# **Practical: API Testing Research Assessment Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Functionality** | API tests demonstrate comprehensive & robust coverage on the following:   * CRUD (create, read, update & delete) functionality. * Authentication. * Validation rules. * Query parameters, i.e., filtering & sorting data. * Status codes, i.e., checking if a response returns 200. * The shape of the data, i.e., does the response data contain a specific column? | API tests demonstrate clear & detailed coverage on the following:   * CRUD (create, read, update & delete) functionality. * Authentication. * Validation rules. * Query parameters, i.e., filtering & sorting data. * Status codes, i.e., checking if a response returns 200. * The shape of the data, i.e., does the response data contain a specific column? | API tests demonstrate coverage on the following:   * CRUD (create, read, update & delete) functionality. * Authentication. * Validation rules. * Query parameters, i.e., filtering & sorting data. * Status codes, i.e., checking if a response returns 200. * The shape of the data, i.e., does the response data contain a specific column? | API tests does not, or does not fully demonstrate coverage on the following:   * CRUD (create, read, update & delete) functionality. * Authentication. * Validation rules. * Query parameters, i.e., filtering & sorting data. * Status codes, i.e., checking if a response returns 200. * The shape of the data, i.e., does the response data contain a specific column? |
| **Code Elegance** | API tests thoroughly demonstrate code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Sufficient modularity. * Idiomatic use of control flow, data structures and in-built functions. * Adheres to an OO architecture. * Formatted code. * No dead or unused code. * Database configured for testing environment. | API tests clearly demonstrate code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Sufficient modularity. * Idiomatic use of control flow, data structures and in-built functions. * Adheres to an OO architecture. * Formatted code. * No dead or unused code. * Database configured for testing environment. | API tests demonstrate code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Sufficient modularity. * Idiomatic use of control flow, data structures and in-built functions. * Adheres to an OO architecture. * Formatted code. * No dead or unused code. * Database configured for testing environment. | API tests do not or do not fully demonstrate code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Sufficient modularity. * Idiomatic use of control flow, data structures and in-built functions. * Adheres to an OO architecture. * Formatted code. * No dead or unused code. * Databases configured for testing environment. |
| **Documentation & Git Usage** | README file contains thorough evidence of how to setup the environment for development & run the tests.  Git commit messages are comprehensively formatted & reflect the functionality changes in succinct detail. | README file contains clear evidence of how to setup the environment for development & run the tests.  Git commit messages are clearly formatted & reflect the functionality changes in substantial detail. | README file contains evidence of how to setup the environment for development & run the tests.  Git commit messages are formatted & reflect the functionality changes in detail. | README file does not or does not fully contain evidence of how to setup the environment for development & run the tests.  Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes. |

# **Practical: API Testing Research**

Name:

Date:

Learner ID:

Assessor’s Name:

Assessor’s Signature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Out Of** | **Weighting** | **Final Result** |
| Functionality | 10 | 60 |  |
| Code Elegance | 10 | 30 |  |
| Documentation & Git Usage | 10 | 10 |  |
| **Final Result** | | | /100 |
| **This assessment is worth 20% of the final mark for the Introductory Application Development Concepts course.** | | | |

**Feedback:**

Functionality:

Code Elegance:

Documentation & Git Usage: