Marking Rubric

Functionality: REST API - 25%

Criteria	10-9	8-7	6-5	4-0
Environments	API seamlessly runs in both development and production environments without modification.	Minor issues in environment compatibility.	Significant issues in environment compatibility.	API struggles to run in development or production environments.
Model Development	Six models are developed, each with a minimum of four fields, demonstrating diverse data types.	Minor issues in model development, lacking diversity or meeting the minimum requirements.	Significant issues in model development, failing to meet the minimum requirements.	Models are missing or poorly developed.
Relationships Between Models	Six relationships between models are established, showcasing a clear understanding of associations.	Minor issues in establishing relationships.	Significant issues in establishing relationships.	Relationships between models are missing or incorrectly defined.
Enum Field in Model	One model includes an enum field, demonstrating knowledge and implementation.	Minor issues in implementing an enum field.	Significant issues in implementing an enum field.	No enum field is implemented.
Controller and Route Files	A controller and route file exist for each model,	Minor issues in the implementation	Significant issues in the implementation	Missing or incomplete

Criteria	10-9	8-7	6-5	4-0
	providing necessary operations (POST, GET all, GET one, PUT, DELETE).	of controllers and routes.	of controllers and routes.	controllers and routes.
Success/Failure Messages and Status Codes	Appropriate success and failure messages, along with correct status codes, are returned for each operation.	Minor issues in message or status code accuracy.	Significant issues in message or status code accuracy.	Messages or status codes are inconsistent or incorrect.
Filtering and Sorting	Data can be filtered and sorted using query parameters for all fields in ascending and descending order.	Minor issues in filtering or sorting implementation.	Significant issues in filtering or sorting implementation.	Filtering or sorting is not working as expected.
Pagination	Data is paginated using query parameters, with a default of 25 items per page.	Minor issues in pagination implementation.	Significant issues in pagination implementation.	Pagination is not functioning correctly.
Endpoint Validation and Index Route	Proper validation for each field using Joi during creation and updating. Index	Minor issues in validation or missing some validations.	Significant issues in validation or missing many validations.	Validation is incomplete or missing, and index route is not displaying routes.

Criteria	10-9	8-7	6-5	4-0
	route displays all existing routes.			
Database Storage and Deployment	Data is stored in a PostgreSQL database on Render. API is deployed as a web service on Render.	Minor issues in database storage or deployment.	Significant issues in database storage or deployment.	Data is not stored in PostgreSQL, or API deployment on Render is unsuccessful.

Functionality: CRUD Application - 20%

Criteria	10-9	8-7	6-5	4-0
Environments	Application seamlessly runs in the development environment without modification.	Minor issues in environment compatibility.	Significant issues in environment compatibility.	Application struggles to run in development environment.
Integration with REST API	Requests data from four API resource groups using Axios. Implements CRUD operations effectively.	Minor issues in integration or incomplete CRUD functionality.	Significant issues in integration or incomplete CRUD functionality.	Fails to integrate with the REST API or lacks essential CRUD functionality.
Create Operation	Successfully creates new data via the REST API.	Minor issues in the create operation.	Significant issues in the create operation.	Create operation is not working as expected.
Read/View Operation	Effectively reads/views data from the REST API.	Minor issues in the read/view operation.	Significant issues in the read/view operation.	Read/view operation is not working as expected.

Criteria	10-9	8-7	6-5	4-0
Update Operation	Successfully updates data via the REST API.	Minor issues in the update operation.	Significant issues in the update operation.	Update operation is not working as expected.
Delete Operation	Deletes REST API data with user confirmation using the confirm() JavaScript function.	Minor issues in the delete operation or confirmation process.	Significant issues in the delete operation or confirmation process.	Delete operation or confirmation process is not working as expected.
Validation Error Handling	Gracefully handles incorrectly formatted form field values with validation error messages.	Minor issues in validation error handling.	Significant issues in validation error handling.	Validation error handling is not working as expected.
UI Design and Style	UI is visually attractive with a coherent graphical theme and style using Reactstrap.	Minor issues in UI design or style.	Significant issues in UI design or style.	UI design or style is not visually attractive or lacks coherence.

Functionality: Scripts - 5%

Criteria	5	4	3-2	1-0
Run REST API and CRUD Application Locally	Successfully run both REST API and CRUD application locally without issues.	Minor issues or gaps in running either REST API or CRUD application locally.	Significant issues or incomplete execution of either REST API or CRUD application locally.	No or inadequate execution of both REST API and CRUD application locally.

Criteria	5	4	3-2	1-0
Create and Apply Prisma Migration	Successfully create and apply a migration using Prisma.	Minor issues or gaps in creating and applying Prisma migration.	Significant issues or incomplete creation and application of Prisma migration.	No or inadequate creation and application of Prisma migration.
Reset Database with Prisma	Successfully reset the database using Prisma.	Minor issues or gaps in resetting the database with Prisma.	Significant issues or incomplete database reset with Prisma.	No or inadequate database reset with Prisma.
Open Prisma Studio	Successfully open Prisma Studio.	Minor issues or gaps in opening Prisma Studio.	Significant issues or incomplete opening of Prisma Studio.	No or inadequate opening of Prisma Studio.
Format Code with Prettier	Successfully format code using Prettier.	Minor issues or gaps in code formatting with Prettier.	Significant issues or incomplete code formatting with Prettier.	No or inadequate code formatting with Prettier.

Code Quality and Best Practices - 45%

Criteria	10-9	8-7	6-5	4-0
.gitignore and .env.example	Proper use of Node.js .gitignore file. Environment variables are stored in .env.example.	Minor issues in .gitignore or .env.example.	Significant issues in .gitignore or .env.example.	Missing .gitignore or .env.example.
Appropriate Naming	Clear and meaningful names for files, variables, functions, and resource groups	Minor issues in naming conventions or clarity.	Significant naming issues impacting readability.	Naming conventions are not followed.

Criteria	10-9	8-7	6-5	4-0
	following best practices.			
API Best Practices	API security best practices are followed, including validation of content-type, sending x-content-type-options, x-frame-options, and content-security-policy. Secure communication through HTTPS is ensured.	Minor omissions in API security best practices.	Significant issues in API security best practices.	Critical omissions in API security, exposing vulnerabilities.
Idiomatic Use	Effective use of control flow, data structures, and in-built functions in an idiomatic manner.	Some instances of non-idiomatic code or inefficient use.	Significant non-idiomatic code affecting efficiency.	Poor use of control flow, data structures, or functions.
Efficient Algorithmic Approach	Implementation demonstrates a highly efficient algorithmic approach.	Some areas could be optimised for better efficiency.	Significant optimisation opportunities are missed.	Inefficient algorithmic approach throughout.
Sufficient Modularity	Code is well-organised and modular, enhancing maintainability and readability.	Some modules could be better organised for clarity.	Significant issues in modularity and organisation.	Lack of modularity, impacting code maintainability.
JSDoc Comments	Each file includes clear and comprehensive JSDoc header comments.	Minor issues or missing JSDoc comments in some files.	Significant issues with JSDoc comments throughout.	No JSDoc comments in files.
Code Formatting	Code is formatted using Prettier.	Minor issues in code	Significant issues in code	Code lacks proper

Criteria	10-9	8-7	6-5	4-0
	Prettier is installed as a development dependency.	formatting or Prettier setup.	formatting or Prettier setup.	formatting and Prettier setup.
No Dead or Unused Code	Codebase contains no dead or unused code.	Minor instances of dead or unused code.	Significant dead or unused code in the project.	Extensive dead or unused code affecting the project.

Documentation - 5%

Criteria	5	4	3-2	1-0
GitHub Project Board or Issues	Consistent and effective use of GitHub project board or issues for organising and prioritising development work.	Minor inconsistencies in GitHub project board or issues usage.	Significant issues in GitHub project board or issues usage.	GitHub project board or issues is not used or poorly maintained.
README Content	Well-documented repository README.md with required information.	Minor issues in documentation or missing some details.	Significant issues in documentation or missing key information.	README.md lacks essential information or is poorly structured.
Markdown Usage	Effective use of Markdown, including headings, bold text, code blocks, etc.	Some Markdown elements are used, but inconsistencies are present.	Limited use of Markdown or improper formatting.	Markdown is not used or severely misformatted.
Spelling and Grammar	Correct spelling and grammar throughout the documentation.	Minor spelling or grammatical errors are present.	Significant spelling or grammatical issues.	Numerous spelling and grammatical errors.

Criteria	5	4	3-2	1-0
Git Commit Messages	Git commit messages consistently reflect context and follow naming conventions.	Git commit messages generally reflect context and follow naming conventions.	Some Git commit messages reflect context but lack consistent naming conventions.	Git commit messages do not reflect context or follow naming conventions.

Marking Cover Sheet

Course Title:

Assignment Title:

Learner Name:

Learner ID:

Date Submitted:

Overall Comments:

[Insert overall comments and feedback here.]

Functionality: REST API: [/25]

Functionality: CRUD Application: [/20]

Functionality: Scripts: [/5]

Code Quality and Best Practices: [/45]

Documentation: [/5]

Total Marks (100%):

Assessor's Name:

Assessor's Signature: ____________________

Note: This cover sheet is to be completed and submitted with the assignment.