Report Summaries

Summarizes deliverables

For each exploratory testing, we have the same charter:

```
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.

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.

-------------------------------------------
```

PART 1: Exploratory testing for ***todos***

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

- Based on the general charter, the purpose of this charter was to identify capabilities and potential issues of the "rest api todo list manager" related to ***todos***.
- All capabilities tested were different actions (GET, HEAD, PUT, POST, DELETE) on these commands:
    - /todos
    - /todos/:id
    - /todos/:id/categories
    - /todos/:id/categories/:id
    - /todos/:id/tasksof
    - /todos/:id/tasksof/:id
- The session notes were taken using Word, testing the API locally on Insomnia API system.
- The testing session was around 45 minutes.
- The reference files (screenshots) are directly included in the session notes to facilitate reading.
- The structure of test input was taken from the documentation, partially using JSON and partially using XML

- The script or program used to test capability is mainly included in the reference pictures. Some are directly included in the session notes.
- The expected behavior of each capability according to the documentation is written in italics under each capability.
- The output results were compared with the documentation.
- Potential bugs were identified (potential risks are marked with PROBLEM keyword inside the session notes)

PART 2: Exploratory testing for ***projects***

Tester: Ying Fei Zang (261054638) ying.zang@mail.mcgill.ca

- Within 45 min, I had time to briefly explore Projects page and test the following API endpoints using Postman:
  - /projects
  - /projects/:id
  - /projects/:id/tasks
- After deleting a Projects instance using Postman, if the previous page was Projects page and we go back, the Project that should be deleted remains there.
- The table is not ordered based on ID
- Sometimes the API doc does not tell enough on the appropriate actions that should be executed, for instance, when I want to do POST on /projects/:id/tasks
- All the concerns are listed with ** in projects session note.

PART 3: Exploratory testing for ***categories***

Tester: Yu Tong Hu (262051311) yu.tong.hu@mail.mcgill.ca

- Session notes were taking on Words and API used was Postman
- Testing session was 45mins
- The capability tested were:
  - /categories: GET, HEAD, POST
  - /categories/:id : GET, HEAD, PUT, POST, DELETE
  - /categories/:id/projects: GET, HEAD, POST
  - /categories/:id/projects/:id : DELETE
  - /categories/:id/todos: GET, HEAD, POST
  - /categories/:id/todos/:id : DELETE
- Structure of the tests was in JSON only
- GET: works normally when getting all categories or categories/:id.
- GET: the ordering of the categories seems random.
- GET: when getting the todos or projects in the categories, an empty list returns.
- HEAD: works normally with categories.

- POST: works normally when putting the correct body field.
- POST: when creating new categories, the ID assigned is in order.
- POST: if the json body value is not a string, it would convert it to a string, but integers are converted to a string of float.
- POST and DELETE: when trying to establish a relationship with existing category ID and existing todos ID or project ID, the server seems to not find the corresponding object. Possible bug.
- POST and PUT: seems both have the same functionality when working with categories/:id.
- PUT: when changing the content of a category with json error, the error message is not a server message.
- DELETE: works normally when deleting with categories/:id.
- OPTIONS/PATCH: doesn't do anything but displays different code.

Findings of exploratory testing

PART 1: Exploratory testing for ***todos***

Throughout my exploratory testing, I found out a few potential issues/bugs. Here is a summary:

- some input examples in documentation does not match with expected input.
- Some GET functionalities return unexpected results for a non-existing todo id.
- No id assignment freedom for the users
- Duplicates are allowed for categories and projects

I also came up with different testing ideas throughout the session, here is a brief (the full list can be found at the end of the session notes):

- Confirm we can add todos after deleting tasks
- Confirm we can update the fields of an existing todo
- Study the differences between PUT and POST
- Test invalid structure for input (ex: invalid JSON and XML)

PART 2: Exploratory testing for ***projects***

During exploratory testing for projects, we found out that:

- The description in API documentation sometimes causes confusion and is unclear.

- There exist some features which are not documented, such as the deletion of a project instance also leads to the deletion of the id that was associated with it.
- The title of the task is used instead of its id to create relation with a project instance.

PART 3: Exploratory testing for ***categories***

When entering the server for the first time, there are 2 existing categories:

- Office
- Home

Potential issues/bugs:

- User cannot assign IDs
- Duplicate titles are allowed
- Get calls display the categories in random order of ID.
- Head calls display the same headers for every type of calls
- Post calls skips the ID assignments when there are cases of error
  i.e., the last category ID would be 2 and after 4 error POST calls, the next correct category created with POST call would have ID 7.
- Post calls doesn't allowed linking relationships between categories to todos or projects
- The system will automatically convert Post call value to "title" key to a string value.
- Delete calls doesn't allowed the removal of relationships between categories to todos or projects
- Options calls and patch calls does not have functionalities

Testing ideas:

- Confirms we can update the fields of a categories
- Testing the XML and JSON input

Source code repository

The python file *RestAPITester.py* can be found in the corresponding folder 429-ProjectPT1. To run, it needs python, json library, unittest library and requests library. Before running the Tester, make sure to launch the rest api todo list manager with the command:

```
java -jar runTodoManagerRestAPI-1.5.5.jar
```

Structure of unit test suite

The unit test code can be found in `class APITester`, there are three big parts inside the tester: # Todos, # Categories, and # Projects. In total, there are 57 test modules. After the API can be accessed locally, we can run the test with the run button. The test starts with initialization to create a session by starting a connection. After running all the test modules in a random order, the test ends up closing the connection. For all test modules, the comment indicates the specific goal of the test.

PART 1 : Summary of #Todos test suite

- A test method for each type of call
- There are in total 30 tests for testing capabilities related to todos
- When we have unexpected behavior failing (API behavior is different from the documentation), there are two modules, one shows the failing, and the other shows the actual behavior. These modules are marked with the keyword UNEXPECTED (#Todos test 15, 16, 19, 20, 21, 22). These unexpected behaviors are the potential risk/bugs of the API.
- The fail test cases are there to show the failing of unexpected behaviors
- Modules 29 and 30 test with invalid XML and JSON input.

PART 2 : Summary of #Projects test suite

- A test method for each type of call
- There are in total 16 tests for testing capabilities related to projects
- The tests follow the order of API Documentation on Projects
- There were unexpected behaviors that failed some tests due to the lack of information of API documentation. However, to make sure that they have the behavior written in the doc, we adjust some tests accordingly.

PART 3 : Summary of #Categories test suit

- A test method for each type of call and API documentation
- There are in total 18 tests for testing capabilities related to categories
- Tests are in order of GET, HEAD, POST, PUT, DELETE, XML POST
- Test 17 and 18 are testing for XML input
- Test 11, 12, 15, 16 are incapable of testing for the output errorMessages due to json.loads not able to read the output

Findings of unit test suite execution

PART1: Exploratory testing for ***todos***

- Bugs:
    - When we do GET todos/:id/categories with a todo that does not exist. No error message shows up, we get a return body with a list of existing categories inside. Gives false information.
    - Similar bug for GET /todos/:id/tasksof, if the id does not exist. The output is wrong, it returns a list of unrelated projects instead of alerting the invalid id.
- Potential Issues:
    - Inability to restore: Once we delete that todo instance, we cannot force it to restore, since Posting again will just generate a random new id for that instance.
    - Inability to order the display of instances
    - Creation of unidirectional relationship:

        - when we POST /todos/:id/tasksof, we can find the projects through todo. However, when we go to projects, we do not see the corresponding todo.
        - Same thing for POST /todos/:id/categories, we can only view the relation from the todo side (GET todos/:id/categories). This is confusing for the users.
- Undocumented behavior:
    - Unexpected Post for /todos/:id/categories:
        - For POST, if the input id for category already exists, the POST request does not update the fields with the input, it only returns the existing instance of category which has the input id. No error messages.
    - Difference between PUT and POST /todos/:id for update project 'tasks' field:
        - For PUT, when we include 'tasks' field in the input, we can update fields of a project with no error messages (see test 11).
        - u For POST, we cannot update 'tasks' field for a project (see test 12).

PART2: Exploratory testing for ***projects***

- ❖ Bugs: Not found during projects-related endpoints testing
- ❖ Potential Issues:
    - ➢ Meaningless id: when we used POST for /projects/:id/{tasks, categories}, instead of referring to them using their unique id, the title, a field which is not unique, was asked to perform the action. We believe that
    - ➢ Deletion of ID: When non valid JSON or XML codes were sent, an error was returned. However, when we POST a new project instance later on, we discover that the id of this new instance becomes inconsistent from the id of the previous project instance. EX: the list of id before error = [1,2]; the list of id after error = [1,2]; the list of id after we added a new project instance = [1,2,4].
    - ➢ Different behavior in between adding relation with categories and with todos:
      - Ability to add existing tasks (todos)
      - Inability to add existing categories; new category instance with new id will be generated
- ❖ Undocumented Behaviors:
    - ➢ PUT/POST for /projects/:id: PUT deletes the relations to categories and todos while POST does not. Needed to adjust projects test #7 due this behavior.

PART 3: Exploratory testing for ***categories***

- Bugs:
    - When using GET calls on /categories/:id/todos and /categories/:id/projects, an empty list of todos[] and projects[] are outputted respectively
    - When using GET calls on /categories, the order is randomized
- Potential Issues:
    - Inability to establish relationship between categories and todos with POST calls
    - Inability to establish relationship between categories and projects with POST calls

    - IDs are given in chronological order for each new category created with POST calls, regardless if the call is correct or has an error
- Undocumented behavior:
    - Unexpected Put calls and Post calls to display an error message not in the documentation when the body field has an error