Project 1: REST/GraphQL APIs Assessment Rubric

	10-9	**8-7**	**6-5**	**4-0**
Functionality	APIs contains comprehensive & robust evidence on the following: • Developed using Node.js • Run locally without modification. • Three collections containing the correct number of fields. • Collections have separate controllers containing CRUD functionality. • Custom validation when creating & updating a document. • Collections seeded with JSON files. • API version set to v1. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Filter & sort using query parameters. • API data paginated. • Protected routes using JWT. • API rate limit set to 25 requests. • Deployed to & usable on AWS Amplify. • API data stored in a MongoDB Atlas database. • GraphQL API uses REST API. • Schemas return different API data.	APIs contains clear & detailed evidence of functionality on the following: • Developed using Node.js • Run locally without modification. • Three collections containing the correct number of fields. • Collections have separate controllers containing CRUD functionality. • Custom validation when creating & updating a document. • Collections seeded with JSON files. • API version set to v1. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Filter & sort using query parameters. • API data paginated. • Protected routes using JWT. • API rate limit set to 25 requests. • Deployed to & usable on AWS Amplify. • API data stored in a MongoDB Atlas database. • GraphQL API uses REST API. • Schemas return different API data.	APIs contains evidence on the following: • Developed using Node.js • Run locally without modification. • Three collections containing the correct number of fields. • Collections have separate controllers containing CRUD functionality. • Custom validation when creating & updating a document. • Collections seeded with JSON files. • API version set to v1. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Filter & sort using query parameters. • API data paginated. • Protected routes using JWT. • API rate limit set to 25 requests. • Deployed to & usable on AWS Amplify. • API data stored in a MongoDB Atlas database. • GraphQL API uses REST API. • Schemas return different API data.	APIs do not, or do not fully contain evidence on the following: • Developed using Node.js • Run locally without modification. • Three collections containing the correct number of fields. • Collections have separate controllers containing CRUD functionality. • Custom validation when creating & updating a document. • Collections seeded with JSON files. • API version set to v1. • Appropriate status code & message returned when performing CRUD actions. • Appropriate message returned when query does not return any data. • Filter & sort using query parameters. • API data paginated. • Protected routes using JWT. • API rate limit set to 25 requests. • Deployed to & usable on AWS Amplify. • API data stored in a MongoDB Atlas database. • GraphQL API uses REST API. • Schemas return different API data.

Code Elegance	APIs 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. • Functions and variables named appropriately. • Efficient algorithmic approach. • API resource groups named with a plural noun not verb. • Function header and in-line comments explain complex logic. • Formatted code using Prettier. • No dead or unused code. • Databases configured for production environment.	API sclearly 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. • Functions and variables named appropriately. • Efficient algorithmic approach. • API resource groups named with a plural noun not verb. • Function header and in-line comments explain complex logic. • Formatted code using Prettier. • No dead or unused code. • Databases configured for production environment.	APIs 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. • Functions and variables named appropriately. • Efficient algorithmic approach. • API resource groups named with a plural noun not verb. • Function header and in-line comments explain complex logic. • Formatted code using Prettier. • No dead or unused code. • Databases configured for production environment.	APIs 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. • Functions and variables named appropriately. • Efficient algorithmic approach. • API resource groups named with a plural noun not verb. • Function header and in-line comments explain complex logic. • Formatted code using Prettier. • No dead or unused code. • Databases configured for production environment.
Documentation &	README file contains thorough evidence of: • URL to the APIs on AWS Amplify. • How to setup the environment for development & deploy the application. Git commit messages are comprehensively formatted & reflect the functionality changes in succinct detail.	README file contains clear evidence of: • URL to the APIs on AWS Amplify. • How to setup the environment for development & deploy the application. Git commit messages are clearly formatted & reflect the functionality changes in substantial detail.	README file contains evidence of: • URL to the APIs on AWS Amplify. • How to setup the environment for development & 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 the APIs on AWS Amplify. • 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: REST/GraphQL APIs 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: