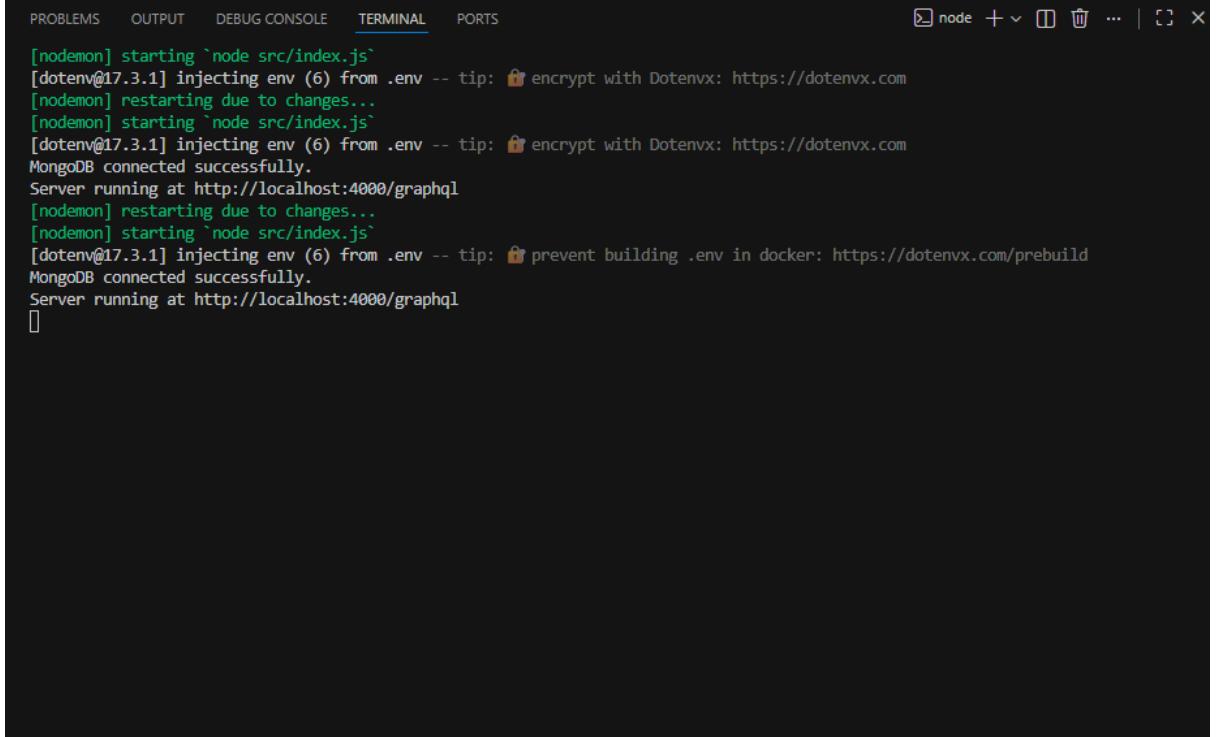


# Assignment01\_COMP3133\_Screenshots

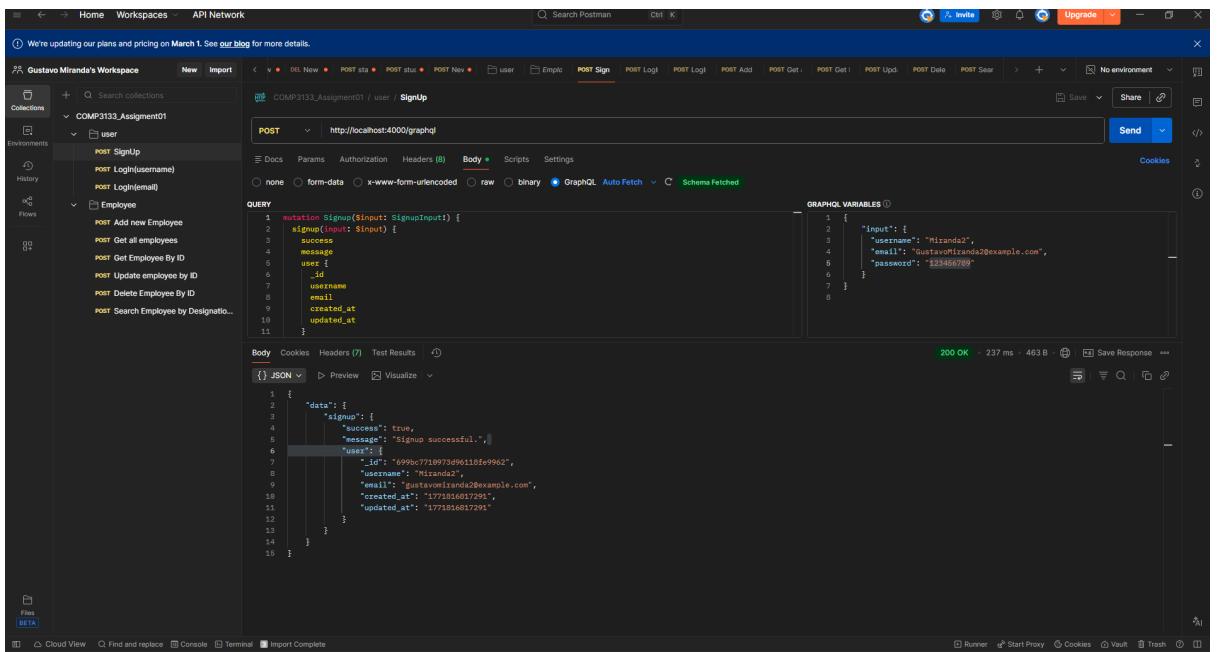
## 1. Server Running



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[nodemon] starting `node src/index.js`
[nodenv@17.3.1] injecting env (6) from .env -- tip: 🛡 encrypt with Dotenvx: https://dotenvx.com
[nodemon] restarting due to changes...
[nodemon] starting `node src/index.js`
[nodenv@17.3.1] injecting env (6) from .env -- tip: 🛡 encrypt with Dotenvx: https://dotenvx.com
MongoDB connected successfully.
Server running at http://localhost:4000/graphql
[nodemon] restarting due to changes...
[nodemon] starting `node src/index.js`
[nodenv@17.3.1] injecting env (6) from .env -- tip: 🛡 prevent building .env in docker: https://dotenvx.com/prebuild
MongoDB connected successfully.
Server running at http://localhost:4000/graphql

```

## 2. Signup - Success



The screenshot shows a Postman collection named "COMP3133\_Assignment01" with a "user" folder containing a "SignUp" request. The request is a POST to `http://localhost:4000/graphql` with the following body:

```
mutation Signup($input: SignupInput!) {
  signup(input: $input) {
    success
    message
    user {
      _id
      username
      email
      created_at
      updated_at
    }
  }
}
```

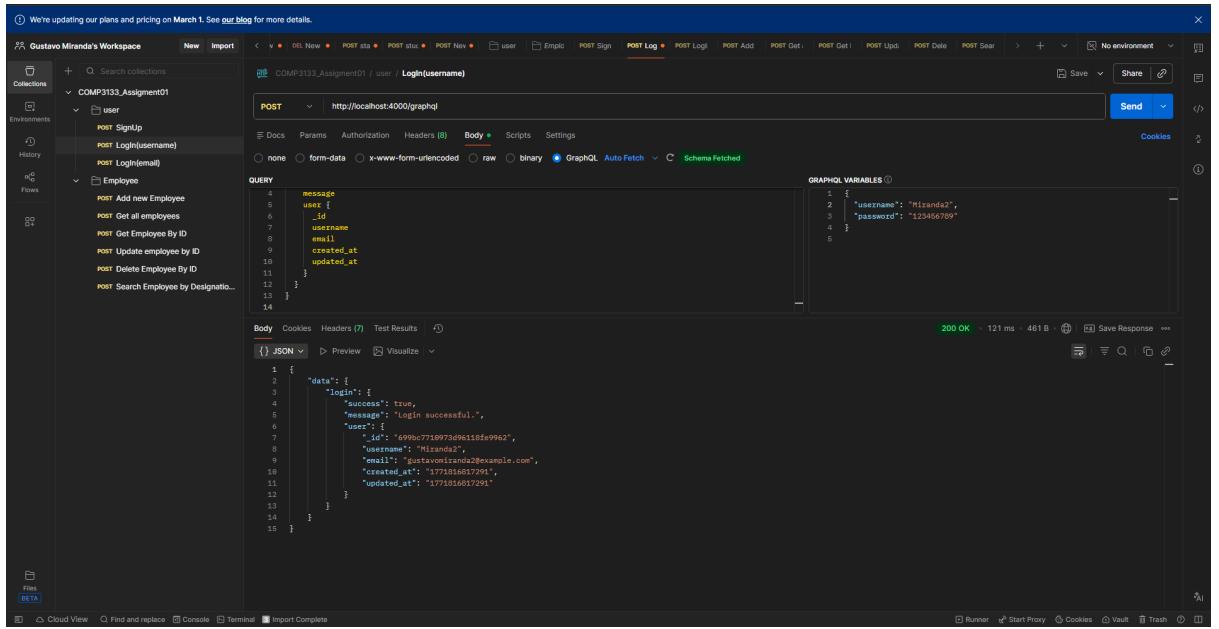
The "GRAPHQL VARIABLES" section shows the input variables:

```
[{"input": {"username": "Miranda", "email": "GustavoMiranda2@example.com", "password": "2294567890"}]}
```

The "Test Results" section shows a successful response with status code 200 OK, duration 237 ms, and a response body:

```
{"data": { "signup": { "success": true, "message": "Signup successful.", "user": { "id": "499bc710973961118fe9962", "username": "Miranda", "email": "gustavomiranda2@example.com", "created_at": "2024-01-17T18:16:01.729Z", "updated_at": "2024-01-17T18:16:01.729Z" } } }}
```

### 3. Login by Username - Success



The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A POST request is made to the endpoint `/user / Login(username)`. The query is:

```
query {   message   user {     _id     username     email     created_at     updated_at   } }
```

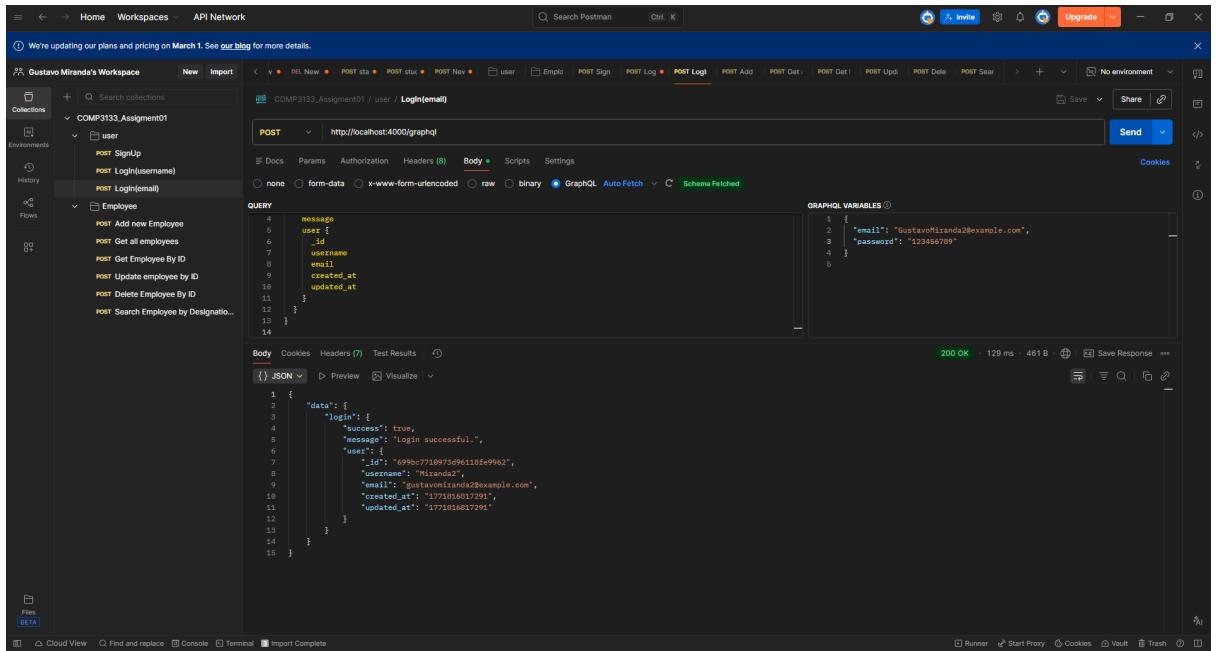
The GRAPHQL VARIABLES section contains:

```
1 {   "username": "Miranda2",   "password": "123456789" }
```

The response status is 200 OK, and the response body is:

```
1 {   "data": {     "login": {       "success": true,       "message": "Login successful.",       "user": {         "_id": "499bc7710973d96118fe9962",         "username": "Miranda2",         "email": "gustavomiranda2@example.com",         "created_at": "1771816817291",         "updated_at": "1771816817291"       }     }   } }
```

### 4. Login by Email - Success



The screenshot shows the Postman interface with the same collection. A POST request is made to the endpoint `/user / Login(email)`. The query is:

```
query {   message   user {     _id     username     email     created_at     updated_at   } }
```

The GRAPHQL VARIABLES section contains:

```
1 {   "email": "GustavoMiranda2@example.com",   "password": "123456789" }
```

The response status is 200 OK, and the response body is identical to the previous screenshot.

## 5. Add New Employee - Success

The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A POST request is made to the URL `http://localhost:4000/graphql`. The query is:

```
query {
  addNewEmployee(
    $input: EmployeeInput!
  ) {
    success
    message
    employee {
      id
      first_name
      last_name
      email
      gender
      designation
      salary
      date_of_joining
      department
      employee_photo
      created_at
      updated_at
    }
  }
}
```

The variables section contains:

```
$input: {  
  first_name: "Sanatha",  
  last_name: "Silva",  
  email: "employee teste_0@example.com",  
  gender: "Female",  
  designation: "Software Engineer",  
  salary: 8888,  
  date_of_joining: "2026-02-28",  
  department: "IT",  
  employee_photo: "https://res.cloudinary.com/demo/image/upload/getting-started/shoes.jpg"  
}
```

The response status is 200 OK, and the response body is:

```
{  
  "data": {  
    "addNewEmployee": {  
      "success": true,  
      "message": "Employee added successfully.",  
      "employee": {  
        "id": "699bc0630973d96110fe996a",  
        "first_name": "Sanatha",  
        "last_name": "Silva",  
        "email": "employee teste_0@example.com",  
        "gender": "Female",  
        "designation": "Software Engineers",  
        "salary": 8888,  
        "date_of_joining": "17715456800000",  
        "department": "IT",  
        "employee_photo": "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817866/comp3133_assignment1_employees/pxu2m6abk74psckdngvo.jpg",  
        "created_at": "1771817866",  
        "updated_at": "1771817869294"  
      }  
    }  
  }  
}
```

## 6. Get All Employees - Success

The screenshot shows the Postman interface with the same collection. A POST request is made to the URL `http://localhost:4000/graphql`. The query is:

```
query {
  getAllEmployees {
    id
    first_name
    last_name
    email
    gender
    designation
    salary
    date_of_joining
    department
    employee_photo
    created_at
    updated_at
  }
}
```

The response status is 200 OK, and the response body is:

```
{  
  "data": {  
    "getAllEmployees": [  
      {  
        "id": "699bc0630973d96110fe996a",  
        "first_name": "Sanatha",  
        "last_name": "Silva",  
        "email": "employee teste_0@example.com",  
        "gender": "Female",  
        "designation": "Software Engineer",  
        "salary": 8888,  
        "date_of_joining": "17715456800000",  
        "department": "IT",  
        "employee_photo": "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817866/comp3133_assignment1_employees/pxu2m6abk74psckdngvo.jpg",  
        "created_at": "1771817866",  
        "updated_at": "1771817869294"  
      }  
    ]  
  }  
}
```

## 7. Search Employee by ID - Success

The screenshot shows the Postman application interface. The collection is 'COMP3133.Assignment01' and the endpoint is 'Get Employee By ID'. The query is:

```
query { employee(id: "699bc8638973d96110fe996a") { designation salary date_of_joining department employee_photo created_at updated_at } }
```

The response status is 200 OK with a response time of 83 ms and a body size of 772 B. The response JSON is:

```
{ "data": { "employee": { "id": "699bc8638973d96110fe996a", "success": true, "message": "Employee fetched successfully.", "employee": { "id": "699bc8638973d96110fe996a", "first_name": "Samantha", "last_name": "Silva", "email": "employee_teste_01@example.com", "gender": "Female", "designation": "Software Engineer", "salary": 6000, "date_of_joining": "177154566868000", "department": "IT", "employee_photo": "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817866/comp3133_assignment1_employees/pxu2m6abk74psckdnvo.jpg", "created_at": "1771817869284", "updated_at": "1771817869284" } } }}
```

## 8. Update Employee by ID - Success

The screenshot shows the Postman application interface. The collection is 'COMP3133.Assignment01' and the endpoint is 'Update employee by ID'. The query is:

```
mutation { updateEmployeeById(id: "699bc8638973d96110fe996a", input: { designation: "Software Engineer", salary: 6000, date_of_joining: "177154566868000", department: "Engineering", employee_photo: "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817866/comp3133_assignment1_employees/pxu2m6abk74psckdnvo.jpg", created_at: "1771817869284", updated_at: "1771817869284" }) { success, message, employee { id, first_name, last_name, email, gender, designation, salary, date_of_joining, department, employee_photo, created_at, updated_at } }
```

The response status is 200 OK with a response time of 155 ms and a body size of 771 B. The response JSON is:

```
{ "data": { "updateEmployeeById": { "success": true, "message": "Employee updated successfully.", "employee": { "id": "699bc8638973d96110fe996a", "first_name": "Samantha", "last_name": "Silva", "email": "employee_teste_01@example.com", "gender": "Female", "designation": "Software Engineer", "salary": 6000, "date_of_joining": "177154566868000", "department": "Engineering", "employee_photo": "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817866/comp3133_assignment1_employees/pxu2m6abk74psckdnvo.jpg", "created_at": "1771817869284", "updated_at": "1771817869284" } } }}
```

## 9. Search by Designation or Department - Success

The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A POST request is made to `http://localhost:4000/graphq` with the following query:

```
query {
  gender
  designation
  id
  name
  date_of_joining
  department
  employee_photo
  created_at
  updated_at
}
```

The response is a 200 OK status with a JSON payload containing one employee record:

```
{
  "data": [
    {
      "searchEmployeeByDesignationOrDepartment": [
        {
          "id": "699bc8638973d9611bfef99ea",
          "first_name": "Senthil",
          "last_name": "Silva",
          "email": "employee teste_B@example.com",
          "gender": "Female",
          "designation": "S3333",
          "date_of_joining": "177151456800000",
          "department": "engineering",
          "employee_photo": "https://res.cloudinary.com/dc8edthcc/image/upload/v1771817866/comp3133_assignment1_employees/exu2m6abk74psckngv0.jpg",
          "created_at": "1771817869984",
          "updated_at": "1771817196019"
        }
      ]
    }
  ]
}
```

## 10. Delete Employee by ID - Success

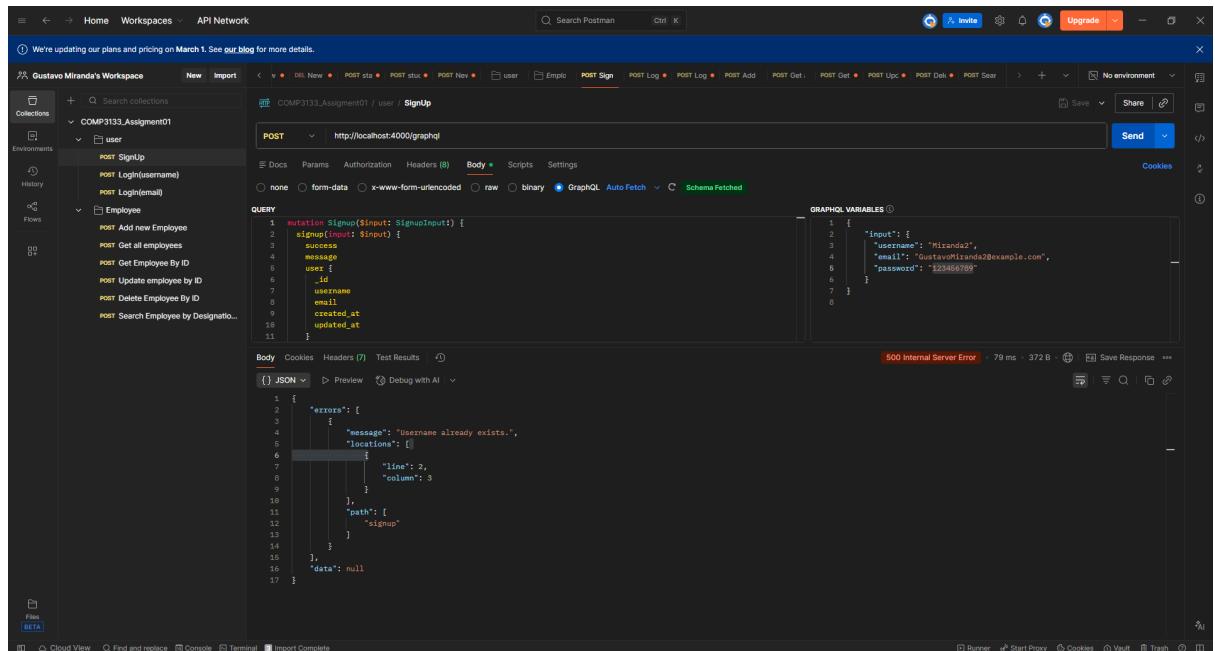
The screenshot shows the Postman interface with the same collection. A POST request is made to `http://localhost:4000/graphq` with the following mutation:

```
mutation DeleteEmployeeById($id: ID!) {
  deleteEmployeeById(id: $id) {
    success
    message
  }
}
```

The response is a 200 OK status with a JSON payload indicating the deletion was successful:

```
{
  "data": {
    "deleteEmployeeById": {
      "success": true,
      "message": "Employee deleted successfully."
    }
  }
}
```

## 11. Signup - Duplicate Username Error



The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A POST request is made to `http://localhost:4000/graphq` to the endpoint `/user | SignUp`. The body contains a GraphQL mutation:

```
mutation Signup($input: SignupInput!) {
  signup(input: $input) {
    success
    message
    user {
      _id
      username
      email
      created_at
      updated_at
    }
  }
}
```

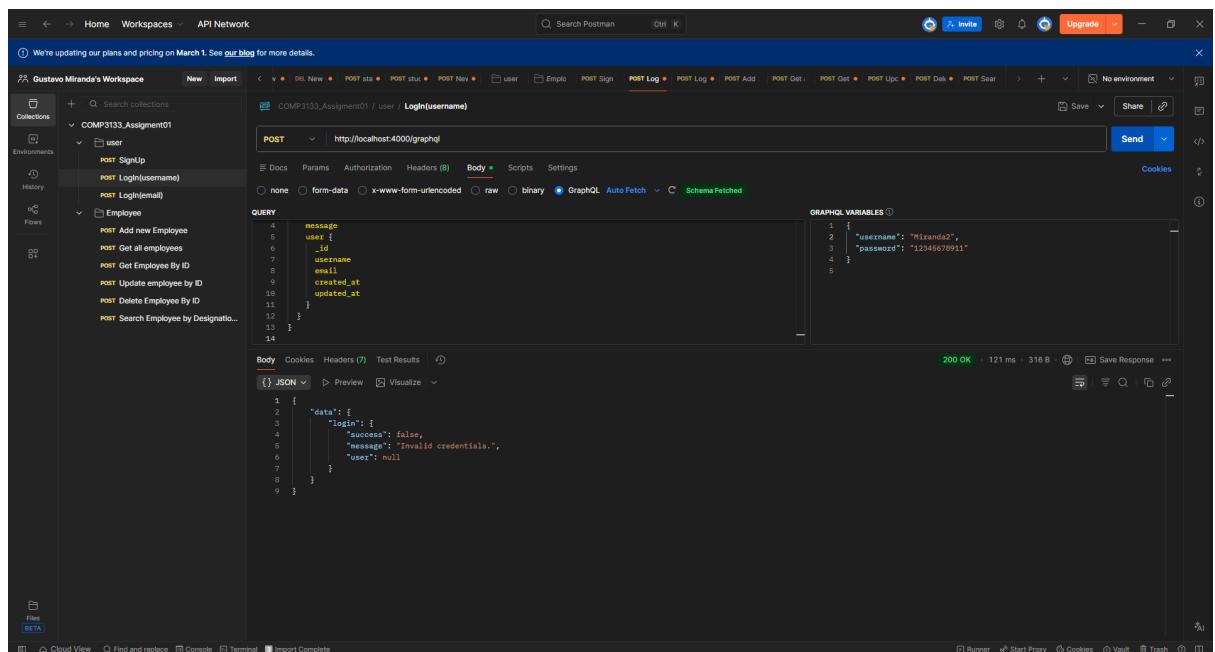
The `GRAPHQL VARIABLES` pane shows the input variables:

