Project 1: Laravel API Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality	API contains comprehensive & robust evidence on the following: • Application runs locally without modification. • Models contain appropriate number of columns. • Controller for each model which contain CRUD functionality. • API version set to v1. • Custom validation when creating & updating data. • Database tables seeded with JSON files. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Appropriate data returned using API Resources. • Filter & sort using query parameters. • API data paginated. • Data stored in & removed from the cache • Protected routes using Sanctum. • API rate limit set to 25 requests. • Deployed to & usable on Heroku. • API data stored in a MySQL development database & PostgreSQL production database.	API contains clear & detailed evidence of functionality on the following: • Application runs locally without modification. • Models contain appropriate number of columns. • Controller for each model which contain CRUD functionality. • API version set to v1. • Custom validation when creating & updating data. • Database tables seeded with JSON files. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Appropriate data returned using API Resources. • Filter & sort using query parameters. • API data paginated. • Data stored in & removed from the cache • Protected routes using Sanctum. • API rate limit set to 25 requests. • Deployed to & usable on Heroku. • API data stored in a MySQL development database & PostgreSQL production database.	API contains evidence on the following: • Application runs locally without modification. • Models contain appropriate number of columns. • Controller for each model which contain CRUD functionality. • API version set to v1. • Custom validation when creating & updating data. • Database tables seeded with JSON files. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Appropriate data returned using API Resources. • Filter & sort using query parameters. • API data paginated. • Data stored in & removed from the cache • Protected routes using Sanctum. • API rate limit set to 25 requests. • Deployed to & usable on Heroku. • API data stored in a MySQL development database & PostgreSQL production database.	API does not, or does not fully contain evidence on the following: • Application runs locally without modification. • Models contain appropriate number of columns. • Controller for each model which contain CRUD functionality. • API version set to v1. • Custom validation when creating & updating data. • Database tables seeded with JSON files. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Appropriate data returned using API Resources. • Filter & sort using query parameters. • API data paginated. • Data stored in & removed from the cache • Protected routes using Sanctum. • API rate limit set to 25 requests. • Deployed to & usable on Heroku. • API data stored in a MySQL development database & PostgreSQL production database.

Code Elegance	API thoroughly demonstrates code elegance on the following: • Use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures and in-built functions. • Adheres to an OO architecture. • Efficient algorithmic approach, i.e., correct use of Eloquent. • API resource groups named with a plural noun not verb. • In-line comments explain complex logic. • Formatted code. • No dead or unused code. • 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. • Idiomatic use of control flow, data structures and in-built functions. • Adheres to an OO architecture. • Efficient algorithmic approach, i.e., correct use of Eloquent. • API resource groups named with a plural noun not verb. • In-line comments explain complex logic. • Formatted code. • No dead or unused code. • Databases configured for development & production environments.	API demonstrates code elegance on the following: • Use of intermediate variables, i.e., no method calls as arguments. • Idiomatic use of control flow, data structures and in-built functions. • Adheres to an OO architecture. • Efficient algorithmic approach, i.e., correct use of Eloquent. • API resource groups named with a plural noun not verb. • In-line comments explain complex logic. • Formatted code. • No dead or unused code. • 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. • Idiomatic use of control flow, data structures and in-built functions. • Adheres to an OO architecture. • Efficient algorithmic approach, i.e., correct use of Eloquent. • API resource groups named with a plural noun not verb. • In-line comments explain complex logic. • Formatted code. • No dead or unused code. • Databases configured for development & production environments.
Documentation & Git Usage	API documented in succinct detail using Postman. README file contains thoroughly evidence of: • URL to API application on Heroku. • URL to API documentation on Postman. • How to setup the environment for development & deploy the application. Git commit messages comprehensively formatted & reflect the functionality changes in succinct detail.	API documented in substantial detail using Postman. README file contains clear evidence of: • URL to API application on Heroku. • URL to API documentation on Postman. • How to setup the environment for development & deploy the application. Git commit messages clearly formatted & reflect the functionality changes in substantial detail.	API documented in detail using Postman. README file contains evidence of: • URL to API application on Heroku. • URL to API documentation on Postman. • How to setup the environment for development & deploy the application. Git commit messages formatted & reflect the functionality changes in detail.	API not or not full documented in detail using Postman. README file does not or does not fully contain evidence of: • URL to API application on Heroku. • URL to API documentation on Postman. • How to setup the environment for development & deploy the application. Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes.

Project 1: Laravel 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: