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

Typical Scenarios – main functions and capabilities of relationship between TODOs and PROJECTs

Exploratory testing for API endpoints

Environment

MacOS 26.3

Postman software

START

3:15PM 12/02/2026

TESTER

Ethan Wu	261117309	ethan.wu2@mail.mcgill.ca
Kenny Nguyen	261120429	kenny.nguyen@mail.mcgill.ca

#DURATION

45 minutes

TEST NOTES

3:15 PM 12/02/2026 [GET /todos] Get request to todos endpoint. Returns a list of the current projects as expected. Status 200 OK.

3:16 PM 12/02/2026 [GET /projects] Get request to projects endpoint. Returns a list of the current projects as expected. Status 200 OK.

3:17 PM 12/02/2026 [POST /todos/ :id/ tasksof] Post request to tasksof to relate a project to a todo with a valid todo id. The project id is set in the body. status 201 CREATED.

3:21 PM 12/02/2026 [POST /todos/ :id/tasksof] Post request to tasksof to relate a project to a todo with a valid todo id. The project id is not set thus an empty project is also created. status 201 CREATED.

3:25 PM 12/02/2026 [GET /todos/ :id/tasksof] Get request to tasksof endpoint. Returns the list of relationships of the todo with projects. Status 200 OK.

3:28 PM 12/02/2026 [GET /todos/ :id/tasksof] Get request to tasksof endpoint with invalid id. Returns the list of relationships of the todo with projects. Status 200 OK. BUG since it returns the list anyways.

???? bug

3:29 PM 12/02/2026 [HEAD /todos/ :id/tasksof] Get request to tasksof endpoint with valid id. Returns the header of the relationships of todo with projects. Status 200 OK.

3:32 PM 12/02/2026 [HEAD /todos/ :id/tasksof] Get request to tasksof endpoint with invalid id. Returns the header of the relationships of todo with projects. Status 200 OK.

???? bug same bug

3:34 PM 12/02/2026 [Delete /todos/ :id/tasksof/ :id] Delete request to the tasksof endpoint with specific id. It successfully deletes the relationship between the project and the todo. Status 200 OK.

3:35 PM 12/02/2026 [POST /projects/ :id/ tasks] Post request to tasks to relate a project to a todo with a valid project id. The todosid is set in the body. status 201 CREATED.

3:37 PM 12/02/2026 [POST /projects/ :id/tasks] Post request to tasksof to relate a project to a todo with a valid project id. The todos id is not set in the body. The request fails with status 400 BAD REQUEST.

3:38 PM 12/02/2026 [GET /projects/ :id/tasks] Get request to tasks endpoint with valid id. Returns the list of relationships of the todo with projects. Status 200 OK.

3:40 PM 12/02/2026 [GET /projects/ :id/tasks] Get request to tasks endpoint with invalid id. Returns the list of relationships of the todo with projects. Status 200 OK. BUG since it returns the list anyways.

?? bug

3:41 PM 12/02/2026 [HEAD /projects/ :id/tasks] Get request to tasks endpoint with valid id. Returns the header of the relationships of todo with projects. Status 200 OK.

3:42 PM 12/02/2026 [HEAD /projects/ :id/tasks] Get request to tasksof endpoint with invalid id. Returns the header of the relationships of todo with projects. Status 200 OK.

??bugg

3:44 PM 12/02/2026 [Delete /todos/ :id/tasks/ :id] Delete request to the tasks endpoint with specific id. It successfully deletes the relationship between the project and the todo. Status 200 OK.

3:45 PM 12/02/2026 [Delete /todos/ :id/tasks/ :id] Delete request to the tasks endpoint with invalid project id. It fails to delete a relationship between the project and the todo as expected. Status 404 NOT FOUND.

3:46 PM 12/02/2026 [Delete /todos/ :id/tasks/ :id] Delete request to the tasks endpoint with invalid todo id. It fails to delete a relationship between the project and the todo as expected. Status 404 NOT FOUND.

3:49 PM 12/02/2026 [POST /todos/ :id/ tasksof] Post request to relate a tasksof to a todo endpoint with specific todo valid id and in xml format. Does not return anything even if the format is the same as the api asks. Status 400 BAD REQUEST (cannot parse XML body).

??bug

3:55 PM 12/02/2026 [POST /projects/ :id/ tasks] Post request to relate a task to a project endpoint with specific project valid id and in xml format. Does not return anything even if the format is the same as the api asks. Status 400 BAD REQUEST (cannot parse XML body).

??bug

3:57 PM 12/02/2026 [GET /projects/ :id/tasks] Get request to tasks endpoint with no id in the URL. Returns the list of relationships of the todo with projects. Status 200 OK. Undocumented api capability where its functionality makes sense.

undocumented api capability

Areas covered:

---

endpoints related to the relationship between TODOs and PROJECTs

** Ending session

Decisions Made During Session

Decided to focus only on relationship endpoints between TODOs and PROJECTs due to time constraint (45 minutes).

Decided not to test core CRUD endpoints (POST /todos, DELETE /projects) since these were covered in Session 1.

After discovering invalid ID bug on /todos/:id/tasksof, decided to test similar behavior on /projects/:id/tasks to check for symmetry.

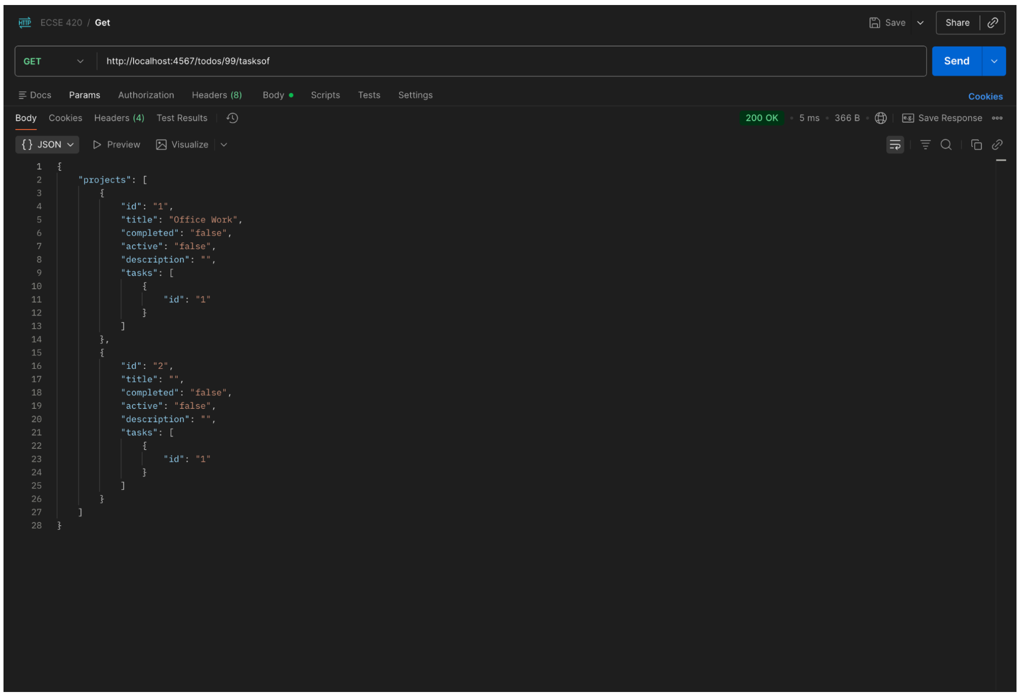
After discovering JSON works but XML fails, decided to test XML format for both relationship endpoints.

After discovering invalid ID bug on GET /projects/:id/tasks, decided to test with GET /projects/tasks (no id in URL) to see if it behaves the same. Turns out to be an undocumented api capability.

Bugs logged

1) endpoint: GET /todos/:id/tasksof

Performing GET /todos/:id/tasksof with non-existing TODO ID (99) returned data for all other valid IDs instead in the JSON message. This is incorrect behavior. Should return an error with corresponding status message. ##BUG (Refer to .docx)

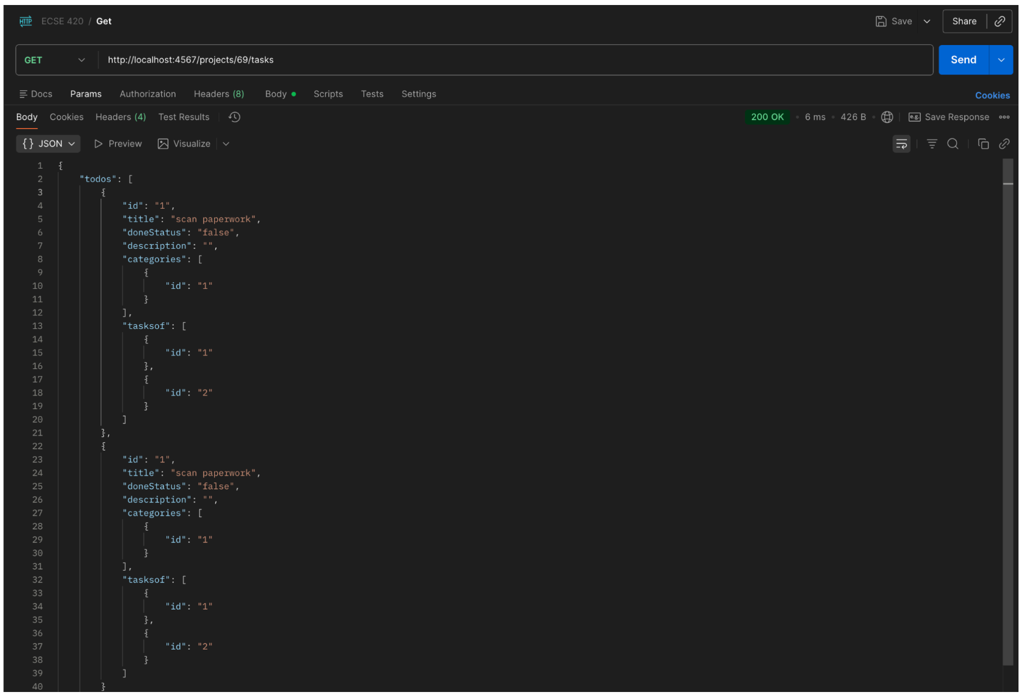


```
GET http://localhost:4567/todos/99/tasksof
200 OK • 5 ms • 366 B
```

```
{
    "projects": [
        {
            "id": "1",
            "title": "Office Work",
            "completed": "false",
            "active": "false",
            "description": "",
            "tasks": [
                {
                    "id": "1"
                }
            ]
        },
        {
            "id": "2",
            "title": "",
            "completed": "false",
            "active": "false",
            "description": "",
            "tasks": [
                {
                    "id": "1"
                }
            ]
        }
    ]
}
```

2) endpoint: GET /projects/:id/tasks

Performing GET /projects/:id/tasks with non-existing PROJECT ID (69) returned data for all other valid IDs instead in the JSON message. This is incorrect behavior. Should return an error with corresponding status message. ##BUG (Refer to .docx)

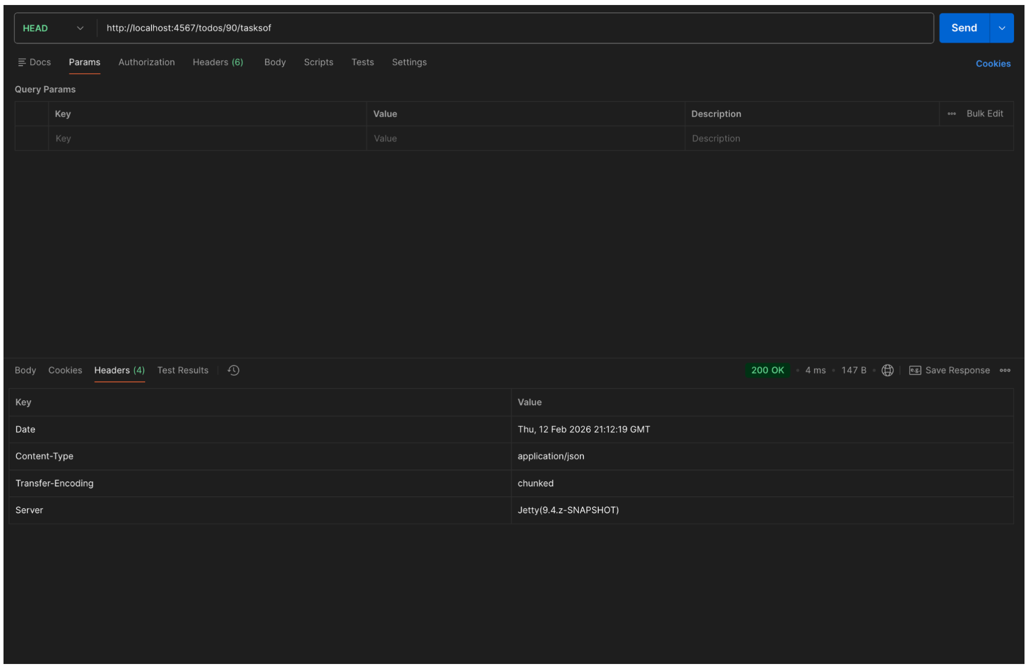


```
GET http://localhost:4567/projects/69/tasks
200 OK • 6 ms • 426 B
```

```
{
    "todos": [
        {
            "id": "1",
            "title": "scan paperwork",
            "doneStatus": "false",
            "description": "",
            "categories": [
                {
                    "id": "1"
                }
            ],
            "tasksof": [
                {
                    "id": "1"
                },
                {
                    "id": "2"
                }
            ]
        },
        {
            "id": "1",
            "title": "scan paperwork",
            "doneStatus": "false",
            "description": "",
            "categories": [
                {
                    "id": "1"
                }
            ],
            "tasksof": [
                {
                    "id": "1"
                },
                {
                    "id": "2"
                }
            ]
        }
```

3) endpoint: HEAD /todos/:id/tasksof

Performing HEAD /todos/:id/tasksof with non-existing TODO ID (90) returned header data in the JSON message. This is incorrect behavior. Should return an error with corresponding status message. ##BUG (Refer to .docx)

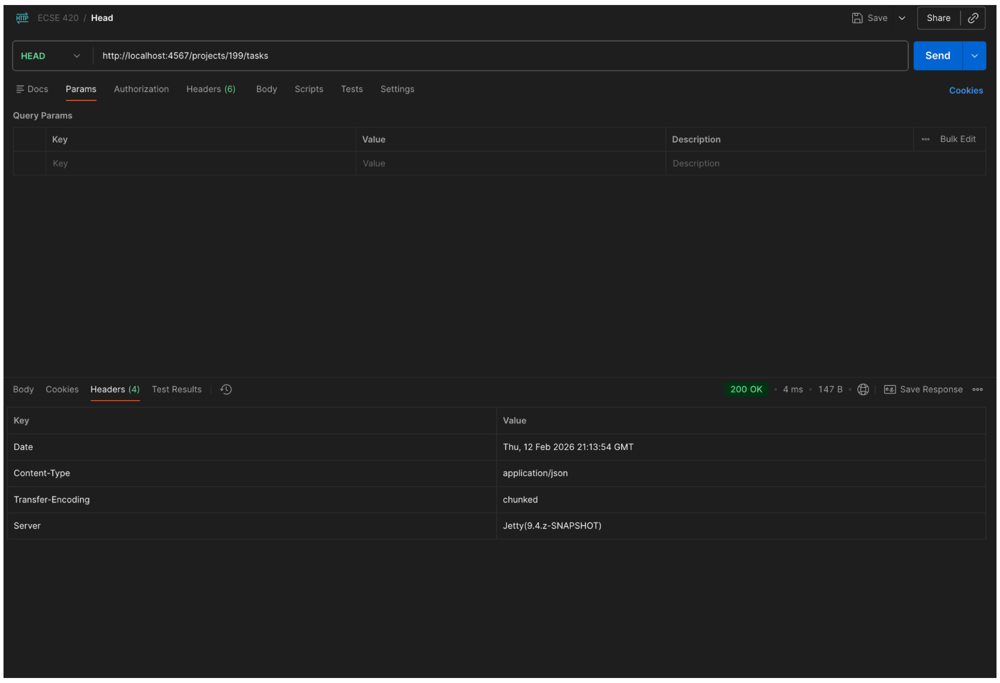

HEAD http://localhost:4567/todos/90/tasksof

200 OK · 4 ms · 147 B

Key	Value
Date	Thu, 12 Feb 2026 21:12:19 GMT
Content-Type	application/json
Transfer-Encoding	chunked
Server	Jetty(9.4.z-SNAPSHOT)

4) endpoint: HEAD /projects/:id/tasks

Performing HEAD /projects/:id/tasks with non-existing PROJECT ID (199) returned header data in the JSON message. This is incorrect behavior. Should return an error with corresponding status message. ##BUG (Refer to .docx)


ECSE 420 / Head

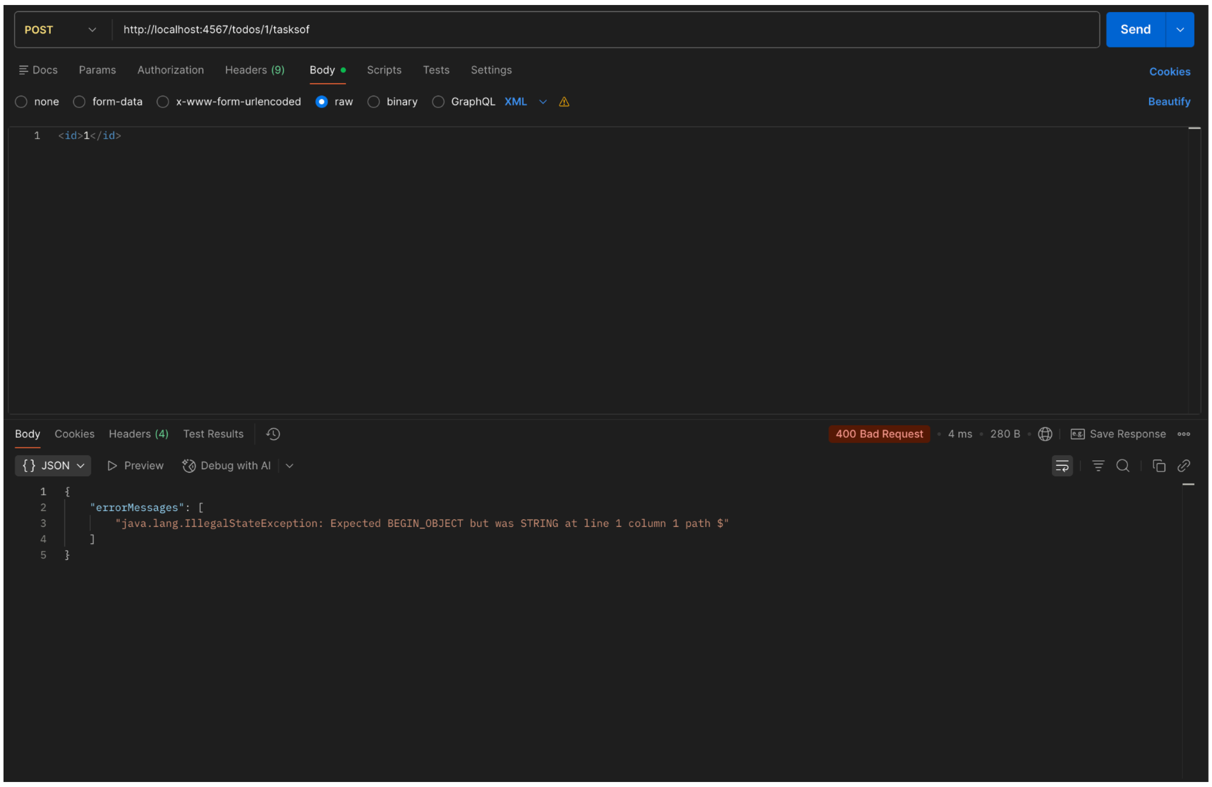
HEAD http://localhost:4567/projects/199/tasks

200 OK · 4 ms · 147 B

Key	Value
Date	Thu, 12 Feb 2026 21:13:54 GMT
Content-Type	application/json
Transfer-Encoding	chunked
Server	Jetty(9.4.z-SNAPSHOT)

5) endpoint: POST /todos/:id/tasksof

Performing POST /todos/:id/tasksof with a valid TODO ID and correct format body in XML format (<id>1</id>) returned an error (cannot parse XML body).

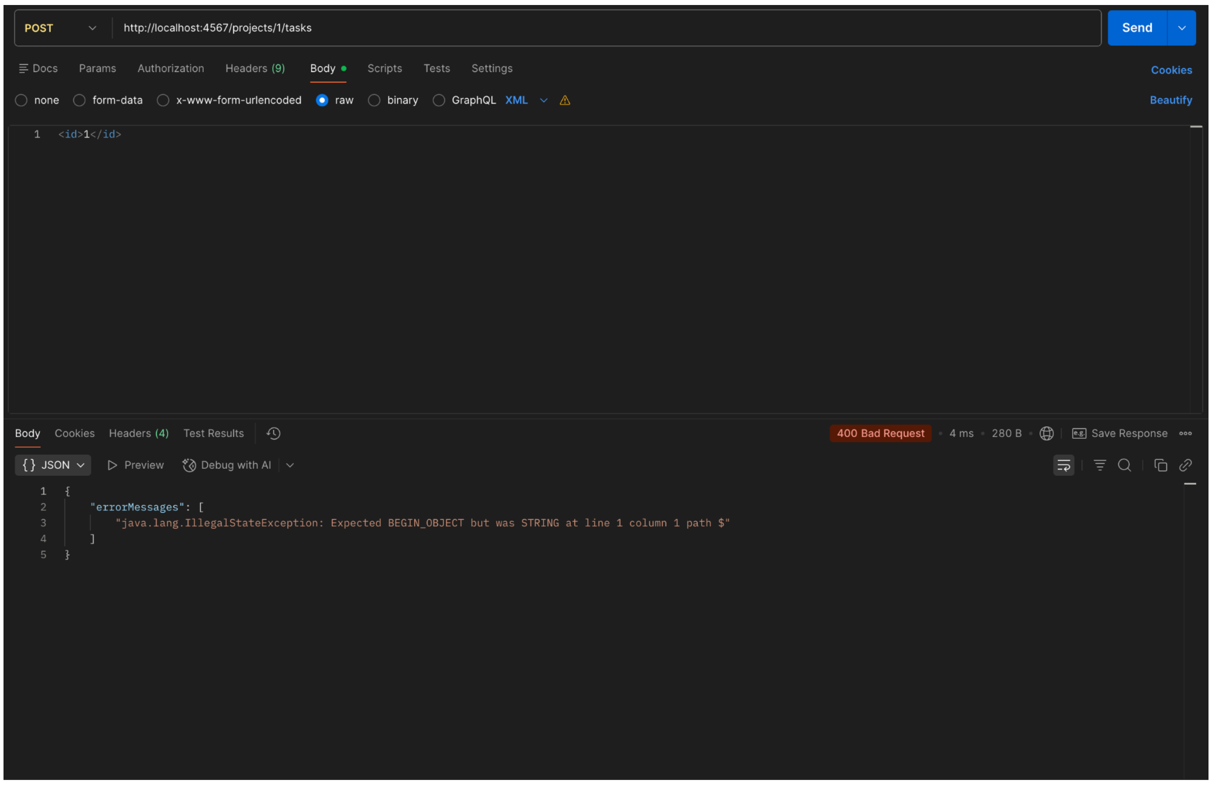


```
POST http://localhost:4567/todos/1/tasksof
Body: raw, XML

<id>1</id>
```

```
400 Bad Request

{
    "errorMessages": [
        "java.lang.IllegalStateException: Expected BEGIN_OBJECT but was STRING at line 1 column 1 path $"
    ]
}
```

6) endpoint: POST /projects/:id/tasks

Performing POST /projects/:id/tasks with a valid PROJECT ID and correct format body in XML format (<id>1</id>) returned an error (cannot parse XML body).



http://localhost:4567/projects/1/tasks

```
<id>1</id>
```

```
{
    "errorMessages": [
        "java.lang.IllegalStateException: Expected BEGIN_OBJECT but was STRING at line 1 column 1 path $"
    ]
}
```

Ideas/Plans for Future Sessions

Cover relationships between todos/categories and categories/projects

Test duplicate relationship creation (add same project twice)

Test deleting relationship twice

Test performance with 100+ todos

Test content-type mismatch (send XML header but JSON body)

Areas of Potential Risk

BUG #1, #2, #3, #4 are data concerns, they return data for an invalid request. The user should not have access to this data.

BUG #5 and #6 are format handling risks. XML not parsed even though API suggests it is supported. Could indicate incomplete implementation.

Capabilities

Querying and deleting Projects/Tasks and Todos/Tasksof relationships in JSON

Refer to Capabilities Script: projectsCapabilities.sh and todosCapabilities.sh

Summary of session findings:

This session focused on testing the relationship endpoints between TODOs and PROJECTs. Six bugs were identified: HEAD requests on relationship endpoints with non-existing IDs incorrectly return header data instead of an error response (bugs #1–4), and POST requests to /todos/:id/tasksof and /projects/:id/tasks fail to parse XML bodies despite the API advertising XML support (bugs #5–6).

The key risks identified are unauthorized data exposure on invalid HEAD requests and incomplete XML parsing support, both indicating potentially unfinished implementation. Future sessions should explore duplicate relationship creation, deletion edge cases, and content-type mismatches.