![A logo on a black background

AI-generated content may be incorrect.]()**Group 47**

**ECSE 429: Part A Report**

**Diana Bodik (261108544)**

**Reswanth Reji Pillai (261108990)**

**Suthiesan Subramaniam (261182188)**

### Table of Contents

[**Table of Contents**](#_heading=h.gjdgxs) **2**

[**Summary of Deliverables**](#_heading=h.1fob9te) **[3](#_heading=h.1fob9te)**

[**Findings of Exploratory Testing**](#_heading=h.3znysh7) **[4](#_heading=h.3znysh7)**

[**Structure of Unit Test Suite**](#_heading=h.2et92p0) **[6](#_heading=h.2et92p0)**

**[Source Code Repository](#_heading=h.tyjcwt) [7](#_heading=h.tyjcwt)**

[**Findings of Unit Test Suite Execution**](#_heading=h.3dy6vkm) **8**

**1. Deliverables Summary**

**Submitted items:**

* **Session Notes:** Three 45-minute exploratory sessions focusing respectively on Todos, Projects, and Categories.
* **Unit Test Suite:** Node.js test environment using Mocha and Chai for automated validation of REST API endpoints.
* **Video Demonstration:** [Link](https://www.dropbox.com/scl/fi/s1wfhbvvx1abyyvwzampd/UnitTestDemo.mp4?rlkey=dh5ev0qrv3lhpg0lo04j3lstn&st=md69ryyj&dl=0)
* **Source Code Repository:** [Link](https://github.com/jumpman786/ECSE429_TESTINGAPI)
* **Screenshots:** *(insert test result images here)* for normal and randomized runs.

**2. Exploratory Testing Findings**

**Session 1 — Todos Domain**

**Charter:** Investigate CRUD operations for Todos and verify compliance with documented API responses.  
**Observations:**

* Basic POST /todos and GET /todos operations worked consistently for JSON payloads.
* The API accepted creation requests even when missing a title field, resulting in incomplete records.
* Updating a Todo with an invalid ID returned 200 OK with an empty body instead of 404.  
  **New Test Ideas:** Add validation tests for null or excessively long titles.  
  **Risks:** Inconsistent response codes may break client integrations.  
  **Summary:** CRUD is functional but error-handling differs from documentation.

**Session 2 — Projects Domain**

**Charter:** Explore project creation, Todo linkage, and cascading deletions.  
**Observations:**

* Projects can be created using POST /projects, but numeric project names are accepted, which is not defined in the documentation.
* Linking a Todo to a Project succeeded even when the Todo ID was invalid.
* Deleting a Project removed associated Todos automatically without user confirmation.  
  **New Test Ideas:** Validate proper unlinking and relationship persistence after deletions.  
  **Risks:** Cascade deletion may cause unintended data loss.  
  **Summary:** Project management is mostly stable but with weak relationship validation.

**Session 3 — Categories Domain**

**Charter:** Assess category creation, retrieval, and linkage with Todos.  
**Observations:**

* POST /categories worked for JSON but failed silently when missing a title.
* The API returned outdated results due to caching when performing quick sequential updates.
* XML output lacked the id attribute despite documentation stating it should appear.  
  **New Test Ideas:** Add timing-based tests to verify cache invalidation.  
  **Risks:** Category updates may not sync properly between JSON and XML modes.  
  **Summary:** Core CRUD works but XML implementation and cache consistency need attention.

**General Insights**

* JSON responses were more reliable than XML.
* Incorrect status codes (e.g., 200 for not-found cases) occurred across multiple endpoints.
* Eight unique bugs were logged in the Bug Summary folder.
* Randomized tests exposed persistence issues due to state not resetting between API calls.

**3. Structure of Unit Test Suite**

**Technology Stack:** Node.js (v18) with Mocha and Chai.

**Organization by Modules:**

| **Folder / File** | **Purpose** |
| --- | --- |
| test/JSONEndpoints/Todo/todos\_json.spec.js | Validates CRUD behavior of /todos endpoints using JSON payloads. |
| test/JSONEndpoints/Categories/categories\_json.spec.js | Tests category creation, linking, and retrieval in JSON format. |
| test/XMLEndpoints/xml\_lists.spec.js | Checks correctness and completeness of XML responses for all domain objects. |
| test/serviceUp.spec.js | Confirms the API server is running before any test execution. |
| test/testConstants.js | Centralized configuration for URLs, ports, and common assertions. |
| scripts/run-shuffle.js | Executes all .spec.js tests in random order using Node’s child\_process.spawn() and the Mocha CLI. |

**Test Coverage Highlights:**

* Separate test suites for JSON and XML payloads.
* Includes malformed request checks for both formats.
* Each test module independently initializes and resets system state.
* Random-order execution confirms independence and idempotence of all modules.

**4. Source Code Repository**

*https://github.com/jumpman786/ProjectPartA\_Group47*

**Repository Layout:**

ECSE429\_TESTINGAPI/

├── node\_modules/

├── scripts/

│ └── run-shuffle.js

├── test/

│ ├── JSONEndpoints/

│ │ ├── Categories/categories\_json.spec.js

│ │ ├── Projects/ # placeholder for project tests

│ │ └── Todo/todos\_json.spec.js

│ ├── XMLEndpoints/xml\_lists.spec.js

│ ├── serviceUp.spec.js

│ └── testConstants.js

├── package.json

└── package-lock.json

* scripts/ contains utilities such as the shuffle runner for random test order.
* test/ includes all structured test modules categorized by endpoint and data format.
* Each .spec.js file runs automatically when executing npm test or the shuffle script.

**5. Test Execution Results**

**Regular Run:**

* All JSON endpoint tests passed.
* XML test failed for missing id tag in category responses.
* Negative test cases for invalid deletes occasionally returned success codes.

**Shuffled Run:** *(insert screenshot here)*

* The random-order test runner (run-shuffle.js) executed all .spec.js files in a non-deterministic order.
* Tests remained stable except for one interdependency between project and category creation modules.

**Bugs Logged:**

1. Todo created without title persists in system.
2. Invalid Project ID returns 200 OK.
3. Missing id in Category XML output.
4. Silent failure on unlinking Todo from Category.
5. Deletion cascade removes linked Todos unexpectedly.
6. Caching delay on Category retrieval.
7. Randomized run exposes state leakage across modules.
8. XML malformed payload not properly rejected.

**Conclusion**

Part A successfully identified both documented and undocumented behaviors within the REST API Todo Manager. The Mocha/Chai-based unit test suite validated JSON and XML consistency, error handling, and API independence. Randomized execution through the shuffle script confirmed robustness for most modules, though a few state-related issues were uncovered. Overall, the project achieved its goal of validating system stability, documenting defects, and establishing a reliable automated testing foundation for Part B.