

# **Assignment – I (12%)**

## COMP 3133 – Full Stack Development – II

**Submission: Sunday, 22<sup>nd</sup> Feb 2026, 11:59 PM**

---

No extension or late submission will be allowed. Late submission will be awarded ZERO points.

As a newly hired Jr. Software Engineer my manager assigned me a task to develop (Employee Management System) backend application using NodeJS, Express, GraphQL and MongoDB. Manager also wants me to apply my VCS (**GitHub**) skills along with DevOps skills to develop these projects where I will commit and push all my code to **COMP3133\_StudentID\_Assignment1** repository.

- **Instructions:**

You are required to implement the following GraphQL API. Ensure that each option performs the correct operation and returns the specified response object and error handling.

- **Objectives:**

1. Understand GraphQL API design principles.
2. Implement mutation and Query operations.
3. Test the response to ensure they return the correct response objects and error in case of failure.

- **GraphQL Implementation**

Following is the list of **GraphQL APIs** to develop which accept all data as JSON Object whenever needed:

Sr. #	Method	Operations	Description	Points
1	<b>Mutation</b>	Signup	Allow user to create new account	<b>05</b>
2	<b>Query</b>	Login	Allow user to access the system	<b>05</b>
3	<b>Query</b>	Get all employees	User can get all employee list	<b>10</b>
4	<b>Mutation</b>	Add New employee	User can create new employee <b>Store employee profile picture on <a href="https://cloudinary.com/">https://cloudinary.com/</a></b>	<b>10</b>
5	<b>Query</b>	Search employee by eid	User can get employee details by employee id	<b>10</b>
6	<b>Mutation</b>	Update employee by eid	User can update employee details	<b>10</b>
7	<b>Mutation</b>	Delete employee by eid	User can delete employee by employee id	<b>05</b>
8	<b>Query</b>	Search Employee by designation or department	User can get employee list by designation or department	<b>05</b>

- **Database Implementation**

MongoDB Database name: **comp3133\_StudentID\_Assignment1**

#### **Users Collection**

Field Name	Type	Constraint
<b>_id</b>	<b>Object ID</b>	<b>Auto Generate and not needs to add the fields in model</b>
username	String	Primary Key
email	String	Unique
password	String	Encrypted values must be stored
created_at	Date	Object creation data
updated_at	Date	Object updation date

User can login using username/email and password

#### **Employee Collection**

Field Name	Type	Constraint
<b>_id</b>	<b>Object ID</b>	<b>Auto Generate and not needs to add the fields in models</b>
first_name	String	Required
last_name	String	Required
email	String	Unique
gender	String	Male/Female/Other
designation	String	Required
salary	Float	Required must be $\geq 1000$
date_of_joining	Date	Required
department	String	Required
employee_photo	String	Store Image name/path. <b>Store employee profile picture on <a href="https://cloudinary.com/">https://cloudinary.com/</a></b>
created_at	Date	Object creation data
updated_at	Date	Object updation date

- **Testing & Validation**

1. **Validation:**

- Use libraries like **express-validator** to validate incoming requests.

2. **Testing with Postman:**

- Test all API endpoints and save the Postman collection.
- Capture screenshots of each test.

- **Implementation Guideline:**

- Implement GraphQL API using Apollo server or express-graphql
- Test GraphQL API using GraphiQL OR Postman
- Validate the input data whenever required
- Return error details or success response details whenever required
- All data must be sent back and forth in JSON Object format
- Optionally apply JWT security concept to secure all your API calls

- **Submission Checklist**

***Make single docx file with all labeled screenshots.***

1. **MongoDB Console Screenshots:**

- Provide screenshots of your MongoDB database showing collections.

2. **Postman API Collection:**

- Export your Postman collection and include it in your submission.

3. **Screenshots:**

- Include screenshots of each API being tested along with responses.

4. **Project ZIP File:**

- Remove the **node\_modules** folder and create a ZIP file of your project.

5. **GitHub Project Link:**

- Provide the link to your GitHub repository.

6. **Sample User Detail:**

- Include a sample user's details for testing login.

7. **Comments:**

- Add any comments or notes that could be helpful.

8. **Hosting:**

- If hosted, provide the URL to your deployed application on platforms like [Vercel](#), [Heroku](#), [Render](#), Docker, etc.

- **Evaluations:**

Points	Evaluation Component
<b>10</b>	GraphQL Object creation for both collection with MongoDB schemas
<b>60</b>	GraphQL APIs
<b>10</b>	Screenshots, Sending validation and error messages to client
<b>10</b>	Maintaining GitHub repository with ReadMe file
<b>10</b>	Test GraphQL API using GraphiQL OR Postman with Screenshots

- **Communication:**

- Please contact on [pritesh.patel2@georgebrown.ca](mailto:pritesh.patel2@georgebrown.ca) or SLACK channel for any question or query.
- NO requesting extension before 3 days for submission deadline

- **References:**

- <https://graphql.org/learn/>
- <https://www.howtographql.com/>
- [https://www.apollographql.com/blog/backend/using-express-with-graphql-server-node-js/?\\_ga=2.81606039.1894545270.1674864877-990100504.1674864877](https://www.apollographql.com/blog/backend/using-express-with-graphql-server-node-js/?_ga=2.81606039.1894545270.1674864877-990100504.1674864877)

*~~~ Wish you all the Best ~~~*