# **Project 1: Node.js REST API Assessment Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Functionality** | REST API contains comprehensive & robust evidence on the following:   * REST API is developed using Node.js & can run locally without modification. * An appropriate number of collections & fields with different data types. * Separate controller & route file for each collection. * Custom validation when creating & updating a field. * Collections are seeded with a JSON file. * REST API version is v1. * Appropriate status code & message returned when performing CRUD operations if a query does not return any API data & if an endpoint does not exist. * Filter, sort & paginate REST API data. * POST, PUT & DELETE routes are protected. * Rate limit is 25 requests per minute. * REST API is deployed to Heroku. * REST API data is stored in a MongoDB Atlas database. | REST API contains clear & detailed evidence of functionality on the following:   * REST API is developed using Node.js & can run locally without modification. * An appropriate number of collections & fields with different data types. * Separate controller & route file for each collection. * Custom validation when creating & updating a field. * Collections are seeded with a JSON file. * REST API version is v1. * Appropriate status code & message returned when performing CRUD operations if a query does not return any API data & if an endpoint does not exist. * Filter, sort & paginate REST API data. * POST, PUT & DELETE routes are protected. * Rate limit is 25 requests per minute. * REST API is deployed to Heroku. * REST API data is stored in a MongoDB Atlas database. | REST API contains evidence on the following:   * REST API is developed using Node.js & can run locally without modification. * An appropriate number of collections & fields with different data types. * Separate controller & route file for each collection. * Custom validation when creating & updating a field. * Collections are seeded with a JSON file. * REST API version is v1. * Appropriate status code & message returned when performing CRUD operations if a query does not return any API data & if an endpoint does not exist. * Filter, sort & paginate REST API data. * POST, PUT & DELETE routes are protected. * Rate limit is 25 requests per minute. * REST API is deployed to Heroku. * REST API data is stored in a MongoDB Atlas database. | REST API does not, or does not fully contain evidence on the following:   * REST API is developed using Node.js & can run locally without modification. * An appropriate number of collections & fields with different data types. * Separate controller & route file for each collection. * Custom validation when creating & updating a field. * Collections are seeded with a JSON file. * REST API version is v1. * Appropriate status code & message returned when performing CRUD operations if a query does not return any API data & if an endpoint does not exist. * Filter, sort & paginate REST API data. * POST, PUT & DELETE routes are protected. * Rate limit is 25 requests per minute. * REST API is deployed to Heroku. * REST API data is stored in a MongoDB Atlas database. |
| **Code Elegance** | REST API thoroughly demonstrates code elegance on the following:   * Intermediate variables, idiomatic control flow, data structures & in-built functions, & sufficient modularity. * Functions & variables are named appropriately. * Efficient algorithmic approach. * REST API groups are named with a plural. * Filer header & in-line comments. * Formatted code using Prettier. * Prettier installed as a dev dependency. * No dead or unused code. * Database configured for production environment. | REST API clearly demonstrates code elegance on the following:   * Intermediate variables, idiomatic control flow, data structures & in-built functions, & sufficient modularity. * Functions & variables are named appropriately. * Efficient algorithmic approach. * REST API groups are named with a plural. * Filer header & in-line comments. * Formatted code using Prettier. * Prettier installed as a dev dependency. * No dead or unused code. * Database configured for production environment. | REST API demonstrates code elegance on the following:   * Intermediate variables, idiomatic control flow, data structures & in-built functions, & sufficient modularity. * Functions & variables are named appropriately. * Efficient algorithmic approach. * REST API groups are named with a plural. * Filer header & in-line comments. * Formatted code using Prettier. * Prettier installed as a dev dependency. * No dead or unused code. * Database configured for production environment. | REST API does not or does not fully demonstrate code elegance on the following:   * Intermediate variables, idiomatic control flow, data structures & in-built functions, & sufficient modularity. * Functions & variables are named appropriately. * Efficient algorithmic approach. * REST API groups are named with a plural. * Filer header & in-line comments. * Formatted code using Prettier. * Prettier installed as a dev dependency. * No dead or unused code. * Database configured for production environment. |
| **Documentation & Git Usage** | REST API documented in succinct detail using Postman.  README file contains thorough evidence of:   * URL to the REST API on Heroku. * URL to the REST API documentation on Postman. * How to setup the environment for development & deploy the REST API.   Git branches are thoroughly named with convention & contain the correct code relating to the functional requirement.  Git commit messages are comprehensively formatted & reflect the functionality changes in succinct detail. | REST API documented in substantial detail using Postman.  README file contains clear evidence of:   * URL to the REST API on Heroku. * URL to the REST API documentation on Postman. * How to setup the environment for development & deploy the REST API.   Git branches are mostly named with convention & contain the correct code relating to the functional requirement.  Git commit messages are clearly formatted & reflect the functionality changes in substantial detail. | REST API documented in detail using Postman.  README file contains evidence of:   * URL to the REST API on Heroku. * URL to the REST API documentation on Postman. * How to setup the environment for development & deploy the REST API.   Some git branches are named with convention & contain the correct code relating to the functional requirement.  Git commit messages are formatted & reflect the functionality changes in detail. | REST API not or not fully documented in detail using Postman.  README file does not or does not fully contain evidence of:   * URL to the REST API on Heroku. * URL to the REST API documentation on Postman. * How to setup the environment for development & deploy the REST API.   Git branches are not or are not fully named with convention & do not or do not fully contain the correct code relating to the functional requirement.  Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes. |

# **Project 1: Node.js REST API Marking Cover Sheet**

Name:

Date:

Learner ID:

Assessor’s Name:

Assessor’s Signature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Out Of** | **Weighting** | **Final Result** |
| Functionality | 10 | 40 |  |
| Code Elegance | 10 | 45 |  |
| Documentation & Git Usage | 10 | 15 |  |
| **Final Result** | | | /100 |
| **This assessment is worth 30% of the final mark for the Introductory Application Development Concepts course.** | | | |

**Feedback:**

**Functionality:**

**Code Elegance:**

**Documentation & Git Usage:**