**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 interoperability

**ENVIRONMENT**

MacOS Ventura

**START**

5:10PM 02/11/2025

**DURATION**

45 mins

**TESTER**

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

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

**SESSION NOTES**

5:11 PM – 11/02/2025: Set request headers to accept JSON data.

5:17 PM: Sent a GET request to the /todos endpoint to retrieve all todos. Response returns the full list of todos as expected.

5:20 PM: Created a new todo with a POST request to /todos. The ID is auto-generated, and the todo is successfully added.

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

5:25 PM: Created a new project with a POST request to /projects. The ID is auto-generated, and the project is successfully added.

5:27 PM: Created a new category with a POST request to /categories. The ID is auto-generated, and the category is successfully added.

5:29 PM: Created a relationship between todo ID = 1 and project ID = 1 using POST /todos/1/tasksof. Relationship created successfully.

5:31 PM: Created a relationship between todo ID = 1 and category ID = 1 using POST /todos/1/categories. Relationship created successfully.

5:34 PM: Create a relationship between category ID = 1 and project ID = 1 using POST /categories/1/projects. Relation created successfully.

5:35 PM: Retrieved projects of a non-existing category ID = 20. Response is 200 with an empty list **##BUG**

5:37 PM: Retrieved all categories related to todo ID = 1 (GET /todos/1/categories). Response returns the correct category.

5:37 PM: Retrieved all todos related to project ID = 1 (GET /projects/1/tasks). Response returns the correct todo.

5:39 PM: Retrieved all todos related to category ID = 1 (GET /categories/1/todos). Response returns the correct todo.

5:42 PM: Get all todos of a non-existing category ID = 20. Response return 200 and an empty list of todos. **##BUG**

5:45 PM: Create a relationship between category ID = 1 with non-existing project ID = 20. Returns 404 as expected.

5:47 PM: Deleted the relationship between todo ID = 1 and project ID = 1 using DELETE /todos/1/tasksof/1. Relationship deleted successfully.

5:49 PM: Deleted the relationship between todo ID = 1 and category ID = 1 using DELETE /todos/1/categories/1. Relationship deleted successfully.

5:50 PM: Deleted todo ID = 1 using DELETE /todos/1. Todo deleted successfully.

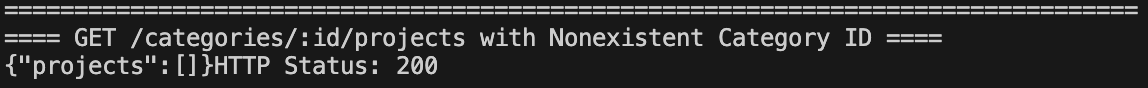
5:52 PM: Deleted project ID = 1 using DELETE /projects/1. Project deleted successfully.

5:53 PM: Deleted category ID = 1 using DELETE /categories/1. Category deleted successfully.

**BUGS LOGGED**

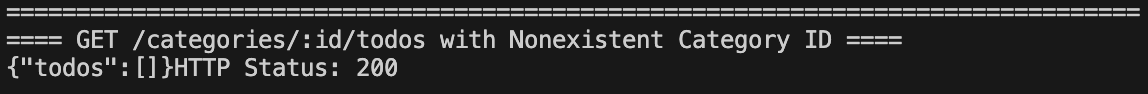
1) Endpoint: /categories/:id/projects, Request Type: GET

Description: When retrieving projects for a nonexistent category (ID = 20), the API returns a status 200 with an empty list. This is unexpected because the category does not exist, and the response should indicate an error (e.g., 404 Not Found).

Risk: Data integrity concern. The API should not return a successful response for a nonexistent resource.

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

Description: When retrieving todos for a nonexistent category (ID = 20), the API returns a status 200 with an empty list. This is unexpected because the category does not exist, and the response should indicate an error (e.g., 404 Not Found).

Risk: Data integrity concern. The API should not return a successful response for a nonexistent resource.

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

**AREAS OF POTENTIAL RISK**

Data Integrity: The API outputs 200 status codes for non-existing objects. This behavior can mislead clients into thinking the request was successful, even though the resource does not exist.

Consistency: The API behaves inconsistently, outputting 200 status codes when there are errors.

**CAPABILITIES**

Create, Read, Update, and Delete todos, projects, and categories.

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

Retrieve todos, projects, and categories related to each other.

Refer to capabilities scripts: interoperabilityJSON.sh and interoperabilityXML.sh

**SUMMARY OF SESSION FINDINGS**

* The API supports CRUD operations for todos, projects, and categories.
* We can create, retrieve and delete relationships between categories, projects and todos.
* The API returns status 200 with an empty list for nonexistent resources which is unexpected behavior.
* 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
* Verify that all endpoints behave similarly when handling non-existing IDs
* Test API’s response time with larger data