# **Project Assessment Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **10-9** | **8-7** | **6-5** | **4-0** |
| **Functionality** | API contains comprehensive & robust evidence on the following:   * Application can run locally without modification. * Create, read, update & delete API data from at least three models. * Filter, sort and page API data from multiple models using query parameters. * Custom validation rules & messages applied to API data. * HTTP error code handling. * API endpoints tested using PHPUnit. * Deployed to & usable on Heroku. * API data stored in a MySQL (development) & PostgreSQL (production) database. * Database seeded with at least three JSON files. | API contains clear & detailed evidence of functionality on the following:   * Application can run locally without modification. * Create, read, update & delete API data from at least three models. * Filter, sort and page API data from multiple models using query parameters. * Custom validation rules & messages applied to API data. * HTTP error code handling. * API endpoints tested using PHPUnit. * Deployed to & usable on Heroku. * API data stored in a MySQL (development) & PostgreSQL (production) database. * Database seeded with at least three JSON files. | API contains evidence on the following:   * Application can run locally without modification. * Create, read, update & delete API data from at least three models. * Filter, sort and page API data from multiple models using query parameters. * Custom validation rules & messages applied to API data. * HTTP error code handling. * API endpoints tested using PHPUnit. * Deployed to & usable on Heroku. * API data stored in a MySQL (development) & PostgreSQL (production) database. * Database seeded with at least three JSON files. | API does not, or does not fully contain evidence on the following:   * Application can run locally without modification. * Create, read, update & delete API data from at least three models. * Filter, sort and page API data from multiple models using query parameters. * Custom validation rules & messages applied to API data. * HTTP error code handling. * API endpoints tested using PHPUnit. * Deployed to & usable on Heroku. * API data stored in a MySQL (development) & PostgreSQL (production) database. * Database seeded with at least three JSON files. |
| **Code Elegance** | API thoroughly demonstrates code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Appropriate use of control flow, data structures and in-built functions. * Sufficient code modularity. * Adheres to an OO architecture. * Efficient algorithmic approach, i.e., correct use of Eloquent. * API resource groups named with a plural nouns not verbs. * In-line comments explain complex logic. * Formatted code. * No dead or unused code. * Well-designed models containing fields, behaviours & relationships. * Databases configured for development & production environments. | API clearly demonstrates code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Appropriate use of control flow, data structures and in-built functions. * Sufficient code modularity. * Adheres to an OO architecture. * Efficient algorithmic approach, i.e., correct use of Eloquent. * API resource groups named with a plural nouns not verbs. * In-line comments explain complex logic. * Formatted code. * No dead or unused code. * Well-designed models containing fields, behaviours & relationships. * Databases configured for development & production environments. | API demonstrates code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Appropriate use of control flow, data structures and in-built functions. * Sufficient code modularity. * Adheres to an OO architecture. * Efficient algorithmic approach, i.e., correct use of Eloquent. * API resource groups named with a plural nouns not verbs. * In-line comments explain complex logic. * Formatted code. * No dead or unused code. * Well-designed models containing fields, behaviours & relationships. * Databases configured for development & production environments. | API does not or does not fully demonstrate code elegance on the following:   * Use of intermediate variables, i.e., no method calls as arguments. * Appropriate use of control flow, data structures and in-built functions. * Sufficient code modularity. * Adheres to an OO architecture. * Efficient algorithmic approach, i.e., correct use of Eloquent. * API resource groups named with a plural nouns not verbs. * In-line comments explain complex logic. * Formatted code. * No dead or unused code. * Well-designed models containing fields, behaviours & relationships. * Databases configured for development & production environments. |
| **Documentation & Git Usage** | README file contains thoroughly evidence of:   * URL to API on Heroku. * URL to API documentation on Postman. * How to setup the environment for development, run the tests & deploy the application.   API documented in succinct detail using Postman.  Git commit messages comprehensively formatted & reflect the functionality changes in succinct detail. | README file contains clear evidence of:   * URL to API on Heroku. * URL to API documentation on Postman. * How to setup the environment for development, run the tests & deploy the application.   API documented in substantial detail using Postman.  Git commit messages clearly formatted & reflect the functionality changes in substantial detail. | README file contains evidence of:   * URL to API on Heroku. * URL to API documentation on Postman. * How to setup the environment for development, run the tests & deploy the application.   API documented in detail using Postman.  Git commit messages formatted & reflect the functionality changes in detail. | README file does not or does not fully contain evidence of:   * URL to API on Heroku. * URL to API documentation on Postman. * How to setup the environment for development, run the tests & deploy the application.   API not or not full documented in detail using Postman.  Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes. |

# **Project Marking Cover Sheet**

Name:

Date:

Learner ID:

Assessor’s Name:

Assessor’s Signature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Out Of** | **Weighting** | **Final Result** |
| Functionality | 10 | 50 |  |
| Code Elegance | 10 | 40 |  |
| Documentation & Git Usage | 10 | 10 |  |
| **Final Result** | | | /100 |
| **This assessment is worth 80% of the final mark for the Introductory Application Development Concepts course.** | | | |

Feedback: