

## ECSE Software Validation Term Project

### Part A Exploratory Testing of Rest API

#### Application Under Test

The application under test is a “rest api todo list manager” which may be run as a local host.

A copy of the application is available in myCourses content under topic project.

The application is made available by Alan Richardson and can be found online at:

<https://github.com/eviltester/thingifier/releases>

Launch the rest api todo list manager with the command:

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

Basic documentation about the api todo list manager is found at:

<http://localhost:4567/docs>

A swagger description of the rest api todo list manager is referenced in the documentation and can be found at the link:

<http://localhost:4567/docs/swagger>

Individuals interested in learning about or using Swagger may benefit from a free individual account at:

<https://swagger.io/tools/>

#### Exploratory Testing

Team members are required to use Charter Driven Session Based Exploratory Testing to study the behavior of the “rest api todo list manager”.

Team members may do exploratory testing sessions individually or in pairs. All team members must participate in exploratory testing sessions.

Exploratory testing sessions should be timeboxed at 45 minutes.

The charter is:

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.

- If teams have one member, they should focus on capabilities related to todos.
- If teams have two members, they should focus on capabilities related to todos and projects.
- If teams have three members, they should focus on capabilities related to todos and projects and categories.
- If team members have four members, they should focus on capabilities related to todos and projects and categories and the interoperability of these capabilities.

Deliverables from each session should include:

- Session notes
  - Explicitly reference any scripts, programs, screen shots, video clips, spreadsheets or any other files used in or created during the session.
  - Include name and student id number and email address of session participants.
- Include any files created in the session.
- Summary of session findings
  - This is a bullet list created when the session findings are reviewed.
  - This list answers the question what we learned.
- List of concerns identified in session.
- List of new testing ideas identified in session.

### Unit Test Suite

Team members are required to implement a suite of unit tests using an open-source unit test tool such as JUnit. The unit test tool selected depends on which programming language the team chooses to create the unit tests. All unit tests from a team must be in the same suite using the same programming language and the same unit testing tool.

Teams must not use automatically generated unit testing code.

*At least one separate unit test module is required for each API identified in the exploratory testing. This must include at least one unit test module for each documented API and at least one unit test module for each undocumented API discovered during exploratory testing.*

Confirm the API does what it is supposed to do.

Identify bugs in the API implementation if the actual behavior is different from the documented behavior.

Note for cases when the API behavior is different from the documentation please include two separate modules one showing the expected behavior failing and one showing the actual behavior working. *The API may not behave as documented, but it may still allow the operation to succeed in an undocumented manner.*

Confirm the API does not have unexpected side effects. *It is sufficient to show that changes to data in the system are restricted to those which should change based on the API operation.*

Confirm that each API can generate payloads in JSON or XML.

Confirm that command line queries function correctly.

Confirm return codes are correctly generated.

Unit test modules must:

- Ensure the system is ready to be tested
- Save the system state
- Set up the initial conditions for the test
- Execute the tests
- Assess correctness
- Restore the system to the initial state
- Run in any order
- Use clean, well-structured code follow guidelines of Bob Martin on Clean Code.

### Additional Unit Test Considerations

Ensure unit tests fail if service is not running.

Include at least one test to see what happens if a JSON payload is malformed.

Include at least one test to see what happens if an XML payload is malformed.

For each API *identified in the exploratory testing* include tests of invalid operations, for example, attempting to delete an object which has already been deleted.

### Additional Bug Summary Considerations

The project team should define a form used to collect bug information which includes at least the following elements.

- Executive summary of bug in 80 characters or less
- Description of bug
- Potential impact of bug on operation of system
- Steps to reproduce the bug

Teams can choose any tool to track bug information.

### Unit Test Suite Video

Show video of all unit tests running in the teams selected development environment.

Include demonstration of tests run in random order determined using pseudo random number generation to dictate the order.

### Written Report

Target report size is between 5 and 10 pages.

Summarizes deliverables.

Describes findings of exploratory testing.

Describes structure of unit test suite.

Describes source code repository.

Describe findings of unit test suite execution.