Course Title: Fundamentals in Backend Development

ASSESSMENT PLAN

Assessment Tools	Descriptions	Total Weightage	Type	Assessment Schedule
Quiz or Assignment 1	MCQ on Basic Node.js and NPM MCQ on General Development Skills	**50%**	Individual	End of Lesson 1
Quiz or Assignment 2	Project building an application using MVC with Express.js Project designing and provision a PostgreSQL database		Individual	End of Lesson 2
Quiz or Assignment 3	Project building a RESTful API with complete unit, integration, and e2e tests		Individual	End of Lesson 3
Quiz or Assignment 4	Project work on containerising and deploying the previously built app		Individual	End of Lesson 4
End of Module Project	Student to build a complete CRUD web application with Node.js, with extra functionalities that require event-driven processing and/or cron processing	**50%**	Individual	End of Lesson 4 and 5
	Total	**100%**		

END OF MODULE PROJECT (CAPSTONE PROJECT)

Project Title / Description	:	Implement a collaborative TODO-list application
Project Objective(s)	:	Create a TODO-list CRUD API with these below endpoints: ● [Public] A registration endpoint that would accept an email and password, and rejects any emails that have been registered before ● [Public] A login endpoint that would return a JSON Web token that could be used on authenticated endpoint ● [Auth-ed] CRUD endpoints for TODO lists: ○ A Create endpoint with the list being created belongs to and can only be accessed by the creator or anyone added to access the list ○ A GET all TODO-list endpoint that would return an array of TODO-lists with their titles based on who the currently authenticated user is ○ A GET a single TODO-list by its ID endpoint that would return the corresponding TODO-list together with all of the items in the list based on who the current authenticated user is. Returns 403 forbidden with a proper error JSON object if the user cannot access the list ○ A PUT/PATCH endpoint to update a TODO-list's title by its ID based on who the current authenticated user is. Returns 403 forbidden with a proper error JSON object if the user cannot access the list ○ A DELETE endpoint to remove a TODO-list. Soft-delete should be used

		● [Auth-ed] An endpoint to add someone by email to be able to access a TODO list: ○ This operation should be processed in an event-driven manner: The endpoint would immediate respond with an appropriate 200 JSON response after putting an event into a message broker (recommended rabbitmq as there's a free plan) ○ There will be a separate worker process that would consume the message and: ■ Do nothing if there's no existing user with such email ■ Give the corresponding user with such email access to the list ■ Requeue the message if there are errors during processing ● [Auth-ed] CUD endpoints for items in a TODO list, only for those with access to the specific list: ○ Create an item in the list ○ Update an item in the list ○ Delete an item from the list. Soft delete should be used ○ **Note:** There's no R endpoint as that's been covered in the TODO-list CRUD endpoint The app should be deployed to heroku. For the database, you can use the heroku postgres plugin free tier. For the message broker, you can use the free tier from rabbitmq. The code should be covered with unit test for at least 50% Bonus: ● Write integration tests with supertest for all endpoints ● Produce an OpenAPI yaml specs, and use it for request and response validation with express-openapi-validator ● Have a cronjob that update a global counter in the application on how many tasks have been completed for the entire user base every 5 minutes ● Have a public socket endpoint that would push updates on the above-mentioned counter whenever it's updated
Mode of Delivery	:	Individual Consultation

		+ Individual Presentation

MARKING RUBRICS FOR THE END OF MODULE PROJECT (CAPSTONE PROJECT)

	Score		
Category	**0-1**	**2-3**	**4-5**
Problem and Solution (x2 with 4 bonus points)	▪ **0-1** - No to very minimal attempt made ▪ **2-3** – The app is unusable.	▪ **2-4** Students implement only 1-2 of the requirements. ▪ **5-6 -** Students implement half of the requirements.	▪ **7-8** Students implement 80% of the requirements. ▪ **9-10** Students implement the complete set of requirements.

			▪ **11-14** Every bonus task completed grants students 1 bonus point
Documentation	▪ **0** - No code / attempt documentation written. ▪ **1** –The code is undocumented.	▪ **2 -** There are some comments explaining the code, however, provides no extra context to the code readers. ▪ **3 -** There are some comments explaining the code that the readers can't understand without.	▪ **4 -** The code is self-documented and easy to understand **or** there's external README file that's really helpful ▪ **5** – The code is written well, easy to understand. The app is documented in a README that someone can get up and running with
Presentation and Communication	▪ **0** - Student do not present clearly on the work and/or unable to answer questions related to the work. ▪ **1** – Student could briefly	▪ **2 -** Student present the work and are able to answer some questions related to the work. ▪ **3 -** Student present the	▪ **4 -** Student communicates well during presentation and able to answer most of the questions related to the work. ▪ **5** – Student

	present the work and/or unable to answer questions related to the work.	work and are able to answer most questions related to the work.	communicates well during presentation and able to answer all the questions related to the work.