Session Notes - 4

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:
java -jar runTodoManagerRestAPI-1.5.5.jar

Area:
Capabilities and areas of potential instability (documented and undocumented) as well as tests using data typical to the intended use of the application.

Environment:
OS Version: MacOS Sonoma 14.0, Screen Resolution: 3458 x 2234

Performed by:

Session Participants	Student Id	Email address
Abiola Olaniyan	260983951	abiola.olaniyan@mail.mcgill.ca
Mihail Calitoiu	260972537	mihail.calitoiu@mail.mcgill.ca

Time:
Session start: 2:50 PM, October 20, 2023
Session end: 3:35 PM, October 20, 2023

Roles:
Abiola - Tester, OS Version: Windows 10, Screen Resolution: 1920 x 1080
Mihail - Note taker

Summary of session:

The main objective of this exploratory testing session was to test the endpoints for the project entity, as well as the interoperability between projects and categories. Each call was done in a

consecutive fashion without shutting down the API or deleting entities between calls (unless required for a test).

The first phase involved executing calls to endpoints for projects without the use of an ID for the three types available GET, HEAD and POST. These were straightforward calls that included verifying the GUI and responses reported by Postman in parallel. After each call, a screenshot of the associated Postman result was attached to the table listing the calls.

The next phase was running through the available types when using an ID in the URL. Since some of the results were expected to be the same (e.g. PUT and POST), we used these endpoints to attempt invalid operations such as using malformed request JSON bodies or manipulating the types of fields the server expected to receive.

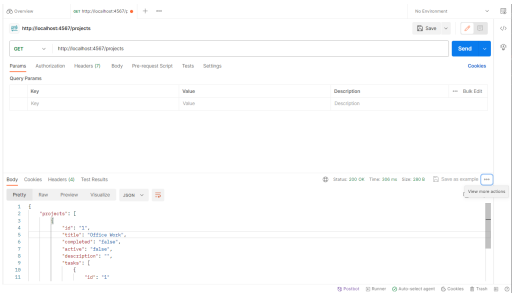
Finally, we used the category relationship endpoints linked to projects to make use of the remaining time in the session. This comprised of many GET calls to verify the links between the entities.

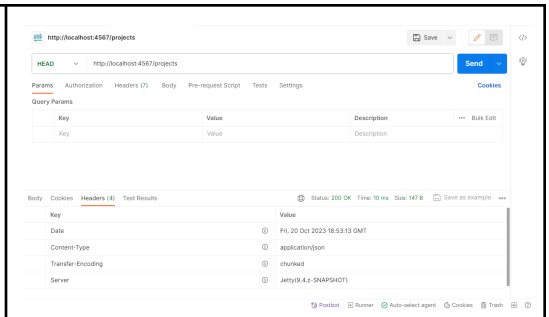
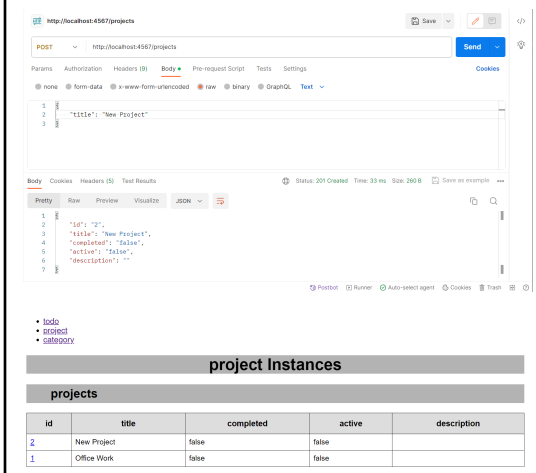
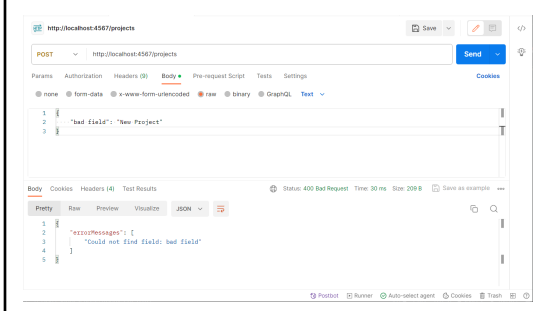
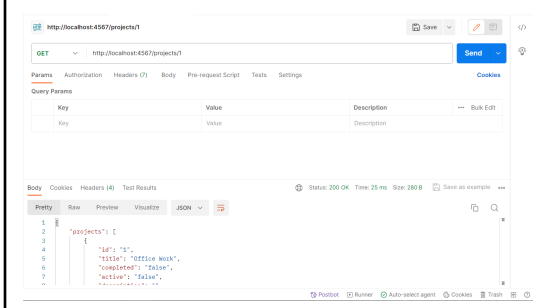
List of concerns identified in session

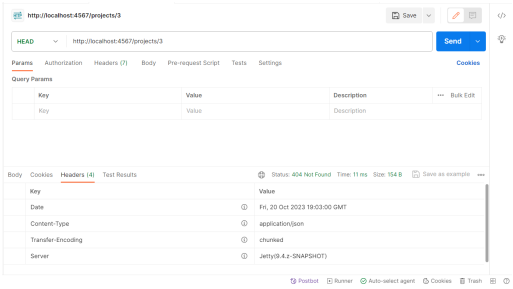
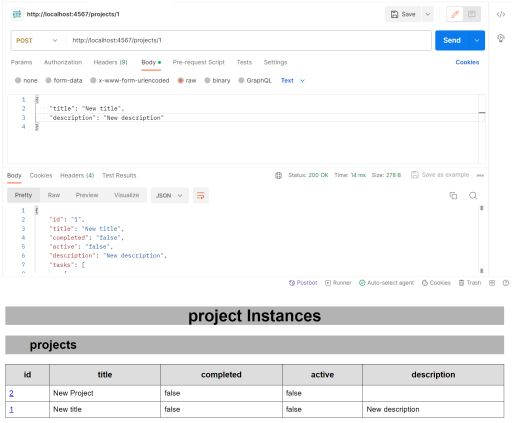
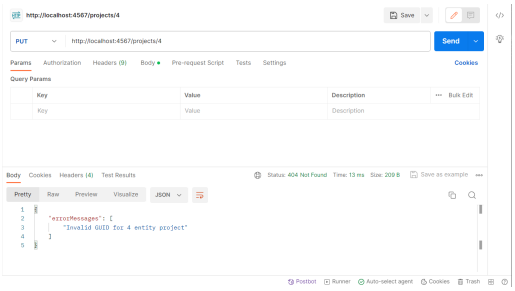
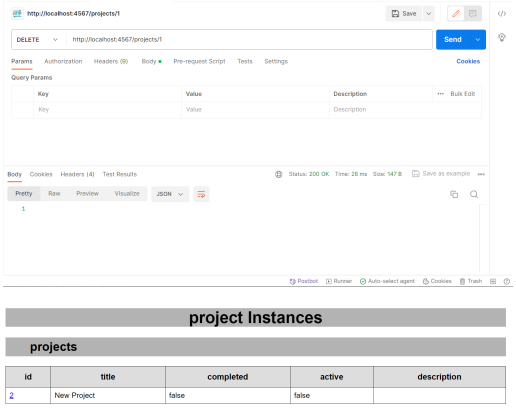
After verifying that the application is able to handle complex relationship handling, we found no areas of concern.

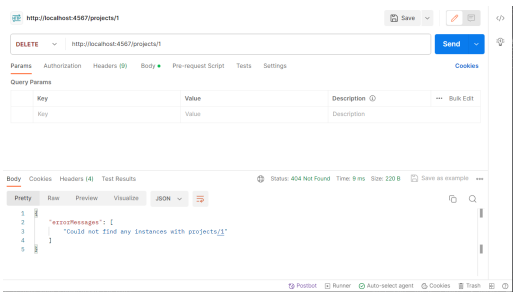
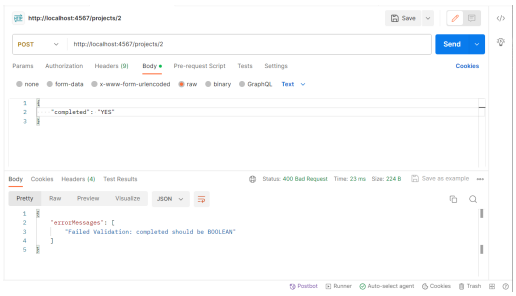
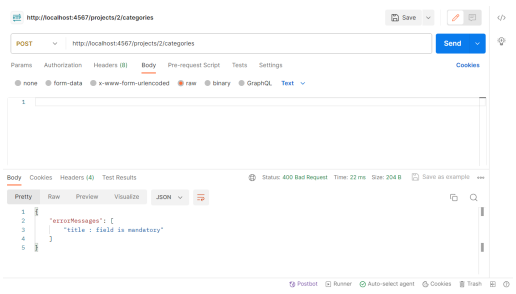
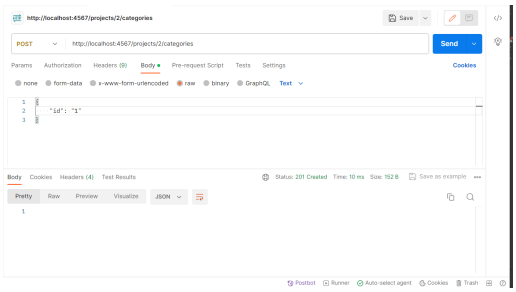
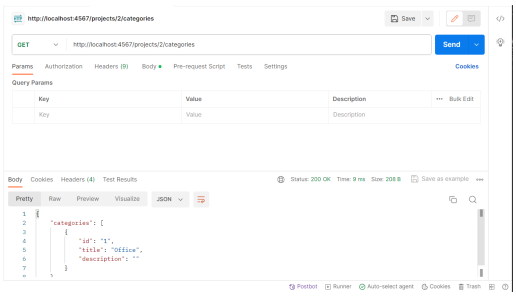
List of new testing ideas identified in session

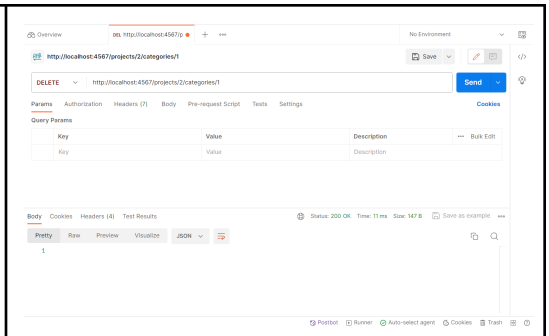
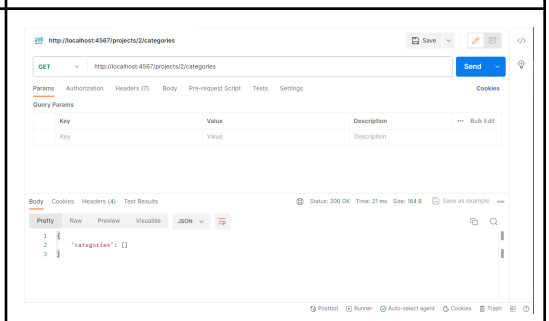
A testing idea that could be explored would be to use XML format when sending a request as opposed to JSON on Postman; that will likely be included in the individual classes made as part of the test suite created by the group members.

Time	Tester	Request	Type	Comment	Screenshot
2:50	Abiola	http://localhost:4567/projects/	GET	Verify if the initial project is returned	

2:53			HEAD	Verify that headers for projects call are returned	
2:55			POST	Verify the creation of a new project in GUI and Postman	
2:57			POST	Create another project with malformed field	
3:00		http://localhost:4567/projects/:id	GET	Get project with default id (1)	

3:03			HEAD	Get headers for project that doesn't exists (id = 3)	
3:05			POST	Update fields for project with id 1 + verify if GUI reflects changes	
3:07			PUT	Trying to amend fields for project that doesn't exist	
3:09			DELETE	Deleting original project (id = 1); GUI also shows change	

3:09			DELETE	Trying to delete project already deleted	
3:10			POST	Using the wrong type for a field	
3:12		http://localhost:4567/projects/:id/categories	POST	Create relationship between a category and project without an id for categories	
3:17			POST	Create relationship with proper id	
3:19			GET	Verify relationship was successfully created	

3:23		http://localhost:4567/projects/:id/categories/:id	DELETE	Delete the relationship between the category and project	
3:27		http://localhost:4567/projects/:id/categories	GET	Ensure that it was deleted for projects	
3:34		http://localhost:4567/categories/:id/projects	GET	Ensure that it was deleted on the other side for categories