CS6.401 Software Engineering Spring 2024

Project 2 Report (Bonus Task)

Team 23

Team Members:

Name	**Roll Number**
Karan Bhatt	2022202003
Madhusree Bera	2022202007
Vedashree Ranade	2022201073
Piyush Rana	2022202012
Yash Maheshwari	2022201074

Bonus Task: Public & Private Bookshelves

Objective:

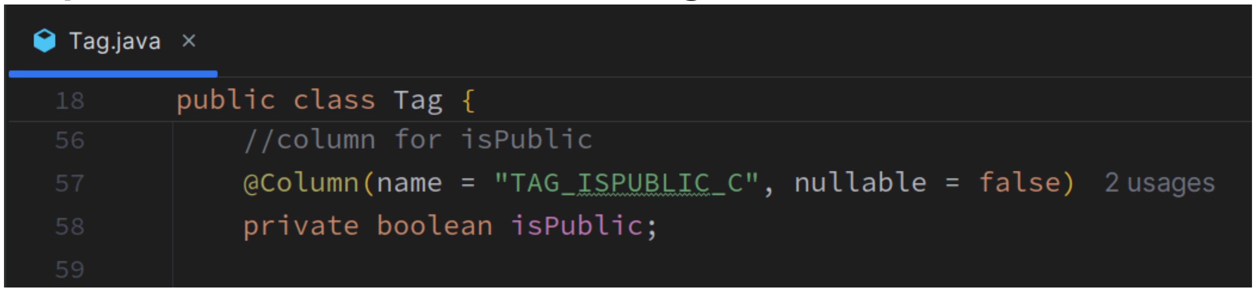
The objective of this task is to introduce the capability for users to adjust the privacy settings of their bookshelves. This enhancement empowers users to mark their bookshelves as either private or public.

Detailed Description:

Implement functionality within the app to enable users to mark their bookshelves as either private or public.
Define “private” bookshelves as accessible only to the respective user who created them, ensuring confidentiality and privacy of personal reading collections.
Designate “public” bookshelves as accessible to all users of the common library system, enabling others to view the books within these shelves in a read-only mode.

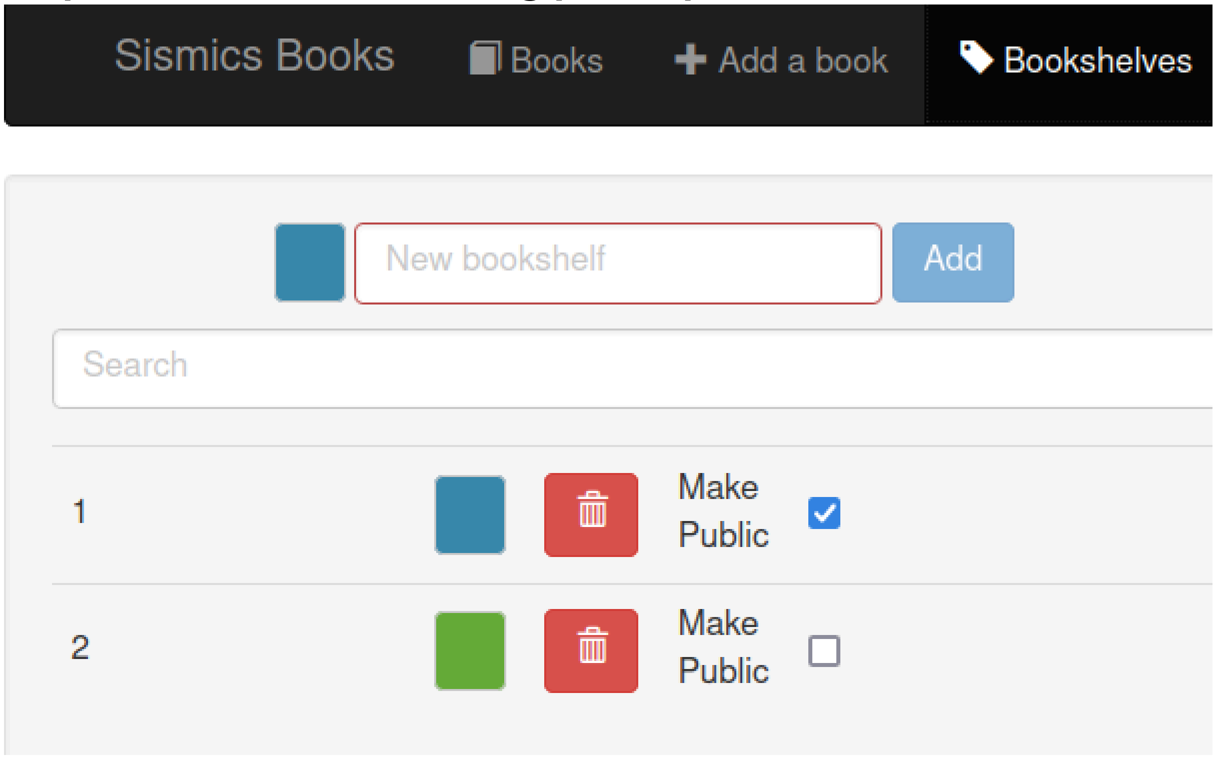
Procedure:

Step 1: Define a new attribute for Tag class named isPublic

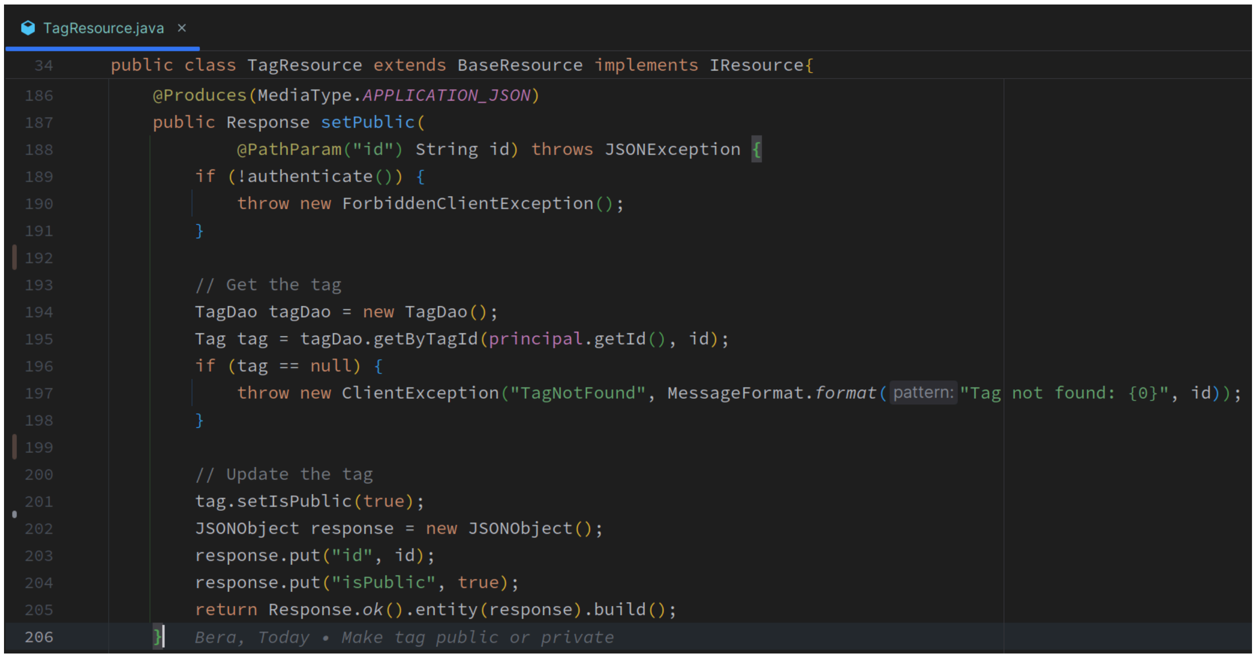


```java
public class Tag {
    //column for isPublic
    @Column(name = "TAG_ISPUBLIC_C", nullable = false)  2 usages
    private boolean isPublic;

```

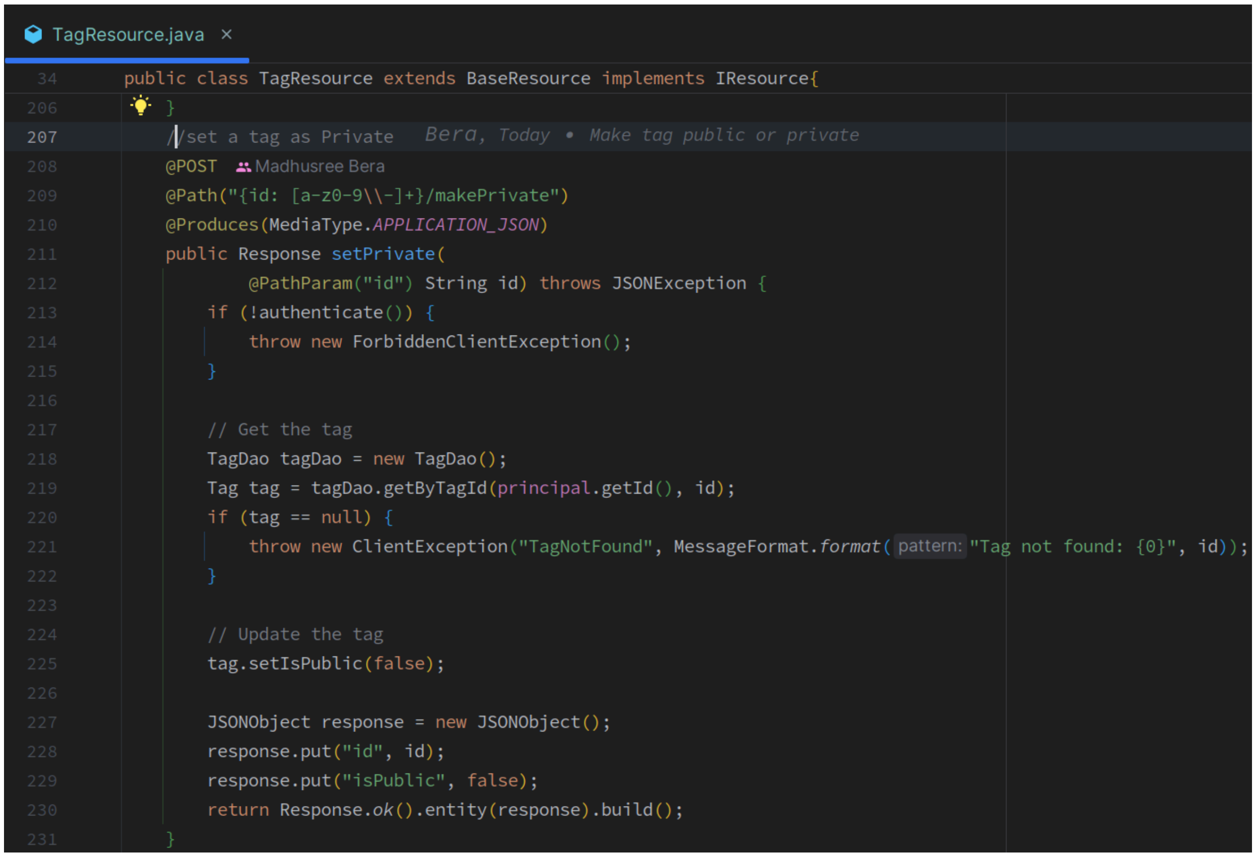
Step 2: Checkbox for setting public/private



Step 3: setPublic and setPrivate functions called when checkbox is ticked

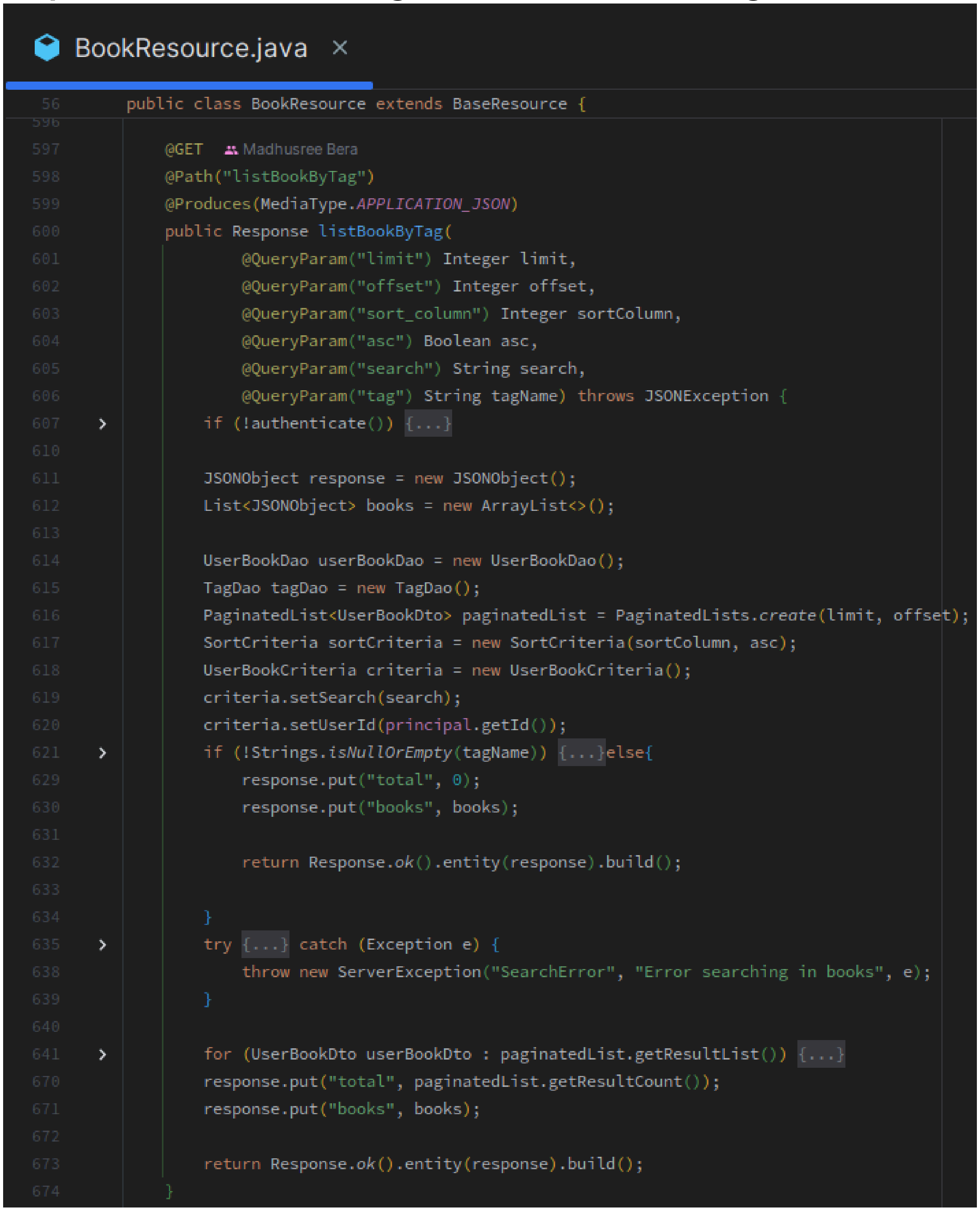


```java
public class TagResource extends BaseResource implements IResource{
    @Produces(MediaType.APPLICATION_JSON)
    public Response setPublic(
            @PathParam("id") String id) throws JSONException {
        if (!authenticate()) {
            throw new ForbiddenClientException();
        }

        // Get the tag
        TagDao tagDao = new TagDao();
        Tag tag = tagDao.getByTagId(principal.getId(), id);
        if (tag == null) {
            throw new ClientException("TagNotFound", MessageFormat.format( pattern: "Tag not found: {0}", id));
        }

        // Update the tag
        tag.setIsPublic(true);
        JSONObject response = new JSONObject();
        response.put("id", id);
        response.put("isPublic", true);
        return Response.ok().entity(response).build();
    }  Bera, Today • Make tag public or private
```



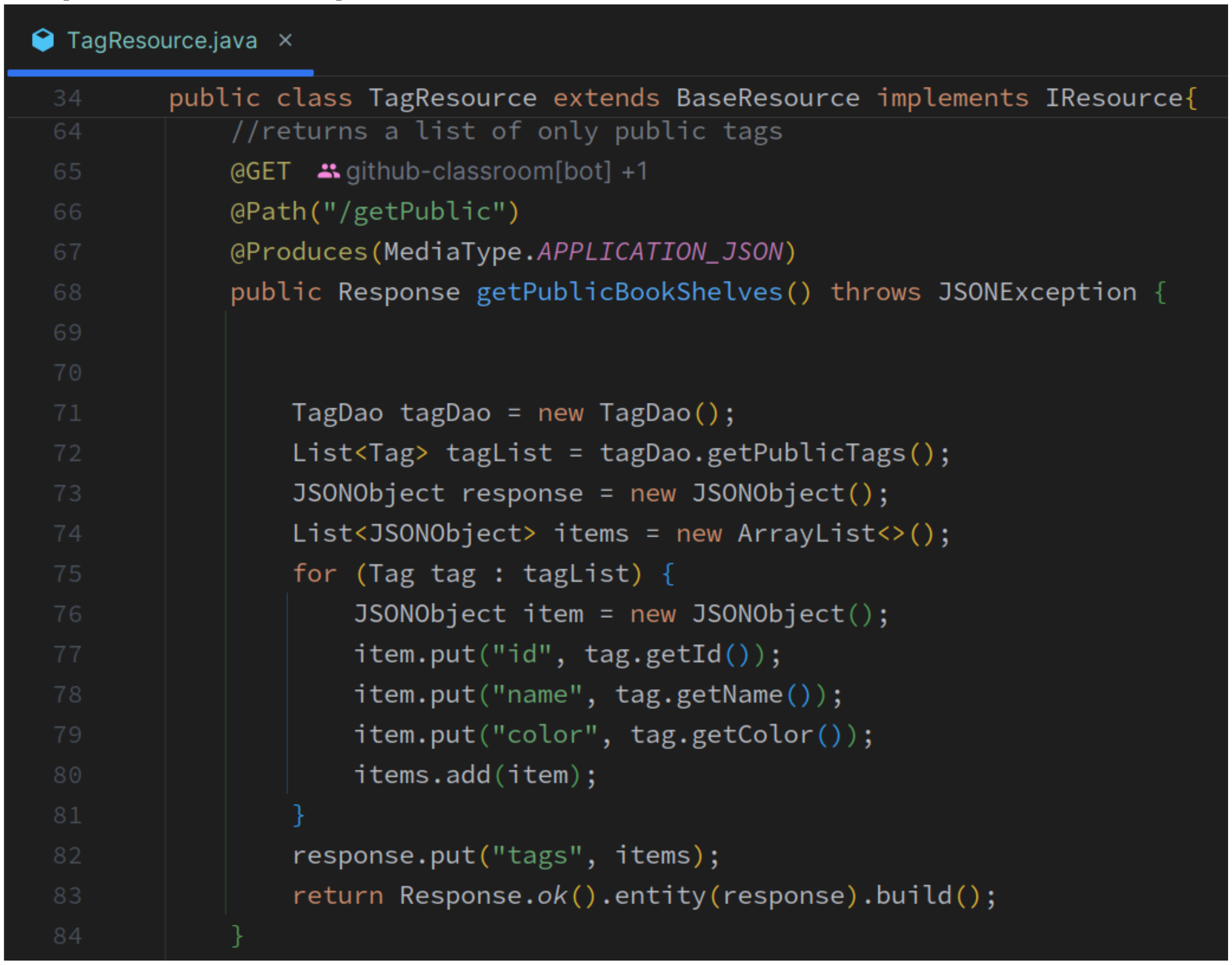
```java
public class TagResource extends BaseResource implements IResource{
    }
    //set a tag as Private   Bera, Today • Make tag public or private
    @POST  Madhusree Bera
    @Path("{id: [a-z0-9\\-]+}/makePrivate")
    @Produces(MediaType.APPLICATION_JSON)
    public Response setPrivate(
            @PathParam("id") String id) throws JSONException {
        if (!authenticate()) {
            throw new ForbiddenClientException();
        }

        // Get the tag
        TagDao tagDao = new TagDao();
        Tag tag = tagDao.getByTagId(principal.getId(), id);
        if (tag == null) {
            throw new ClientException("TagNotFound", MessageFormat.format( pattern: "Tag not found: {0}", id));
        }

        // Update the tag
        tag.setIsPublic(false);

        JSONObject response = new JSONObject();
        response.put("id", id);
        response.put("isPublic", false);
        return Response.ok().entity(response).build();
    }
```

Step 4: Create function to get books of selected tag


BookResource.java ×

```java
public class BookResource extends BaseResource {

    @GET    Madhusree Bera
    @Path("listBookByTag")
    @Produces(MediaType.APPLICATION_JSON)
    public Response listBookByTag(
            @QueryParam("limit") Integer limit,
            @QueryParam("offset") Integer offset,
            @QueryParam("sort_column") Integer sortColumn,
            @QueryParam("asc") Boolean asc,
            @QueryParam("search") String search,
            @QueryParam("tag") String tagName) throws JSONException {
        if (!authenticate()) {...}

        JSONObject response = new JSONObject();
        List<JSONObject> books = new ArrayList<>();

        UserBookDao userBookDao = new UserBookDao();
        TagDao tagDao = new TagDao();
        PaginatedList<UserBookDto> paginatedList = PaginatedLists.create(limit, offset);
        SortCriteria sortCriteria = new SortCriteria(sortColumn, asc);
        UserBookCriteria criteria = new UserBookCriteria();
        criteria.setSearch(search);
        criteria.setUserId(principal.getId());
        if (!Strings.isNullOrEmpty(tagName)) {...}else{
            response.put("total", 0);
            response.put("books", books);

            return Response.ok().entity(response).build();

        }
        try {...} catch (Exception e) {
            throw new ServerException("SearchError", "Error searching in books", e);
        }

        for (UserBookDto userBookDto : paginatedList.getResultList()) {...}
        response.put("total", paginatedList.getResultCount());
        response.put("books", books);

        return Response.ok().entity(response).build();
    }
```

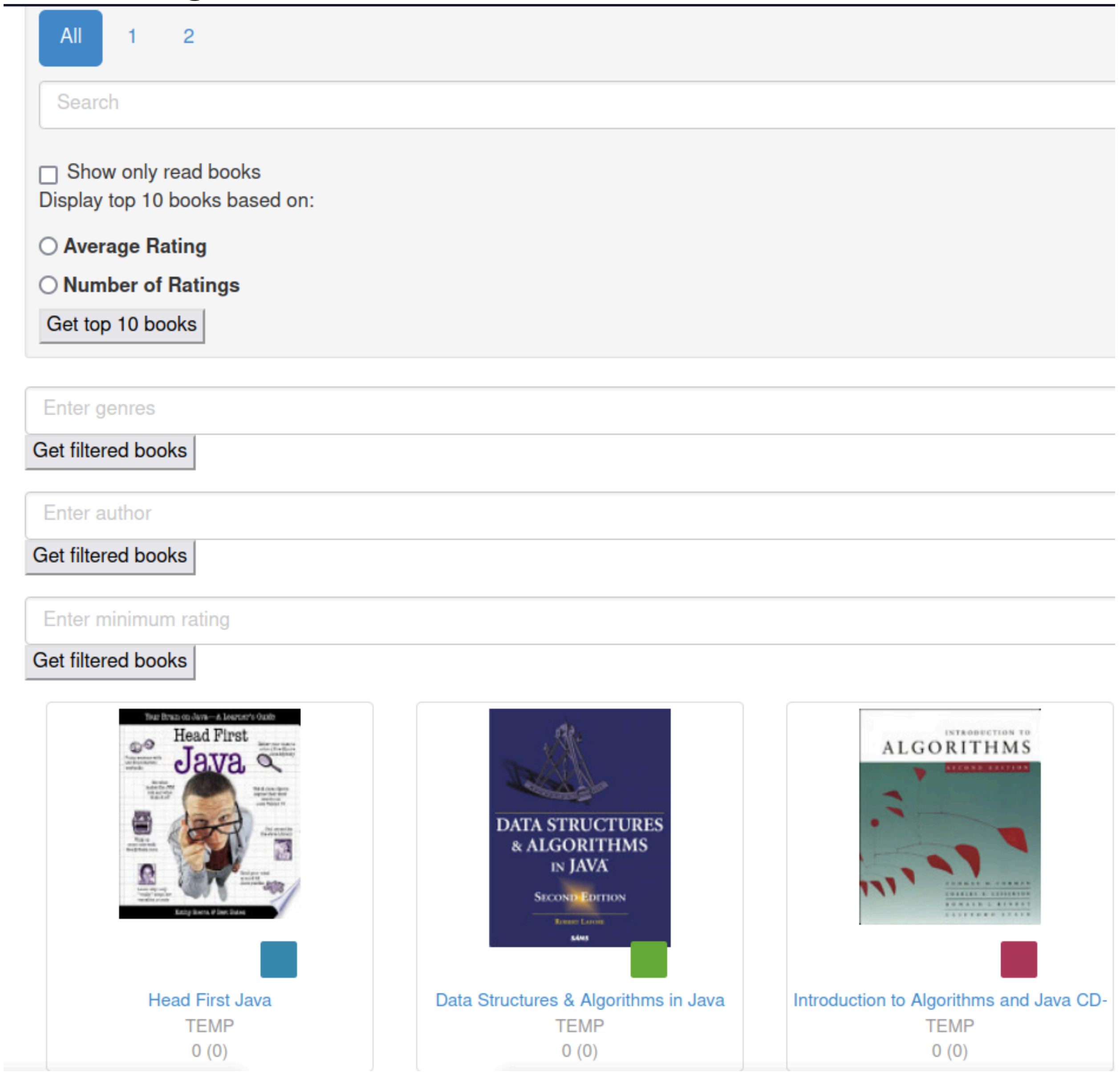
Step 5: Get list of public bookshelves



TagResource.java ×

```java
public class TagResource extends BaseResource implements IResource{
    //returns a list of only public tags
    @GET  github-classroom[bot] +1
    @Path("/getPublic")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getPublicBookShelves() throws JSONException {


        TagDao tagDao = new TagDao();
        List<Tag> tagList = tagDao.getPublicTags();
        JSONObject response = new JSONObject();
        List<JSONObject> items = new ArrayList<>();
        for (Tag tag : tagList) {
            JSONObject item = new JSONObject();
            item.put("id", tag.getId());
            item.put("name", tag.getName());
            item.put("color", tag.getColor());
            items.add(item);
        }
        response.put("tags", items);
        return Response.ok().entity(response).build();
    }
```

Step 6: Create Public Bookshelves Page and list books of the selected tag



All 1 2

Search

☐ Show only read books

Display top 10 books based on:

○ **Average Rating**

○ **Number of Ratings**

Get top 10 books

Enter genres

Get filtered books

Enter author

Get filtered books

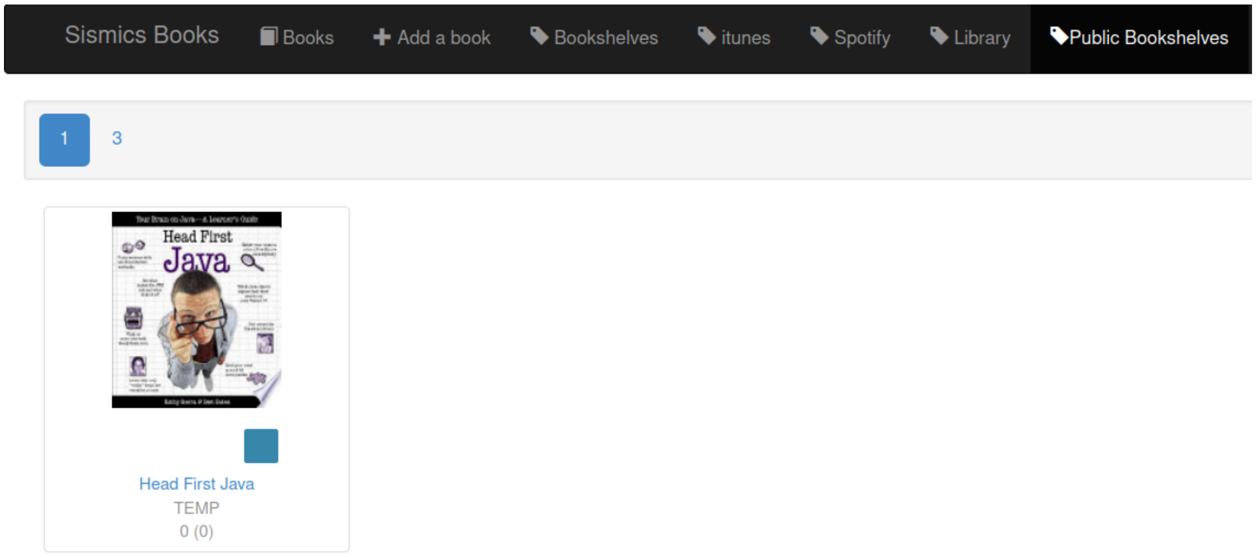
Enter minimum rating

Get filtered books

Head First Java
TEMP
0 (0)

Data Structures & Algorithms in Java
TEMP
0 (0)

Introduction to Algorithms and Java CD-
TEMP
0 (0)



<u>Summary:</u>

In this feature, we defined a new attribute called "isPublic" for the Tag class. Users can then use a checkbox to set their bookshelves as public or private. When the checkbox is ticked, corresponding functions are called to set the bookshelf as either public or private.

Additionally, we created functions to retrieve books of selected tags and to list public bookshelves. A Public Bookshelves Page was designed to display books of selected tags from public bookshelves.

After creating a bookshelf, the user can mark it as Public and add books to that bookshelf. Other registered users can see the list of all public bookshelves and the books it contains.

<u>Code:</u>

The code for the bonus feature is present in branch: *project2_bonus*