

Tester: YunShan Nong (261055472) ([yunshan.nong@mail.mcgill.ca](mailto:yunshan.nong@mail.mcgill.ca))

CHARTER: TD000A-001 Rest API Todo List Manger

-----

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

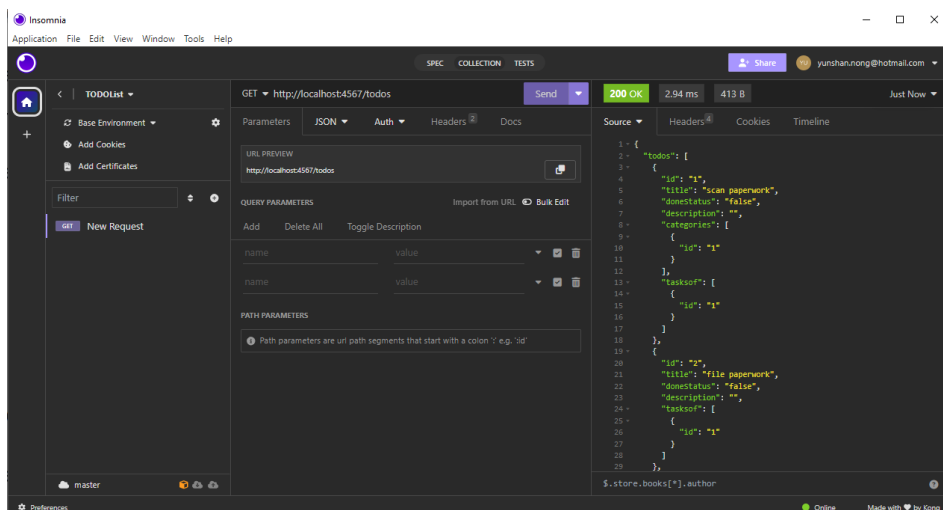
Identify documented and undocumented “rest api todo list manager” capabilities related to todos.

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.

-----

For this session, I am using Insomnia API system to test Rest API todo list manager. The reference files (screenshots) are directly included in the session notes to facilitate readings. The script used for testing capabilities is mainly in the screenshots.



START

-----

02/04/2024 10:00am-10:45am

TEST NOTES

-----

**/todos**

**GET /todos** (return all the instances of todo)

Before any new creation, GET <http://localhost:4567/todos> returns the existing tasks in todo list.

**HEAD /todos** (headers for all the instances of todo)

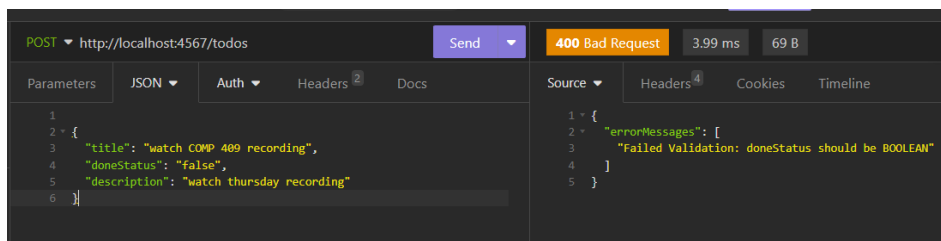
HEAD <http://localhost:4567/todos>: No error message, no response body, as expected

**POST /todos** (we should be able to create todo without a ID using the field values in the body of the message)

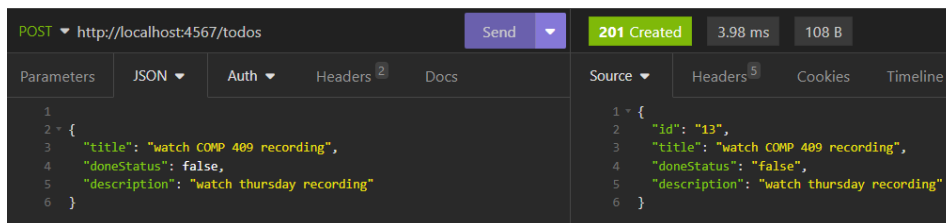
**PROBLEM:** Using the example JSON input structure in the documentation to make API call, we get an error for doneStatus

Example JSON Input to API calls

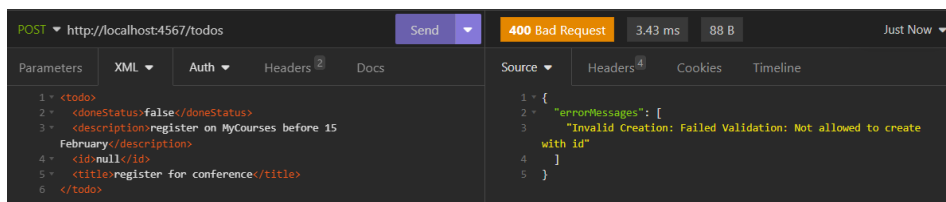
```
{
  "title": "dunt ut labore et do",
  "doneStatus": "false",
  "description": "labore et dolore maa"
}
```



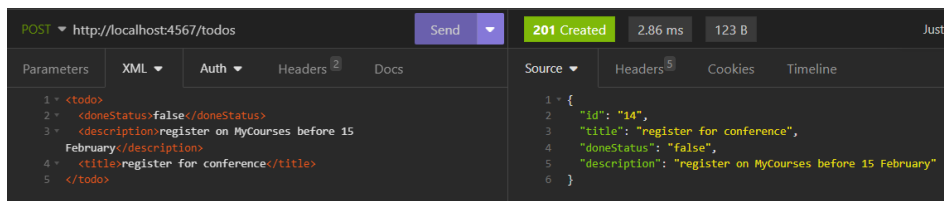
After removing the quotation mark for doneStatus, we can create todo. A random id is assigned to the task. The output JSON is the same as expected.



**PROBLEM:** Using the example XML input to make API calls, we get an error message, not allowed to create with id. (Potential inability)



After removing the id, creation succeed as expected. The output is in JSON, not in XML.



## /todos/:id

**GET /todos/:id** (return a specific instances of todo using a id)

GET <http://localhost:4567/todos/:1> failed, error message: "Could not find an instance with todos/:1"

Get <http://localhost:4567/todos/1> worked, return the specific instance of id 1, also includes the Categories and the Taskof (project) id if the task has a category or project associated. If not, only displays id, title, doneStatus and description.

```

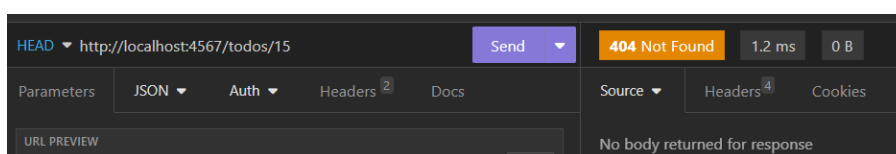
1 {
2   "todos": [
3     {
4       "id": "1",
5       "title": "scan paperwork",
6       "doneStatus": "false",
7       "description": "",
8       "categories": [
9         {
10          "id": "1"
11        }
12      ],
13       "tasksof": [
14         {
15          "id": "1"
16        }
17      ]
18     }
19   ]
20 }

```

For a non-existing task, we get an errorMessage: ["Could not find an instance with todos/id"]

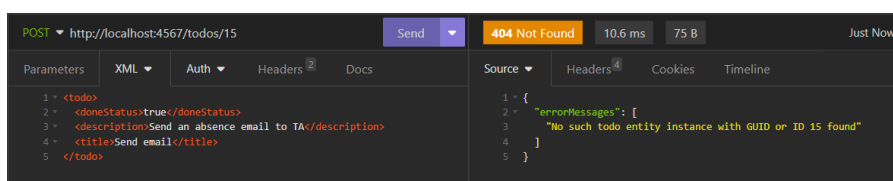
**HEAD /todos/:id** (headers for a specific instances of todo using a id)

Working as expected, no return body. When the task does not exist, we get 404 Not Found.

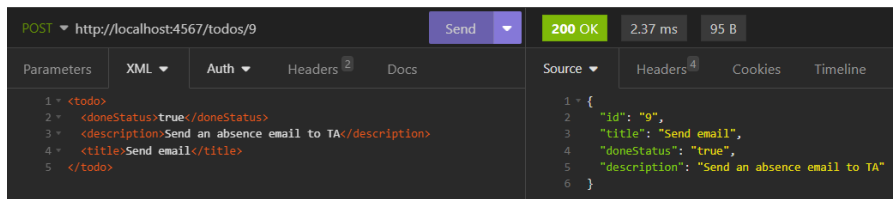


**POST /todos/:id** (amend a specific instances of todo using a id with a body containing the fields to amend)

Post a task with a non-existing id, we get error message



Post a task with an existing id, the fields got updated with no error



**PUT /todos/:id** (amend a specific instances of todo using a id with a body containing the fields to amend)

Act the same way as POST for existing id.

For non-existing id, we get a different error message: "Invalid GUID for 19 entity todo" no matter what is the input data.

**DELETE /todos/:id** (delete a specific instances of todo using a id)

Deleting a non-existing id, we get an error message as expected: "Could not find any instances with todos/19"

Deletion of a specific instance works fine, no return body, the task disappears from the full todo lists.

**/todos/:id/categories**

**GET /todos/:id/categories** (return all the category items related to todo, with given id, by the relationship named categories)

Return the category item related to todo, with the given id, without error. Returns an empty array if there is no category for the task.

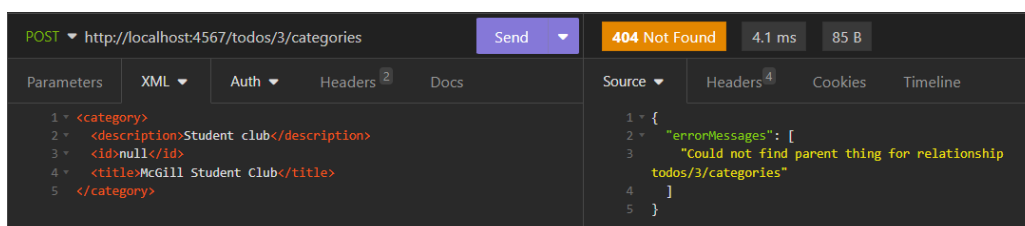
PROBLEM: using a non-existing id, non error message, still returns category 1. This is undocumented and might create instability (First goal in the charter)

**HEAD /todos/:id/categories** (headers for the category items related to todo, with given id, by the relationship named categories)

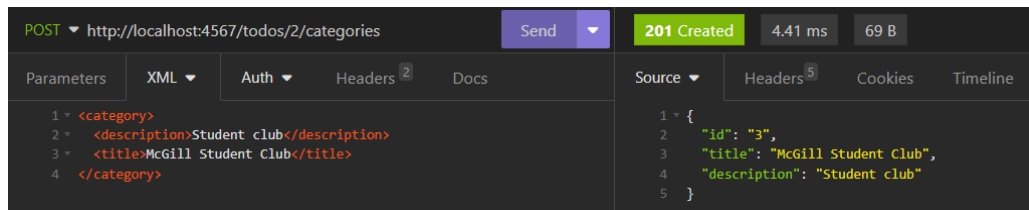
Headers for the category items related to todo, works like GET. Same problem with GET, no error using a non-existing todo

**POST /todos/:id/categories** (create an instance of a relationship named categories between todo instance :id and the category instance represented by the id in the body of the message)

Using the example XML input structure, we get an error message. When I used a non-existing id, we get the same error message.

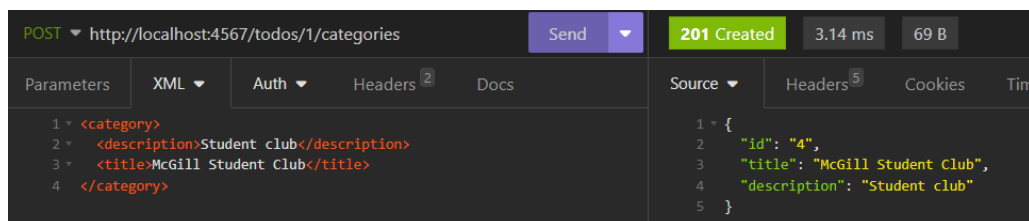


After taking off `<id></id>`, a relation between the new Category and the todo instance is created with success.



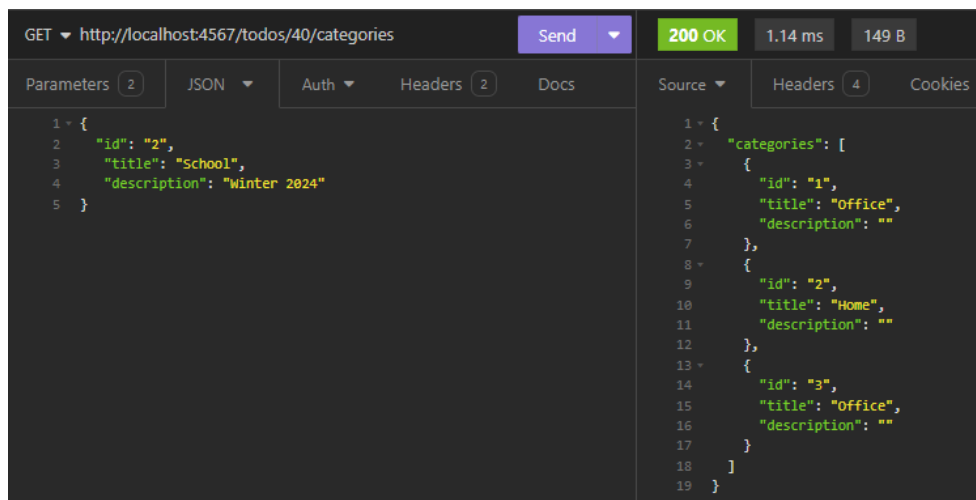
```
POST http://localhost:4567/todos/2/categories 201 Created 4.41 ms 69 B
Parameters XML Auth Headers 2 Docs Source Headers 5 Cookies Timeline
1 <category>
2 <description>Student club</description>
3 <title>McGill Student Club</title>
4 </category>
1 {
2   "id": "3",
3   "title": "McGill Student Club",
4   "description": "Student club"
5 }
```

PROBLEM: We can not create a Category with the same id. Id varies each time we create a relation, even if the fields are the same. This is also undocumented behavior.



```
POST http://localhost:4567/todos/1/categories 201 Created 3.14 ms 69 B
Parameters XML Auth Headers 2 Docs Source Headers 5 Cookies Tim
1 <category>
2 <description>Student club</description>
3 <title>McGill Student Club</title>
4 </category>
1 {
2   "id": "4",
3   "title": "McGill Student Club",
4   "description": "Student club"
5 }
```

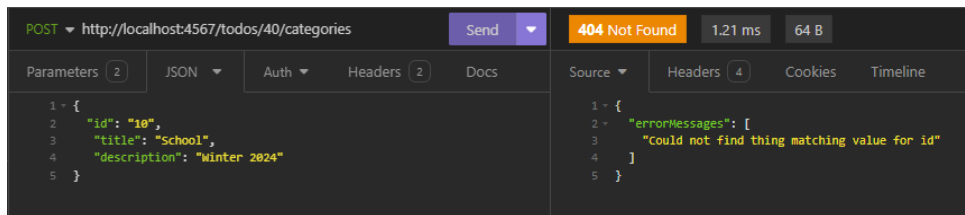
PROBLEM: After I posted a Category with an existing id, the Category got related to the todo task, but the fields of the Category are not updated. This is undocumented behaviour (third goal in the charter) \*Testing idea



```
GET http://localhost:4567/todos/40/categories 200 OK 1.14 ms 149 B
Parameters 2 JSON Auth Headers 2 Docs Source Headers 4 Cookies
1 {
2   "id": "2",
3   "title": "School",
4   "description": "Winter 2024"
5 }
1 {
2   "categories": [
3     {
4       "id": "1",
5       "title": "Office",
6       "description": ""
7     },
8     {
9       "id": "2",
10      "title": "Home",
11      "description": ""
12     },
13     {
14       "id": "3",
15       "title": "Office",
16       "description": ""
17     }
18   ]
19 }
```

(In this picture, we can see that Category with id "2" stayed the same after I posted it with new fields.

If I use POST with a non-existing id for Category, we get an error message. Does not allow to assign customized id to Category. \*Testing idea



## /todos/:id/categories/:id

**DELETE /todos/:id/categories/:id** (delete the instance of the relationship named categories between todo and category using the :id)

Delete works as expected when the relation exists.

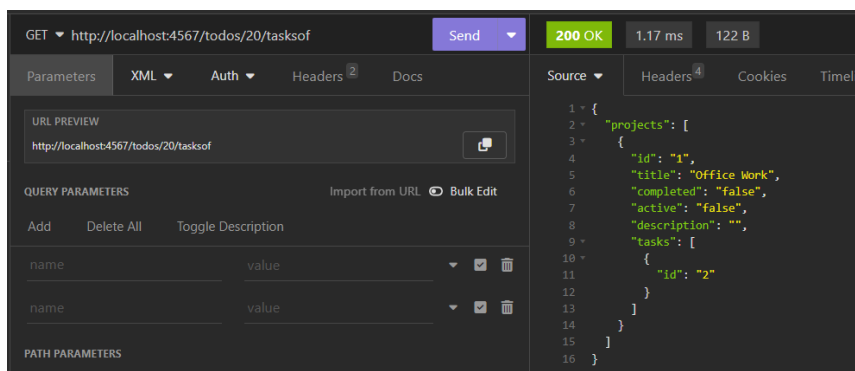
For a non-existing category id or todo id, we get an error message , ex: "Could not find any instances with todos/1/categories/5"

## /todos/:id/tasksof

**GET /todos/:id/tasksof** (return all the project items related to todo, with given id, by the relationship named tasksof)

Working as expected for existing todo. Returns all the projects related to todo, if empty, returns an empty array.

PROBLEM: When the todo does not exist, still returns a list of projects. This is a potential instability since the return body is inaccurate (first goal in the charter) \*Testing idea



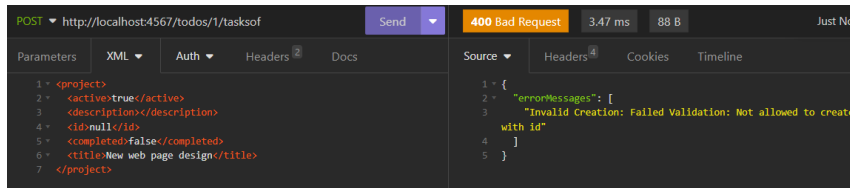
**HEAD /todos/:id/tasksof** (headers for the project items related to todo, with given id, by the relationship named tasksof)

Works fine with existing todos. No body returned.

PROBLEM: Similar problem with GET, when todo does not exist, does not show Not Found.

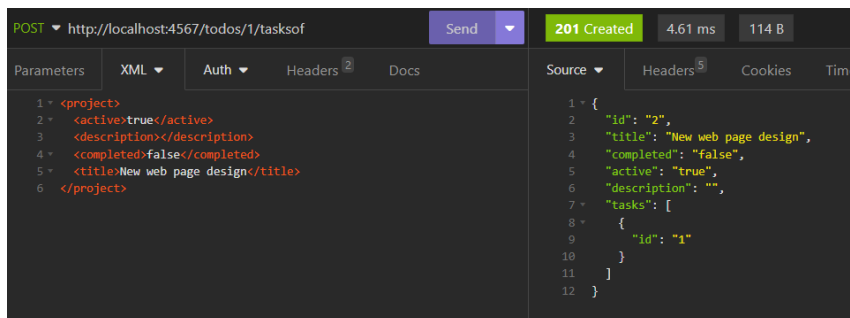
**POST /todos/:id/tasksof** (create an instance of a relationship named tasksof between todo instance :id and the project instance represented by the id in the body of the message)

PROBLEM: Following the example input in documentation, we get an error.

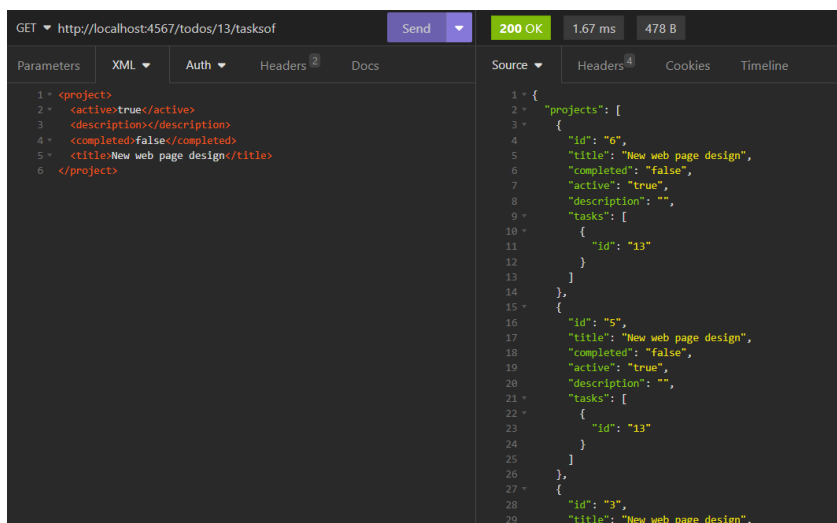


If the todo does not exist, we get an error message, ex: "Could not find parent thing for relationship todos/20/tasksof"

We need to take off <id></id> to make it work.



PROBLEM: The function creates a new project (with different id) each time we post. So it creates duplicates of the same content.



**/todos/:id/tasksof/:id**

**DELETE /todos/:id/tasksof/:id** (delete the instance of the relationship named tasksof between todo and project using the :id)

Works fine. Delete the instance of relationship names tasksof between todo and project if the relationship exists (nothing get returned). Otherwise, we get an error message, ex: "Could not find any instances with todos/13/tasksof/5"

-----

### Summary of session findings

- YunShan did individually this exploratory testing on the charter
- The purpose of the charter was to identify capabilities and potential issues of the “rest api todo list manager” and test these capabilities using a script and input data.
- YunShan took session notes while she tests the API locally on API system Insomnia.
- Exploratory testing session was around 45 minutes
- Mainly tested all functionalities related to todos
- Followed the documentation to create input script, partially using JSON and partially using XML
- Compared the output results with the documentation
- Identified potential bugs (issues are marked with PROBLEM keyword inside the session notes)

### List of concerns

- Users cannot assign id to a todo instance, which is inconvenient -> cannot recreate an deleted instance
- The order of instances in return body is random
- Some input examples in documentation does not match with the expected input
- Some GET functionalities do not handle non-existing todo id, no error messages
- New ids got generated each time even if the fields are the same (create too many duplicates)
- Duplicates for categories is confusing

### List of testing ideas identified in session:

- Confirm we can add todos using after deleting todo tasks
- Confirm we get an error message when we use GET on a non-existing id
- Confirm we get an error message when we use GET on a non-existing relationship
- Confirm we can update the fields of an existing todo
- Confirm an empty array is returned when there is no Category or Project associated with the todo
- Confirm we can add multiple categories and projects
- What if the input type for doneStatus is invalid?
- What if we use invalid JSON or XML structure for input?
- What if the input structure for POST is invalid?
- Study the time taken for Posting a todo
- Study the time taken for Getting a todo from a long list of todos
- Study the differences between PUT and POST
- What if the user sends too many requests within a short period of time
- What if the user wants to reorder the list
- What if the user made a mistake by accidentally deleting a todo and the user wants to restore it.