**CHARTER**

Identify capabilities and areas of potential instability of the “rest api todo list manager”.

Identify documented and undocumented “rest api todo list manager” capabilities.

For each capability create a script or small program to demonstrate the capability.

Exercise each capability identified with data typical to the intended use of the application.

**BUILD**

runTodoManagerRestAPI-1.5.5.jar

**AREA**

Test “category” feature

**ENVIRONMENT**

MacOS Ventura

**START**

3:14PM 02/09/2025

**DURATION**

45 mins

**TESTER**

Mahirul Islam 261055404 mahirul.islam@mail.mcgill.ca

Tiffany Miller 261113912 tiffany.miller@mail.mcgill.ca

**SESSION NOTES**

3:14 PM – 11/02/2025: Set request headers to accept JSON data.

3:17 PM: Sent a GET request to the /categories endpoint to retrieve all categories. Response returns the full list of categories as expected.

3:20 PM: Tested the HEAD request for /categories. Returns status code 200 as expected.

3:21 PM: Created a new category with XML payload (POST /categories), specifying a title and description. The ID is auto-generated, and the category is successfully added.

3:23 PM: Sent a GET request for an existing category with ID = 1 (GET /categories/5). Returns the correct category details.

3:25 PM: Tested a GET request for a nonexistent category (GET /categories/20). Returns a 404 error as expected.

3:27 PM: Sent a PUT request to update the description of an existing category with ID = 3 (PUT /categories/2). Successfully updated the category.

3:29 PM: Attempted to update a nonexistent category (PUT /categories/20). Returns a 404 error as expected.

3:31 PM: Sent a DELETE request to remove a category with ID = 2 (DELETE /categories/2). Successfully deleted the category.

3:33 PM: Tried to delete a category with a nonexistent ID (DELETE /categories/20). Returns a 404 error as expected.

3:35 PM: Retrieved all project items related to category ID = 1 (GET /categories/1/projects). Response returns an empty list with status 200 as expected.

3:37 PM: Retrieved all project items related to non-existing category ID = 20 (GET /categories/20/projects). Response returns status 200 with projects related to another category. **##BUG**

3:39 PM: Sent a POST request to create a project relationship for category ID = 1 and project ID = 4 (POST /categories/1/projects). Relationship created successfully.

3:41 PM: Sent a DELETE request to remove the project relationship (DELETE /categories/1/projects/4). Successfully deleted the relationship.

3:47 PM: Attempted to retrieve todos related to category ID = 1 (GET /categories/1/todos). Returns a list of todos as expected.

3:52 PM: Tested a GET request for todos related to a nonexistent category (GET /categories/20/todos). Response return empty todo, which was unexpected. **##BUG**

**BUGS LOGGED**

1) Endpoint: /categories/:id/projects, Request Type: GET

Description: Trying to get all project items related to a nonexistent category with ID = 20. Returns status 200 and project related to another category. This wasn't expected as the category with the provided ID doesn't exist.

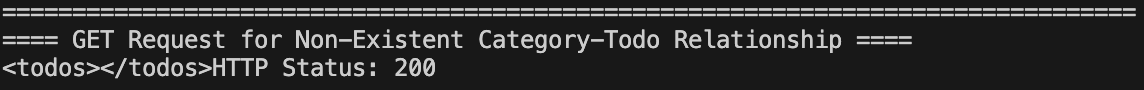
Risk: Data concern. The user should not have access to this data.

A black screen with white text

AI-generated content may be incorrect.

2) Endpoint: /categories/:id/todos, Request Type: GET

Description: Trying to get all todos related to a nonexistent category with ID = 20. Returns empty list. This wasn't expected as the category with the provided ID doesn't exist.

Risk: Data concern. The user should not have access to this data.

Refer to bug summaries: CategoryAndInteroperabilityBug1.docx and CategoryAndInteroperabilityBug2.docx

**AREAS OF POTENTIAL RISK**

Data Exposure:

Bug #1: Returns data for an invalid request (nonexistent category).

Bug #2: Returns empty list for invalid request (nonexistent category).

Risk: Users should not have access to data for invalid requests. This could lead to unintended data exposure.

**CAPABILITIES**

Create, Read, Update, and Delete categories using XML payloads.

Retrieve headers for category endpoints.

Retrieve and delete relationships between categories, projects, and todos.

Refer to capabilities scripts: categoryJSON.sh and categoryXML.sh

**SUMMARY OF SESSION FINDINGS**

* The API supports CRUD operations for categories.
* We can create, retrieve and delete relationships between categories, projects and todos.
* The API returns status 200 with a project of another category when the category requested is invalid. This could leak unwanted data to the user.
* Some endpoints return 404, while others return 200 with an empty list, making the API error handling inconsistent.

**NEW TESTING IDEAS**

* Verify that all endpoints return correct error codes
* Ensure that error codes are consistent for different API’s error handlings