

## Practical Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality	<p>API contains comprehensive &amp; robust evidence on the following:</p> <ul style="list-style-type: none"> <li>• Application can run locally without modification.</li> <li>• Create, read, update &amp; delete API data from at least three models.</li> <li>• Filter, sort and page API data from multiple models using query parameters.</li> <li>• Custom validation rules &amp; messages applied to API data.</li> <li>• HTTP error code handling.</li> <li>• API endpoints tested using PHPUnit.</li> <li>• Deployed to &amp; usable on Heroku.</li> <li>• API data stored in a MySQL (development) &amp; PostgreSQL (production) database.</li> <li>• Database seeded with at least three JSON files.</li> </ul>	<p>API contains clear &amp; detailed evidence of functionality on the following:</p> <ul style="list-style-type: none"> <li>• Application can run locally without modification.</li> <li>• Create, read, update &amp; delete API data from at least three models.</li> <li>• Filter, sort and page API data from multiple models using query parameters.</li> <li>• Custom validation rules &amp; messages applied to API data.</li> <li>• HTTP error code handling.</li> <li>• API endpoints tested using PHPUnit.</li> <li>• Deployed to &amp; usable on Heroku.</li> <li>• API data stored in a MySQL (development) &amp; PostgreSQL (production) database.</li> <li>• Database seeded with at least three JSON files.</li> </ul>	<p>API contains evidence on the following:</p> <ul style="list-style-type: none"> <li>• Application can run locally without modification.</li> <li>• Create, read, update &amp; delete API data from at least three models.</li> <li>• Filter, sort and page API data from multiple models using query parameters.</li> <li>• Custom validation rules &amp; messages applied to API data.</li> <li>• HTTP error code handling.</li> <li>• API endpoints tested using PHPUnit.</li> <li>• Deployed to &amp; usable on Heroku.</li> <li>• API data stored in a MySQL (development) &amp; PostgreSQL (production) database.</li> <li>• Database seeded with at least three JSON files.</li> </ul>	<p>API does not, or does not fully contain evidence on the following:</p> <ul style="list-style-type: none"> <li>• Application can run locally without modification.</li> <li>• Create, read, update &amp; delete API data from at least three models.</li> <li>• Filter, sort and page API data from multiple models using query parameters.</li> <li>• Custom validation rules &amp; messages applied to API data.</li> <li>• HTTP error code handling.</li> <li>• API endpoints tested using PHPUnit.</li> <li>• Deployed to &amp; usable on Heroku.</li> <li>• API data stored in a MySQL (development) &amp; PostgreSQL (production) database.</li> <li>• Database seeded with at least three JSON files.</li> </ul>

Code Elegance	<p>API thoroughly demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> <li>• Use of intermediate variables, i.e., no method calls as arguments.</li> <li>• Appropriate use of control flow, data structures and in-built functions.</li> <li>• Sufficient code modularity.</li> <li>• Adheres to an OO architecture.</li> <li>• Efficient algorithmic approach, i.e., correct use of Eloquent.</li> <li>• API resource groups named with a plural nouns not verbs.</li> <li>• In-line comments explain complex logic.</li> <li>• Formatted code.</li> <li>• No dead or unused code.</li> <li>• Well-designed models containing fields, behaviours &amp; relationships.</li> <li>• Databases configured for development &amp; production environments.</li> </ul>	<p>API clearly demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> <li>• Use of intermediate variables, i.e., no method calls as arguments.</li> <li>• Appropriate use of control flow, data structures and in-built functions.</li> <li>• Sufficient code modularity.</li> <li>• Adheres to an OO architecture.</li> <li>• Efficient algorithmic approach, i.e., correct use of Eloquent.</li> <li>• API resource groups named with a plural nouns not verbs.</li> <li>• In-line comments explain complex logic.</li> <li>• Formatted code.</li> <li>• No dead or unused code.</li> <li>• Well-designed models containing fields, behaviours &amp; relationships.</li> <li>• Databases configured for development &amp; production environments.</li> </ul>	<p>API demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> <li>• Use of intermediate variables, i.e., no method calls as arguments.</li> <li>• Appropriate use of control flow, data structures and in-built functions.</li> <li>• Sufficient code modularity.</li> <li>• Adheres to an OO architecture.</li> <li>• Efficient algorithmic approach, i.e., correct use of Eloquent.</li> <li>• API resource groups named with a plural nouns not verbs.</li> <li>• In-line comments explain complex logic.</li> <li>• Formatted code.</li> <li>• No dead or unused code.</li> <li>• Well-designed models containing fields, behaviours &amp; relationships.</li> <li>• Databases configured for development &amp; production environments.</li> </ul>	<p>API does not or does not fully demonstrate code elegance on the following:</p> <ul style="list-style-type: none"> <li>• Use of intermediate variables, i.e., no method calls as arguments.</li> <li>• Appropriate use of control flow, data structures and in-built functions.</li> <li>• Sufficient code modularity.</li> <li>• Adheres to an OO architecture.</li> <li>• Efficient algorithmic approach, i.e., correct use of Eloquent.</li> <li>• API resource groups named with a plural nouns not verbs.</li> <li>• In-line comments explain complex logic.</li> <li>• Formatted code.</li> <li>• No dead or unused code.</li> <li>• Well-designed models containing fields, behaviours &amp; relationships.</li> <li>• Databases configured for development &amp; production environments.</li> </ul>
---------------	---	--	--	--

<b>Documentation &amp; Git Usage</b>	<p>README file contains thoroughly evidence of:</p> <ul style="list-style-type: none"> <li>• URL to API on Heroku.</li> <li>• URL to API documentation on Postman.</li> <li>• How to setup the environment for development, run the tests &amp; deploy the application.</li> </ul> <p>API documented in succinct detail using Postman.</p> <p>Git commit messages comprehensively formatted &amp; reflect the functionality changes in succinct detail.</p>	<p>README file contains clear evidence of:</p> <ul style="list-style-type: none"> <li>• URL to API on Heroku.</li> <li>• URL to API documentation on Postman.</li> <li>• How to setup the environment for development, run the tests &amp; deploy the application.</li> </ul> <p>API documented in substantial detail using Postman.</p> <p>Git commit messages clearly formatted &amp; reflect the functionality changes in substantial detail.</p>	<p>README file contains evidence of:</p> <ul style="list-style-type: none"> <li>• URL to API on Heroku.</li> <li>• URL to API documentation on Postman.</li> <li>• How to setup the environment for development, run the tests &amp; deploy the application.</li> </ul> <p>API documented in detail using Postman.</p> <p>Git commit messages formatted &amp; reflect the functionality changes in detail.</p>	<p>README file does not or does not fully contain evidence of:</p> <ul style="list-style-type: none"> <li>• URL to API on Heroku.</li> <li>• URL to API documentation on Postman.</li> <li>• How to setup the environment for development, run the tests &amp; deploy the application.</li> </ul> <p>API not or not full documented in detail using Postman.</p> <p>Git commit messages are not or are not fully formatted &amp; do not or do not reflect the functionality changes.</p>
--------------------------------------	---	--	--	--

# 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 20% of the final mark for the Introductory Application Development Concepts course.			

Feedback: