. As a user, I expect the code quality of the application is maintained to the highest standards using the unit and/or integration tests.
5. Your web application must only support [Token-Based authentication and not Session Authentication](https://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication).
6. As a user, I must provide a [basic](https://en.wikipedia.org/wiki/Basic_access_authentication#Client_side) [authentication](https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication) token when making an API call to the authenticated endpoint.
7. Create a new user
   1. As a user, I want to create an account by providing the following information.
      1. Email Address
      2. Password
      3. First Name
      4. Last Name
   2. account\_created field for the user should be set to the current time when user creation is successful.
   3. User should not be able to set values for account\_created and account\_updated. Any value provided for these fields must be ignored.
   4. Password field should never be returned in the response payload.
   5. As a user, I expect to use my *email address* as my *username*.
   6. Application must return 400 Bad Request HTTP response code when a user account with the email address already exists.
   7. As a user, I expect my password to be stored securely using [BCrypt password hashing scheme](https://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html" \o "https://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html" \t "_blank) with [salt](https://en.wikipedia.org/wiki/Salt_(cryptography)).
8. Update user information
   1. As a user, I want to update my account information. I should only be allowed to update the following fields.
      1. First Name
      2. Last Name
      3. Password
   2. Attempt to update any other field should return 400 Bad Request HTTP response code.
   3. account\_updated field for the user should be updated when the user update is successful.
   4. A user can only update their own account information.
9. Get user information
   1. As a user, I want to get my account information. Response payload should return all fields for the user except for password.

AWS Organization Setup

1. Enable support for organizations in your AWS account. This is your root account.
2. Create a dev member account. You will use this account for assignment development. You can use an alias for your personal Gmail email address. For e.g., if your email address is j.doe@gmail.com, you can create an alias such as j.doe+dev@gmail.com.
3. Create a demo member account. You will use this account to demo assignments for grading. You can use an alias for your personal Gmail email address. For e.g., if your email address is j.doe@gmail.com, you can create an alias such as j.doe+demo@gmail.com.

AWS IAM Setup

*No one except you should be able to modify resources in your environment.*

Create [IAM](https://aws.amazon.com/iam/) users, and group in all (root, dev, prod) accounts.

Create Group

1. Create csye6225-ta group for all teaching assistants.
2. The group should have read-only access to AWS services and resources in your account. Search for policy pre-defined read-only policy [arn:aws:iam::aws:policy/ReadOnlyAccess](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/policies/arn:aws:iam::aws:policy/ReadOnlyAccess%24serviceLevelSummary" \t "_blank) and attach it to the group.

Create Users

1. Create a user account for all teaching assistants.
2. Do not create one for the instructor.
3. Use the first name as the username. Do not configure credentials for the users. Leave the default setting *Autogenerated password* checked and copy the generated password. **Autogenerated passwords are not emailed out by AWS. You need to manually send the email with the password.**

Documentation

Web App

* [RESTful API Authentication Basics](https://dzone.com/articles/restful-api-authentication-basics-1)
* <https://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication>
* <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>

Git

* [Pro Git](https://git-scm.com/book/en/v2)
* [Learn Git](https://www.visualstudio.com/learn-git/)
* [Understanding the GitHub Flow](https://guides.github.com/introduction/flow/)
* [Forking Workflow](https://www.atlassian.com/git/tutorials/comparing-workflows#forking-workflow)
* [A collection of useful .gitignore templates](https://github.com/github/gitignore)

AWS Organizations

* [AWS Organizations](https://aws.amazon.com/organizations/)
* [What Is AWS Organizations?](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_introduction.html)

IAM

* [Amazon Resource Names (ARNs)](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html)
* [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/)
* [IAM Actions, Resources, and Condition Keys for AWS Services](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_actions-resources-contextkeys.html)
* [IAM JSON Policy Elements Reference](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html)

Submission

The assignment will be considered late if commits are made to the **main** branch after the due date.

1. All work for this assignment must be done on the feature branch in your fork and merged to main when you are dev complete.
2. The feature and main branches must be in sync.

Grading Guidelines

The following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

Web Application Crash (10% Penalty)

* No 500 internal server errors.
* No restart of application server between API calls.

Web Application (50%)

* Students to demo the web application from their laptops.
* Verify passwords are encrypted with BCrypt hashing and salt in the database.
* Verify that authentication is done via basic auth (token-based) and not session-based.
* APIs can be demoed using any Postman or Restlet or some other REST client but not via browser.
* Check for UI. The application cannot have UI.
* Check the response payload to make sure it meets the assignment objective. Password field should not be part of the response payload.
* Test for duplicate account creation in the application.
  + The application should NOT allow multiple accounts with the same email address.
* Test updating fields such as account\_created and account\_updated.
  + Users should never be able to set values for them. These fields are set by the application.
* Verify non-email username cannot be used for account creation.

Git (20%)

* All students must use the Github forking workflow and their repositories (main branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull requests between their main branch and their assignment branch. There should be no changes. Verify that all assignment changes are in main branch.
* Verify that students have added TAs as collaborators to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for a programming language, frameworks, and third-party libraries.
* Verify that the dev environment is not set up in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https.
  + Verify this by running git remote -v command in the cloned repository in the VM.
* Validate that students have created a fork of the organization repository are working on it.
* Verify that the student has made no direct commits to their organization repository.

Git Repository Content Check (10%)

* Check the repository for any IDE-specific files. IDE configuration files must not be in the repository.
  + Verify their .gitignore configuration.
* Check the repository for build artifacts such as .class, .jar, .war files and build, node\_modules directory. None of these should be checked into the repository.
* Check for dependencies. Dependencies from the Maven repository or npm should not be committed to the git repository.

IAM (10%)

* Check if all users (team and TAs) are created and assigned to the correct group.
* Verify users have console access and not programmatic access.
  + For this check the email students have sent to you. Did the email contain a password or credentials for programmatic access or just a username?
* Check the policy attached to the group is [arn:aws:iam::aws:policy/ReadOnlyAccess](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/policies/arn:aws:iam::aws:policy/ReadOnlyAccess%24serviceLevelSummary" \t "_blank).
* Verify that you (TA) have received both username and password from the student for the IAM console. Log in and verify that you can access their AWS account and your account can only READ.
* Verify that the student has MFA enabled on member accounts and the organization root account.

AWS Organizations (*10%*)

* Validate that student has set up an organization and created 2 member accounts and that they can log in to them.
* Remind students to set up multi-factor authentication for member accounts as well.