Project 2: React Assessment Rubric

	10-9	**8-7**	**6-5**	**4-0**
Functionality	Applications demonstrate comprehensive & robust evidence on the following: • 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. • Search for API data. • UI styled with Material Design. • Application deployed to AWS Amplify. • Register, login and logout functionality tested using Cypress.	Applications demonstrate clear & detailed evidence on the following: • 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. • Search for API data. • UI styled with Material Design. • Application deployed to AWS Amplify. • Register, login and logout functionality tested using Cypress.	Applications demonstrate evidence on the following: • 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. • Search for API data. • UI styled with Material Design. • Application deployed to AWS Amplify. • Register, login and logout functionality tested using Cypress.	Applications does not, or does not fully demonstrate evidence on the following: • 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. • Search for API data. • UI styled with Material Design. • Application deployed to AWS Amplify. • Register, login and logout functionality tested using Cypress.

Code Elegance	Applications thoroughly demonstrate code elegance on the following: • 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.	Applications clearly demonstrate code elegance on the following: • 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.	Applications demonstrate code elegance on the following: • 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.	Applications does not or does not fully demonstrate code elegance on the following: • 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:	README file contains thorough evidence of: • URL to application on AWS Amplify. • How to setup the environment for development, run Cypress tests & deploy the application. Git commit messages are comprehensively formatted & reflect the functionality changes in succinct detail.	README file contains clear evidence of: • URL to application on AWS Amplify. • How to setup the environment for development, run Cypress tests & deploy the application. Git commit messages are clearly formatted & reflect the functionality changes in substantial detail.	README file contains evidence of: • URL to application on AWS Amplify. • How to setup the environment for development, run Cypress tests & deploy the application. Git commit messages are formatted & reflect the functionality changes in detail.	README file does not or does not fully contain evidence of: • URL to application on AWS Amplify. • How to setup the environment for development, run Cypress tests & deploy the application. Git commit messages are not or are not fully formatted & do not or do not reflect the functionality changes.

Project 2: React App Marking Cover Sheet

Name:

Date:

Learner ID:

Assessor's Name:

Assessor's Signature:

Criteria	Out Of	Weighting	Final Result
Functionality	10	45	
Code Elegance	10	45	
Documentation & Git Usage	10	10	
Final Result			/100
This assessment is worth 40% of the final mark for the Year Two – Special Topic course.			

Feedback:

Functionality:

Code Elegance:

Documentation & Git Usage: