

Name: Vraj Patel

ID: 101327087

# ASSIGNMENT 1

## Add Employee

The screenshot shows the GraphQL Playground interface. On the left, the 'Documentation' pane shows the 'addEmployee' message with its arguments: first\_name (String!), last\_name (String!), email (String!), gender (String!), and salary (Float!). Below these are the fields returned: employee (Employee), error (String), message (String), status (Boolean), and user (User). The main editor shows the following GraphQL query:

```
1 mutation AddEmployee($firstName: String!, $lastName: String!, $email: String!, $gender: String!, $salary: Float!) {  
2   addEmployee(first_name: $firstName, last_name: $lastName, email: $email, gender: $gender, salary: $salary) {  
3     status  
4     message  
5     error  
6     employee {  
7       id  
8       first_name  
9       last_name  
10      email  
11      gender  
12      salary  
13    }  
14  }  
15 }
```

The 'Variables' pane shows the input variables:

```
1 {  
2   "firstName": "Vraj",  
3   "lastName": "Patel",  
4   "email": "vraj@gmail.com",  
5   "gender": "Male",  
6   "salary": 4000000.00  
7 }
```

The 'Response' pane shows the JSON output:

```
{  
  "data": {  
    "addEmployee": {  
      "status": true,  
      "message": "Employee Successfully created.",  
      "error": null,  
      "employee": {  
        "id": "63f29c667dfa6a85d7eabcc7",  
        "first_name": "Vraj",  
        "last_name": "Patel",  
        "email": "vraj@gmail.com",  
        "gender": "male",  
        "salary": 4000000  
      }  
    }  
  }  
}
```

The status bar at the top right indicates 'STATUS 200 | 90.0ms | 240B'.

The screenshot shows a document viewer interface. At the top right is an 'INSERT DOCUMENT' button. Below it is a search bar with the text 'FILTER { field: 'value' }' and an 'OPTIONS' button. To the right of the search bar are 'Apply' and 'Reset' buttons. The main area displays two JSON documents. The first document is:

```
{  
  "_id": ObjectId('63f29c667dfa6a85d7eabcc7'),  
  "first_name": "Vraj",  
  "last_name": "Patel",  
  "email": "vraj@gmail.com",  
  "gender": "male",  
  "salary": 4000000,  
  "created": 2023-02-19T22:02:14.190+00:00,  
  "updatedat": 2023-02-19T22:02:14.191+00:00,  
  "__v": 0  
}
```

The second document is:

```
{  
  "_id": ObjectId('63f29dc37dfa6a85d7eabcc9'),  
  "first_name": "Mukta",  
  "last_name": "Patel",  
  "email": "mukta@gmail.com",  
  "gender": "female",  
  "salary": 500000,  
  "created": 2023-02-19T22:08:03.176+00:00  
}
```

## Validation for gender

The screenshot shows a GraphQL Playground interface. On the left, the 'Operation' tab is active, displaying a mutation query: `mutation AddEmployee($firstName: String!, $lastName: String!, $email: String!, $gender: String!, $salary: Float!) { addEmployee(first_name: $firstName, last_name: $lastName, email: $email, gender: $gender, salary: $salary) { status message error employee { id first_name last_name email gender salary } } }`. Below the query, the 'Variables' tab shows the input: `{ "firstName": "Mukta", "lastName": "Patel", "email": "mukta@gmail.com", "gender": "Femle", "salary": 500000.00 }`. On the right, the 'Response' tab shows the JSON output: `{ "errors": [ { "message": "employees validation failed: gender: `femle` is not a valid enum value for path `gender`.", "locations": [ { "line": 5, "column": 5 } ], "path": [ "addEmployee", "error" ], "extensions": { "code": "INTERNAL_SERVER_ERROR", "exception": { "errors": { "gender": { "name": "ValidatorError", "message": "`femle` is not a valid enum value for path `gender`.", "properties": { "message": "`femle` is not a valid enum value for path `gender`.", "type": "enum", "enumValues": [`

## Validation for Email

The screenshot shows a GraphQL Playground interface. On the left, the 'Operation' tab is active, displaying the same mutation query as the previous screenshot. The 'Variables' tab shows the input: `{ "firstName": "Mukta", "lastName": "Patel", "email": "muktagmail.com", "gender": "Female", "salary": 500000.00 }`. On the right, the 'Response' tab shows the JSON output: `{ "errors": [ { "message": "employees validation failed: email: Validator failed for path `email` with value `muktagmail.com`, "locations": [ { "line": 5, "column": 5 } ], "path": [ "addEmployee", "error" ], "extensions": { "code": "INTERNAL_SERVER_ERROR", "exception": { "errors": { "email": { "name": "ValidatorError", "message": "Validator failed for path `email` with value `muktagmail.com`, "properties": { "message": "Validator failed for path `email` with value `muktagmail.com`",`

## Validation for existing employee

The screenshot shows the GraphQL Studio interface with the **AddEmployee** tab selected. The **Operation** pane contains the following query:

```
1 mutation AddEmployee($firstName: String!, $lastName: String!, $email:...  
2 String!, $gender: String!, $salary: Float!) {  
3   addEmployee(first_name: $firstName, last_name: $lastName, email:  
4     $email, gender: $gender, salary: $salary) {  
5     status  
6     message  
7     error  
8     employee {  
9       id  
10      first_name  
11      last_name  
12      email  
13      gender  
14      salary  
15    }  
  }
```

The **Response** pane shows the result of the mutation, which failed with a 200 status:

```
{  
  "data": {  
    "addEmployee": {  
      "status": false,  
      "message": "Employee exists with that  
email.",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

The **Variables** pane shows the input variables:

```
1 {  
2   "firstName": "Vraj",  
3   "lastName": "Patel",  
4   "email": "vraj@gmail.com",  
5   "gender": "Male",  
6   "salary": 4000000.00  
7 }
```

## Delete Employee

The screenshot shows the GraphQL Studio interface with the **DeleteEmployee** tab selected. The **Operation** pane contains the following query:

```
1 mutation DeleteEmployee($deleteEmployeeId: String!) {  
2   deleteEmployee(id: $deleteEmployeeId) {  
3     status  
4     message  
5     error  
6     employee {  
7       id  
8       first_name  
9       last_name  
10      email  
11      gender  
12      salary  
13    }  
14  }  
15 }
```

The **Response** pane shows the result of the mutation, which was successful with a 200 status:

```
{  
  "data": {  
    "deleteEmployee": {  
      "status": true,  
      "message": "63f29c667dfa6a85d7eabcc7  
deleted successfully.",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

The **Variables** pane shows the input variables:

```
1 {  
2   "deleteEmployeeId": "63f29c667dfa6a85d7eabcc7"  
3 }
```

## Delete Employee without ID error

The screenshot shows a GraphQL IDE interface with two tabs: 'AddEmployee' and 'DeleteEmployee'. The 'DeleteEmployee' tab is active. The 'Operation' pane on the left contains the following GraphQL query:

```
1 mutation DeleteEmployee($deleteEmployeeId: String!) {  
2   deleteEmployee(id: $deleteEmployeeId) {  
3     status  
4     message  
5     error  
6     employee {  
7       id  
8       first_name  
9       last_name  
10      email  
11      gender  
12      salary  
13    }  
14  }  
15 }
```

The 'Response' pane on the right shows the JSON response:

```
{  
  "data": {  
    "deleteEmployee": {  
      "status": false,  
      "message": "Check! Check! You didn't  
enter id to delete your employee",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

Below the operation pane, the 'Variables' section shows a single variable:

```
1 {  
2   "deleteEmployeeId": ""  
3 }
```

The status bar at the top right indicates 'STATUS 200 | 37.0ms | 144B'.

## Delete Employee with the wrong ID

The screenshot shows the same GraphQL IDE interface as the first image, but with a different variable value. The 'Operation' pane contains the same GraphQL query:

```
1 mutation DeleteEmployee($deleteEmployeeId: String!) {  
2   deleteEmployee(id: $deleteEmployeeId) {  
3     status  
4     message  
5     error  
6     employee {  
7       id  
8       first_name  
9       last_name  
10      email  
11      gender  
12      salary  
13    }  
14  }  
15 }
```

The 'Response' pane shows the JSON response:

```
{  
  "data": {  
    "deleteEmployee": {  
      "status": false,  
      "message": "There is no employee with  
the id you entered",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

The 'Variables' section shows the variable with a specific ID:

```
1 {  
2   "deleteEmployeeId": "63f29c667dfa6a85d7eabcc5"  
3 }
```

The status bar at the top right indicates 'STATUS 200 | 74.0ms | 132B'.

## Update Employee

**Operation**

```
1 mutation UpdateEmployee($updateEmployeeId: ID!, $firstName: String!, ...  
2 $lastName: String!, $email: String!, $gender: String!, $salary: Float!) {  
3   updateEmployee(id: $updateEmployeeId, first_name: $firstName,  
4     last_name: $lastName, email: $email, gender: $gender, salary: $salary) {  
5     message  
6     error  
7     employee {  
8       email  
9       first_name  
10      gender  
11      id  
12      last_name  
13      salary  
14    }  
  }  
}
```

**Response**

```
{  
  "data": {  
    "updateEmployee": {  
      "message": "63f29dc37dfa6a85d7eabcc9  
updated successfully.",  
      "error": null,  
      "employee": {  
        "email": "mukta01@gmail.com",  
        "first_name": "Mukta",  
        "gender": "female",  
        "id": "63f29dc37dfa6a85d7eabcc9",  
        "last_name": "Patel",  
        "salary": 90000  
      }  
    }  
  }  
}
```

**Variables**

```
1 {  
2   "updateEmployeeId": "63f29dc37dfa6a85d7eabcc9",  
3   "firstName": "Mukta",  
4   "lastName": "Patel",  
5   "email": "mukta01@gmail.com",  
6   "gender": "Female",  
7   "salary": 90000.00  
8 }
```

## Update Employee without id

**Operation**

```
1 mutation UpdateEmployee($updateEmployeeId: ID!, $firstName: String!, ...  
2 $lastName: String!, $email: String!, $gender: String!, $salary: Float!) {  
3   updateEmployee(id: $updateEmployeeId, first_name: $firstName,  
4     last_name: $lastName, email: $email, gender: $gender, salary: $salary) {  
5     message  
6     error  
7     employee {  
8       email  
9       first_name  
10      gender  
11      id  
12      last_name  
13      salary  
14    }  
  }  
}
```

**Response**

```
{  
  "data": {  
    "updateEmployee": {  
      "message": "Check! You didn't enter  
an id to update you employee.",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

**Variables**

```
1 {  
2   "updateEmployeeId": "",  
3   "firstName": "Mukta",  
4   "lastName": "Patel",  
5   "email": "mukta01@gmail.com",  
6   "gender": "Female",  
7   "salary": 90000.00  
8 }
```

## Update Employee with the wrong id

The screenshot shows the GraphQL Studio interface with the 'UpdateEmployee' tab selected. The query is a mutation that updates an employee's details. The response shows a 200 status with a message indicating that no employee was found for the provided ID.

**Operation**

```
1 mutation UpdateEmployee($updateEmployeeId: ID!, $firstName: String!, ...  
2 $lastName: String!, $email: String!, $gender: String!, $salary: Float!) {  
3   updateEmployee(id: $updateEmployeeId, first_name: $firstName,  
4   last_name: $lastName, email: $email, gender: $gender, salary: $salary) {  
5     message  
6     error  
7     employee {  
8       email  
9       first_name  
10      gender  
11      id  
12      last_name  
13      salary  
14    }  
15  }  
16 }
```

**Response** STATUS 200 | 667ms | 124B

```
{  
  "data": {  
    "updateEmployee": {  
      "message": "Sorry! There is no  
employee with the id you entered.",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

**Variables** Headers Script NEW! JSON

```
1 {  
2   "updateEmployeeId": "63f29dc37dfa6a85d7eabcc4",  
3   "firstName": "Mukta",  
4   "lastName": "Patel",  
5   "email": "mukta01@gmail.com",  
6   "gender": "Female",  
7   "salary": 90000.00  
8 }
```

## Signup

The screenshot shows the GraphQL Studio interface with the 'Signup' tab selected. The query is a mutation that creates a new user account. The response shows a 200 status with a success message and the newly created employee's details.

**Operation**

```
1 mutation Signup($username: String!, $email: String!, $password: String!) {  
2   signup(username: $username, email: $email, password: $password) {  
3     status  
4     message  
5     error  
6     employee {  
7       id  
8       first_name  
9       last_name  
10      email  
11      gender  
12      salary  
13    }  
14  }  
15 }
```

**Response** STATUS 200 | 58.0ms | 130B

```
{  
  "data": {  
    "signup": {  
      "status": true,  
      "message": "Thank you! You have  
successfully created an account.",  
      "error": null,  
      "employee": null  
    }  
  }  
}
```

**Variables** Headers Script NEW! JSON

```
1 {  
2   "username": "vrajpatel",  
3   "email": "vraj@gmail.com",  
4   "password": "vraj00"  
5 }
```

## Signup with existing information

The screenshot shows the GraphQL Playground interface with the 'Signup' tab selected. The 'Operation' tab on the left contains a GraphQL mutation: 

```
1 mutation Signup($username: String!, $email: String!, $password: String!) {
2   signup(username: $username, email: $email, password: $password) {
3     status
4     message
5     error
6     employee {
7       id
8       first_name
9       last_name
10      email
11      gender
12      salary
13    }
14  }
15 }
```

 The 'Variables' tab below it shows: 

```
1 {
2   "username": "vrajpatel",
3   "email": "vraj@gmail.com",
4   "password": "vraj00"
5 }
```

 The 'Response' tab on the right shows the result: 

```
{
  "data": {
    "signup": {
      "status": false,
      "message": "Oops! An account already exists with the same email.",
      "error": null,
      "employee": null
    }
  }
}
```

 The status bar indicates a 200 status code, 713ms execution time, and 131B response size.

## Login

The screenshot shows the GraphQL Playground interface with the 'Login' tab selected. The 'Operation' tab on the left contains a GraphQL query: 

```
1 query Login($username: String!, $password: String!) {
2   login(username: $username, password: $password) {
3     message
4   }
5 }
```

 The 'Variables' tab below it shows: 

```
1 {
2   "username": "vrajpatel",
3   "password": "vraj00"
4 }
```

 The 'Response' tab on the right shows the result: 

```
{
  "data": {
    "login": {
      "message": "Welcome, vrajpatel! You have successfully logged in."
    }
  }
}
```

 The status bar indicates a 200 status code, 122ms execution time, and 86B response size.

## Login with an invalid password

The screenshot shows a GraphQL IDE interface with two tabs: "AddEmployee" and "Login". The "Login" tab is active. The "Operation" section contains a query:

```
1 query Login($username: String!, $password: String!) {  
2   login(username: $username, password: $password) {  
3     message  
4   }  
5 }
```

The "Response" section shows the result of the query:

```
{  
  "data": {  
    "login": {  
      "message": "Sorry! Please check the  
username and password you entered."  
    }  
  }  
}
```

The status bar indicates "STATUS 200 | 88.0ms | 92B". Below the operation section, the "Variables" section is visible, showing the input variables:

```
1 {  
2   "username": "vrajpatel",  
3   "password": "vraj5"  
4 }
```

## All Employees

The screenshot shows a GraphQL IDE interface with two tabs: "AddEmployee" and "GetEmployee...". The "GetEmployee..." tab is active. The "Operation" section contains a query:

```
1 query GetEmployeeByID {  
2   getEmployees {  
3     email  
4     first_name  
5     gender  
6     id  
7     last_name  
8     salary  
9   }  
10 }
```

The "Response" section shows the result of the query:

```
{  
  "data": {  
    "getEmployees": [  
      {  
        "email": "mukta01@gmail.com",  
        "first_name": "Mukta",  
        "gender": "female",  
        "id": "63f29dc37dfa6a85d7eabcc9",  
        "last_name": "Patel",  
        "salary": 90000  
      },  
      {  
        "email": "vraj@gmail.com",  
        "first_name": "Vraj",  
        "gender": "male",  
        "id": "63f2a1cf74110348cd2e689f",  
        "last_name": "Patel",  
        "salary": 400000  
      }  
    ]  
  }  
}
```

The status bar indicates "STATUS 200 | 66.0ms | 296B". Below the operation section, the "Variables" section is visible, showing the input variables:

```
1 {  
2 }
```



## Get Employees by ID

The screenshot shows a REST client interface with two tabs: "AddEmployee" and "GetEmployee...". The "GetEmployee..." tab is active, displaying a GET request to the endpoint `GetEmployeeByID($getEmployeeByIdId: ID!)`. The request body is a JSON object with the following fields: `email`, `first_name`, `gender`, `id`, `last_name`, and `salary`. The response is a JSON object with the following fields: `data`, `getEmployeeByID`, `email`, `first_name`, `gender`, `id`, `last_name`, and `salary`. The status is 200, and the response size is 60.0ms and 161B.

```
query GetEmployeeByID($getEmployeeByIdId: ID!) {  
  getEmployeeByID(id: $getEmployeeByIdId) {  
    email  
    first_name  
    gender  
    id  
    last_name  
    salary  
  }  
}
```

```
{  
  "data": {  
    "getEmployeeByID": {  
      "email": "vraj@gmail.com",  
      "first_name": "Vraj",  
      "gender": "male",  
      "id": "63f2a1cf74110348cd2e689f",  
      "last_name": "Patel",  
      "salary": 4000000  
    }  
  }  
}
```

Variables: Headers Script NEW! JSON

```
1 {  
2   "getEmployeeByIdId": "63f2a1cf74110348cd2e689f"  
3 }
```

## MongoDB Employees

The screenshot shows the MongoDB Compass interface. The left sidebar displays a list of databases and collections, including "Labtest2", "comp3123\_assignment1", "comp3133\_ass1", "employees", "h", "users", "comp3133\_lab3", "here", "lab04", "labtest1", and "well". The main panel shows a query filter: `{ field: 'value' }`. The query results are displayed as a list of documents, showing the first two results. The first document has the following fields: `_id`, `first_name`, `last_name`, `email`, `gender`, `salary`, `created`, `updatedat`, and `__v`. The second document has the following fields: `_id`, `first_name`, `last_name`, and `email`.

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 334B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

QUERY RESULTS: 1-2 OF 2

```
{  
  "_id": ObjectId('63f29dc37dfa6a85d7eabcc9'),  
  "first_name": "Mukta",  
  "last_name": "Patel",  
  "email": "mukta91@gmail.com",  
  "gender": "female",  
  "salary": 90000,  
  "created": 2023-02-19T22:08:03.176+00:00,  
  "updatedat": 2023-02-19T22:08:03.176+00:00,  
  "__v": 0  
}
```

```
{  
  "_id": ObjectId('63f2a1cf74110348cd2e689f'),  
  "first_name": "Vraj",  
  "last_name": "Patel",  
  "email": "vraj@gmail.com"  
}
```

System Status: All Good

# MongoDB Users

Q Search Namespaces

Labtest2

comp3123\_assignment1

comp3133\_ass1

employees

h

users

comp3133\_lab3

here

lab04

labtest1

well

STORAGE SIZE: 34KB LOGICAL DATA SIZE: 281B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 108KB

Find

Indexes

Schema Anti-Patterns ⓘ

Aggregation

Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' }

▶ OPTIONS

Apply

Reset

QUERY RESULTS: 1-2 OF 2

\_id: ObjectId('63f2a374c8c4914bfcad0678')

username: "vrajpatel"

email: "vraj@gmail.com"

password: "vraj00"

created: 2023-02-19T22:32:20.182+00:00

updatedat: 2023-02-19T22:32:20.182+00:00

\_\_v: 0

\_id: ObjectId('63f2a90d21b90f32c822ea00')

username: "muktapatel"

email: "mukt0@gmail.com"

password: "mukta00"

created: 2023-02-19T22:56:13.226+00:00

updatedat: 2023-02-19T22:56:13.226+00:00

System Status: All Good