

Project 2: React CRUD 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"> • Register a new user via a form. • User can login and logout. • API data requested from API resource groups using Axios. • Create, update & delete API data. • View API data in a table. • Incorrectly formatted form fields are handled using validation error messages. • API data paginated across several pages. • UI styled with Reactstrap. • Application deployed to Heroku. • Register, login and logout functionality tested using Cypress. | <p>Applications demonstrate clear & detailed evidence on the following:</p> <ul style="list-style-type: none"> • Register a new user via a form. • User can login and logout. • API data requested from API resource groups using Axios. • Create, update & delete API data. • View API data in a table. • Incorrectly formatted form fields are handled using validation error messages. • API data paginated across several pages. • UI styled with Reactstrap. • Application deployed to Heroku. • Register, login and logout functionality tested using Cypress. | <p>Applications demonstrate evidence on the following:</p> <ul style="list-style-type: none"> • Register a new user via a form. • User can login and logout. • API data requested from API resource groups using Axios. • Create, update & delete API data. • View API data in a table. • Incorrectly formatted form fields are handled using validation error messages. • API data paginated across several pages. • UI styled with Reactstrap. • Application deployed to Heroku. • Register, login and logout functionality tested using Cypress. | <p>Applications does not, or does not fully demonstrate evidence on the following:</p> <ul style="list-style-type: none"> • Register a new user via a form. • User can login and logout. • API data requested from API resource groups using Axios. • Create, update & delete API data. • View API data in a table. • Incorrectly formatted form fields are handled using validation error messages. • API data paginated across several pages. • UI styled with Reactstrap. • Application deployed to Heroku. • Register, login and logout functionality tested using Cypress. |

| | | | | |
|--------------------------------------|---|--|--|---|
| Code Elegance | <p>Applications thoroughly 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. • Cypress & Prettier are installed as dev dependencies. • No dead or unused code. | <p>Applications clearly 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. • Cypress & Prettier are installed as dev dependencies. • No dead or unused code. | <p>Applications 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. • Cypress & Prettier are installed as dev dependencies. • 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. • Cypress & Prettier are installed as dev dependencies. • No dead or unused code. |
| Documentation & Git Usage | <p>README file contains thorough evidence of:</p> <ul style="list-style-type: none"> • URL to application on Heroku. • How to setup the environment for development, run Cypress tests & deploy the application. <p>Git branches are thoroughly named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages are 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 application on Heroku. • How to setup the environment for development, run Cypress tests & deploy the application. <p>Git branches are mostly named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages are 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 application on Heroku. • How to setup the environment for development, run Cypress tests & deploy the application. <p>Some git branches are named with convention & contain the correct code relating to the functional requirement.</p> <p>Git commit messages are 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 application on Heroku. • How to setup the environment for development, run Cypress tests & deploy the application. <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 2: React CRUD App 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 50% of the final mark for the Introductory Application Development Concepts course. | | | |

Feedback:

Functionality:

Code Elegance:

Documentation & Git Usage: