Assignment #01

Due Date

Section	Due Date	Grading Deadline
Saturday & Online	06:00pm on 06/07/2019	06:00pm on 06/15/2019

Getting Help

Info

Ask all your questions on Piazza. Assign **assignment1** tag to your posts.

Assignment Weightage

Individual Assignment Weightage on Course Grade of this assignment is **10%**.

Objectives

Team Formation

You can advertise yourself on this Piazza note [https://piazza.com/class/juvloj5ns234x3?cid=5] to find team members. Be very clear about the course section you are enrolled in so that students from same

section can find you. Individual posts on Piazza looking for team members will be deleted.

Setup GitHub Repository

1. Create a GitHub repository for assignments [https://help.GitHub.com/articles/create-a-repo/]. This must be a private repository that only your team, TAs and instructor can access. Make sure to create empty repository. Just like we learned in the lab, one member should create the repo and other's should fork this repo.
2. GitHub repository name must be **ccwebapp**.
3. Add all TAs and me to your GitHub repository as collaborators. Our emails and GitHub IDs can be found on home page.
4. Set up your repository by importing code from template [https://github.com/tejasparikh/csye6225-summer2019-template].
5. Update **README.md** in your repository. Your readme file must contain following:
    a. Team member information such as Name and Email address.
    b. Prerequisites for building and deploying your application locally.
    c. Build and Deploy instructions for web application.

Web Application Development

> **Danger**
>
> Instructor and teaching assistants will not be able to answer programming or framework related questions for the web applications you are building. We will gladly assist you with AWS & GCP SKDs.

Objective of this assignment is to start implementing APIs for the *Library Management System*. See Cloud Native Web Application Requirements [../../cna/] for technical requirements. We will build backend (APIs only, no UI) for a *Library Management System* application. We will implement features of the *Library Management System* through various assignments in this course.

RESTful API Endpoints To Be Implemented

HTTP Method	Endpoint	Authenticated Endpoint	Description
GET	`/`	Yes	Get current time
POST	`/user/register`	No	Create account for user
POST	`/book`	Yes	Create book
GET	`/book`	Yes	Get all books
GET	`/book/{id}`	Yes	Get a books
DELETE	`/book/{id}`	Yes	Delete a book
PUT	`/book`	Yes	Update book information

API Specifications

API specifications [https://app.swaggerhub.com/apis-docs/csye6225/csye6225-summer2019/1.0.0]

User Stories

1. As a user, I want to create an account by providing following information.
    a. Email Address
    b. Password
2. As a user, I expect to use my *email address* as my *username*.
3. As a user, I want to be able to make `GET` requests to `/`.
    a. I expect the API to respond with `current time` if I provide the basic authentication headers.
    b. If I do not provide basic [https://en.wikipedia.org/wiki/Basic_access_authentication#Client_side]

authentication [https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication] headers, application can return a message saying I am not logged in.

4. Your web application must only support Token-Based authentication and not Session Authentication [https://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication].
5. As a user, if I try to create an account and it already exists, system should warn me that account already exists.
6. 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] with salt [https://en.wikipedia.org/wiki/Salt_(cryptography)].
7. As a user, I expect the code quality of the application is maintained to highest standards using unit and/or integration tests.
8. All API request and response payloads should be in JSON.
9. No UI should be implemented for the application.
10. As a user, I expect all APIs call to return with proper HTTP status code [https://en.wikipedia.org/wiki/List_of_HTTP_status_codes].
11. As a user, I want to create a new book with `POST` request to `/book` endpoint.
12. As a user, I want to delete a book with `DELETE` request to `/book/{id}` endpoint.
13. As a user, I want to update a book with `PUT` request to `/book` endpoint.
14. As a user, I want to get a book with `GET` request to `/book/{id}` endpoint.
15. As a user, I want to get all books with `GET` request to `/book` endpoint.

Documentation

- HTTP Request Methods [https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods]

- https://stackoverflow.com/questions/19332821/http-basic-authentication-over-ssl-for-rest-api [https://stackoverflow.com/questions/19332821/http-basic-authentication-over-ssl-for-rest-api]
- https://en.wikipedia.org/wiki/Basic_access_authentication [https://en.wikipedia.org/wiki/Basic_access_authentication]
- 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://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication]
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes [https://en.wikipedia.org/wiki/List_of_HTTP_status_codes]

Submission

Danger

Assignment will be considered late if commits are made to `master` and `feature` branch after due date.

1. All work for this assignment must be done on **assignment1** feature branch and merged to master when you are dev complete.
2. All team member's feature and master branches must be in-sync.

Grading Guidelines

Warning

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

Web Application *(60%)*

- Students will demo the web application from their laptop.
- Verify passwords are encrypted with BCrypt hashing and salt in the database.
    - Run SQL queries to check data from user table.
- 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. Application cannot have UI.
- Check the response payload for sensitive information. User's data such as email, password or even password hash should never be part of response payload.
- Check for duplicate account handling in application.
- Verify non-email username cannot be used for account creation.
- Verify that weak passwords cannot be used to create accounts.
    - Check for password length of 8 or shorter which should be rejected.
    - Check for simple all char passwords.
    - Check for complex passwords.
- Verify the `/book` and `book/{id}` endpoints meet the specification. Check request, response, headers, and return status code.
- Verify basic authentication credentials are passed in header of the request and not in the body of the request.
- Request and response payloads must be in JSON. Check payload is valid JSON.

Git (40%)

- All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync.

Check this by asking students to create pull request between their `master` branch and their scrum master's `master` branch. There should be no changes. Verify that all assignment changes are in `master` branch.

- Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets `0` points for the whole assignment.
- Added TAs and instructor as collaborator 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 programming language, frameworks and third-party libraries.
- Verify that dev environment is not setup 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.
- Ask students to perform `git pull` from scrum master's repo and run the `git status` command. This must be done from terminal.