

Project Assessment Rubric

	10-9	8-7	6-5	4-0
Functionality	<p>Applications demonstrate comprehensive & robust evidence on the following:</p> <ul style="list-style-type: none"> • Routes protected with a token value. • Frontend application requests data from backend application using Axios. • Create, update & delete API data via modal. • Incorrectly formatted form fields are handled using validation error messages. • View API data in a table using an id and a variety of query parameters. • API data paginated across several pages. • UI styled with Reactstrap. • Applications deployed to Heroku. <p>End-to-end tests thoroughly demonstrate coverage of creating, updating, deleting & viewing API data.</p>	<p>Applications demonstrate clear & detailed evidence on the following:</p> <ul style="list-style-type: none"> • Routes protected with a token value. • Frontend application requests data from backend application using Axios. • Create, update & delete API data via modal. • Incorrectly formatted form fields are handled using validation error messages. • View API data in a table using an id and a variety of query parameters. • API data paginated across several pages. • UI styled with Reactstrap. • Applications deployed to Heroku. <p>End-to-end tests mostly demonstrate coverage of creating, updating, deleting & viewing API data.</p>	<p>Applications demonstrate evidence on the following:</p> <ul style="list-style-type: none"> • Routes protected with a token value. • Frontend application requests data from backend application using Axios. • Create, update & delete API data via a modal. • Incorrectly formatted form fields are handled using validation error messages. • View API data in a table using an id and a variety of query parameters. • API data paginated across several pages. • UI styled with Reactstrap. • Applications deployed to Heroku. <p>End-to-end tests demonstrate some coverage of creating, updating, deleting & viewing API data.</p>	<p>Applications does not, or does not fully demonstrate evidence on the following:</p> <ul style="list-style-type: none"> • Routes protected with a token value. • Frontend application requests data from backend application using Axios. • Create, update & delete API data via modal. • Incorrectly formatted form fields are handled using validation error messages. • View API data in a table using an id and a variety of query parameters. • API data paginated across several pages. • UI styled with Reactstrap. • Applications deployed to Heroku. <p>End-to-end tests does not or does not fully demonstrate coverage of creating, updating, deleting & viewing API data.</p>

Code Elegance	<p>Applications thoroughly demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> • Appropriate use of control flow, data structures and in-built functions. • Sufficient code modularity. • Components written as functional, not class. • Adheres to client-server architecture. • Header & in-line comments explain complex logic. • Formatted code using Prettier & npm script. • No dead or unused code. 	<p>Applications clearly demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> • Appropriate use of control flow, data structures and in-built functions. • Sufficient code modularity. • Components written as functional, not class. • Adheres to client-server architecture. • Header & in-line comments explain complex logic. • Formatted code using Prettier & npm script. • No dead or unused code. 	<p>Applications demonstrates code elegance on the following:</p> <ul style="list-style-type: none"> • Appropriate use of control flow, data structures and in-built functions. • Sufficient code modularity. • Components written as functional, not class. • Adheres to client-server architecture. • Header & in-line comments explain complex logic. • Formatted code using Prettier & npm script. • No dead or unused code. 	<p>Applications does not or does not fully demonstrate code elegance on the following:</p> <ul style="list-style-type: none"> • Appropriate use of control flow, data structures and in-built functions. • Sufficient code modularity. • Components written as functional, not class. • Adheres to client-server architecture. • Header & in-line comments explain complex logic. • Formatted code using Prettier & npm script. • No dead or unused code.
Documentation & Git Usage	<p>README file contains thoroughly evidence of:</p> <ul style="list-style-type: none"> • URL to applications on Heroku. • How to setup the environment for development, run the tests & deploy the applications. <p>Git branches thoroughly named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages comprehensively formatted & reflect the functionality changes in succinct detail.</p>	<p>README file contains clear evidence of:</p> <ul style="list-style-type: none"> • URL to applications on Heroku. • How to setup the environment for development, run the tests & deploy the applications. <p>Git branches mostly named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages clearly formatted & reflect the functionality changes in substantial detail.</p>	<p>README file contains evidence of:</p> <ul style="list-style-type: none"> • URL to applications on Heroku. • How to setup the environment for development, run the tests & deploy the applications. <p>Some git branches named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages formatted & 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"> • URL to applications on Heroku. • How to setup the environment for development, run the tests & deploy the applications. <p>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.</p> <p>Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes.</p>

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: