CS F213: Object Oriented Programming

Major Assignment

Spring 2024

List of Clarifications sought on 29th April:

- Sign up: only the email needs to be unique, the name can be repeated. This change will be reflected in the postman collections in sometime
- Negative room capacity: this has been fixed in postman
- Duplicate room names are not allowed in postman now
- Seconds have been included in time in postman
- Room bookings in the past can't be made. The Start Time, in addition to the day, is to be considered for history/upcoming. Anything which has started before 'now' is history and anything starting after 'now' is upcoming
- User (with the userID as the only attribute) in the Room object to be returned, is an object
- roomName is to be included in /room. The doc Now reflects this

---

- **Please check back this document for updates and edits. Updates and edits will be made in red.**
- **Deadline: 26th April, 2024, 23:59 hours**
- This assignment is to be done individually
- You have to solve one of the two problems described below:
  - Social Media Platform (Twitter)
  - Room Booking System
- You have to use SpringBoot to create the REST APIs as described under each problem statement
- You need to use the H2 database to store any persistent data. Please use this properties file. *Do not change the username, password and DB name in the properties file, or your submission will not run on our test platform*. You may add other configuration parameters if required.
- The base URL for the application is: http://127.0.0.1:8080
- You are advised to use Intellij IDEA for development and Postman for API Testing

- What to submit:
    - You will be required to submit a jar of your application
    - And, the source code for the application
    - Submissions will have to be done on CMS. Links will be created soon
- Evaluation:
    - The source code will be compiled to check that it generates the same jar that you submitted. If it does not, your submission will be rejected
    - The app will be run on port 8080 and some test API calls will be made such that at a given instant it will return a unique response
    - You will be marked for the number of test cases correctly passed and the quality of your app design
- Please note:
    - All requests and responses are in JSON format
    - All dates are in the yyyy-MM-dd format
    - All times are in the HH:mm:ss format (24 hours format)
- Rules and penalty for plagiarism remain the same as the other assignments

Problem 1: Social Media Platform (Twitter)

Postman Collection to test your app:
https://bold-robot-635964.postman.co/workspace/My-Workspace~fdcf320c-d95e-4a0f-8014-7f96e0ac11bd/collection/24752303-7dc37d1a-2404-4a0a-8ecf-b33facb1a9b9?action=share&creator=24752303

LOGIN: Endpoint to login an existing user. If the user does not exist or the credentials are incorrect, relevant error is thrown
URL: /login
Method: POST
Request Body: email <str>
password <str>
Response:<One of the following Messages>
- Login Successful
- Username/Password Incorrect
- User does not exist

SIGNUP: Endpoint to register a new user. If the account already exists, relevant error should be returned
URL: /signup

Method: POST
Request Body: email <str>
name <str>
password <str>
Response: <One of the following Messages>

- Account Creation Successful
- Forbidden, Account already exists

USER DETAIL: Endpoint to retrieve details about a user. Relevant error is returned if the user does not exist
URL: /user
Query Parameter: userID
Method: GET
Response Body: One of

- name <str>
  userID <int>
  email <str>

- User does not exist

USER FEED: Endpoint to retrieve all posts by all users in reverse chronological order of creation
URL: /
Method: GET
(All posts by all users sorted in the order of creation, the latest at the top)
Response Body: posts <object>
```
                    -postID <int>
                    -postBody <string>
                    -date<date>
                    -comments <object>
                          -commentID<int>
                          -commentBody<string>
                          -commentCreator<Object>
                               -userID<int>
                               -name<str>
```

Endpoints related to a POST
URL: /post

To create a new post. Return relevant error if the user does not exist
Method: POST

Request Body: postBody<string>
userID<int>
Response: One of:

- Post created successfully
- User does not exist

To retrieve an existing post. Relevant error needs to be returned if the post does not exist
Method: GET
Query Parameter: postID
(All the details about a specific post)
Response Body: One of:

- <objects>
  -postID <int>
  -postBody <string>
  -date<date>
  -comments <object>
  -commentID<int>
  -commentBody<string>
  -commentCreator<Object>
  -userID<int>
  -name<str>
- Post does not exist

To edit an existing post. Relevant error needs to be returned if the post does not exist
Method: PATCH
Request Body:
postBody<str>
postID<int>
Response Body: One of

- Post edited successfully
- Post does not exist

To delete a post. Relevant error needs to be returned if the post does not exist
Method: DELETE
Query Parameter: postID<int>
Response: One of:

- Post deleted
- Post does not exist

Endpoints related to comments on a post
URL: /comment

To create a new comment. Relevant error needs to be returned if the post does not exist
Method: POST
Request Body: commentBody<string>
postID<int>
userID<int>
Response: One of:
- Comment created successfully
- *User does not exist*
- *Post does not exist*

To retrieve an existing comment. Relevant error needs to be returned if the comment does not exist
Method: GET
Request Param: CommentID<int>
Response Body: One of:
- <object>
    - -commentID<int>
    - -commentBody<string>
    - -commentCreator<Object>
        - -userID<int>
        - -name<str>
- Comment does not exist

To edit an existing comment. Relevant error needs to be returned if the comment does not exist
Method: PATCH
Request Body:
commentBody<str>
commentID<int>
Response Body: One of
- Comment edited successfully
- Comment does not exist

To delete an existing comment. Relevant error needs to be returned if the comment does not exist
Method: DELETE
Query Parameter: commentID<int>
Response Body: One of
- Comment deleted
- Comment does not exist

ALL USERS: Endpoint to show details about all the existing users
URL: /users

Method: GET
Response:<Object>

-name <str>
-userID <int>
-email <str>
~~-posts <object>~~

Problem 2: Room Booking Portal

Postman Collection to test your app:
https://bold-robot-635964.postman.co/workspace/My-Workspace~fdcf320c-d95e-4a0f-8014-7f96e0ac11bd/collection/24752303-465777c8-3451-4854-a925-9bd2b05879b8?action=share&creator=24752303

LOGIN: Endpoint to login an existing user. If the user does not exist or the credentials are incorrect, a relevant error is thrown
URL: /login
Method: POST
Body: email <str>
password <str>
Response:<One of the following Messages>

- Login Successful
- Username/Password Incorrect
- User does not exist

SIGNUP: Endpoint to register a new user. If the account already exists, relevant error should be returned
URL: /signup
Method: POST
Request Body: email <str>
name <str>
password <str>
Response: <One of the following Messages>

- Account Creation Successful
- Forbidden, Account already exists

USER DETAIL: Endpoint to retrieve details about a user. Relevant error is returned if the user does not exist
URL: /user
Query Parameter: userID
Method: GET
Response Body: One of:

- name <str>
  userID <int>

email <str>
- User does not exist

ROOM BOOKING HISTORY: Endpoint to retrieve the room booking history for a particular user
URL: /history
Query Parameter: userID
Method: GET
Response Body: <objects>
-roomName <str>
-roomID<int>
-bookingID<int>
-dateOfBooking<date>
-timeFrom<str>
- timeTo<str>
-purpose<str>
Error:
~~Room does not exist~~
User does not exist

UPCOMING ROOM BOOKING: Endpoint to retrieve the upcoming room bookings for a particular user
URL: /upcoming
Query Parameter: userID
Method: GET
Response Body: <object>
-roomName <str>
-roomID<int>
-bookingID<int>
-dateOfBooking<date>
-timeFrom<str>
- timeTo<str>
-purpose<str>
Error:
User does not exist

ROOMS: Endpoints related to rooms
URL: /rooms

Precedence order for errors: existence of room > duplicate room name > invalid capacity

Method: GET
Query Parameters: Filter on ~~Date, time,~~ capacity [Optional Parameter: If the capacity parameter is not specified, then all the rooms should be returned in the response.]

Response Body: room <object>
```
                -roomID<int>
                -roomName<str>
                -capacity<int>
                -booked<object>
                     -bookingID<int>
                     -dateOfBooking<date>
                     -timeFrom<str>
                     -timeTo<str>
                     -purpose<str>
                     -user<object>
                          -userID<int>
```
Error:
Invalid parameters

To add a new room:
Method: POST
Request Body:
roomName<str>
roomCapacity<~~unsigned~~ int> (capacity should not be less than or equal to 0)
Response: One of:
- Room created successfully
- Room already exists
- Invalid capacity

To edit a room:
Method: PATCH
Request Body:
roomID<int>
roomName<str>
roomCapacity<unsigned int>
Response: One of:
- Room edited successfully
- Room does not exist
- Room with given name already exists
- Invalid capacity

To delete a room:
Method: DELETE
Query Parameter: roomID<int>
Response: One of:
- Room deleted successfully
- Room does not exist

BOOKING: Endpoints related to bookings
URL: /book

Precedence order of errors:
- User does not exist
- Room does not exist
- Booking does not exist
- Invalid date/time
- Room unavailable

To create a new Booking:
Method: POST
Body: userID<int>
roomID<int>
dateOfBooking<date>
-timeFrom<str>
-timeTo<str>
-purpose<str>
Response: (Any of the following)
- Booking created successfully
- Room unavailable
- Room does not exist
- User does not exist
- Invalid date/time

To edit an existing booking:
Method: PATCH
bookingID will be matched with an existing booking, other details can be edited
Body: -userID<int>
-roomID<int>
-bookingID<int>
-dateOfBooking<date>
-timeFrom<str>
-timeTo<str>
-purpose<str>
Response: (Any of the following)
- Booking modified successfully
- Room unavailable
- Room does not exist
- User does not exist
- Booking does not exist
- Invalid date/time

Method: DELETE
~~**Body:** bookingID<int>~~
Query Parameter: bookingID
Response: (One of the following)

- Booking deleted successfully
- Booking does not exist

ALL USERS: Endpoint to show details about all the existing users
URL: /users
Method: GET
Response:<Object>

-name <str>
-userID <int>
-email <str>