```
1 [ 2   "input": { 3     "username": "Miranda", 4     "email": "GustavoMiranda2@example.com", 5     "password": "1234567890" 6   } 7 ] 8 }
```

The response status is `500 Internal Server Error` with a duration of `79 ms` and a size of `372 B`. The response body is:

```
{ 1   "errors": [ 2     { 3       "message": "Username already exists.", 4       "locations": [ 5         { 6           "line": 2, 7           "column": 3 8         } 9       ], 10      "path": [ 11        "signup" 12      ] 13     }, 14     { 15       "data": null 16     } 17   ] 18 }
```

## 12. Login - Wrong Password Error



The screenshot shows the Postman interface with the same collection. A POST request is made to `http://localhost:4000/graphq` to the endpoint `/user | LoginByUsername`. The body contains a GraphQL query:

```
query {
  user {
    _id
    username
    email
    created_at
    updated_at
  }
}
```

The `GRAPHQL VARIABLES` pane shows the input variables:

```
1 [ 2   "username": "Miranda2", 3   "password": "12345678911" 4 ] 5 }
```

The response status is `200 OK` with a duration of `121 ms` and a size of `316 B`. The response body is:

```
{ 1   "data": { 2     "login": { 3       "success": false, 4       "message": "Invalid credentials.", 5       "user": null 6     } 7   } 8 }
```

### 13. Add Employee - Salary Validation Error

The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A mutation named "Add new Employee" is selected. The "Body" tab shows a GraphQL query:

```
query {
  designation
  salary
  date_of_joining
  department
  employee_photo
  created_at
  updated_at
}
```

The "Headers" tab includes "Content-Type: application/json". The "GraphQl" tab is selected. The "Test Results" tab shows a JSON response with an error:

```
{
  "errors": [
    {
      "message": "Salary must be a number and >= 1000.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "addNewEmployee"
      ]
    }
  ],
  "data": null
}
```

The status bar at the bottom indicates a "500 Internal Server Error" with a duration of 4 ms and a size of 392 B.

### 14. Add Employee - Gender Validation Error

The screenshot shows the Postman interface with the same collection and mutation. The "Body" tab shows the same GraphQL query as above. The "Headers" tab includes "Content-Type: application/json". The "GraphQl" tab is selected. The "Test Results" tab shows a JSON response with an error:

```
{
  "errors": [
    {
      "message": "Gender must be Male, Female or Other.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "addNewEmployee"
      ]
    }
  ],
  "data": null
}
```

The status bar at the bottom indicates a "500 Internal Server Error" with a duration of 4 ms and a size of 393 B.

## 15. Search Employee by ID - Invalid ID Error

The screenshot shows the Postman interface with a GraphQL query for searching an employee by ID. The query is:

```
query {
  designation
  salary
  date_of_joining
  department
  employee_photo
  created_at
  updated_at
}

```

The GraphQL variables section shows a single variable:

```
eid: "21"
```

The response body shows an error message:

```
{
  "errors": [
    {
      "message": "Invalid employee id.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "searchEmployeeById"
      ]
    }
  ],
  "data": null
}
```

The status bar at the bottom indicates a 500 Internal Server Error.

## 16. Update Employee by ID - Not Found Error

The screenshot shows the Postman interface with a GraphQL mutation for updating an employee by ID. The mutation is:

```
mutation {
  updateEmployeeById(eid: 699bc8638973d96110fe996, input: {designation: "S3333", salary: 666633, department: "engineering"})
}
```

The GraphQL variables section shows the input parameters:

```
eid: "699bc8638973d96110fe996",  
input: {designation: "S3333",  
salary: 666633,  
department: "engineering"}
```

The response body shows an error message:

```
{
  "errors": [
    {
      "message": "Invalid employee id.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "updateEmployeeById"
      ]
    }
  ],
  "data": null
}
```

The status bar at the bottom indicates a 500 Internal Server Error.

## 17. Delete Employee by ID - Not Found Error

The screenshot shows the Postman interface with a collection named "COMP3133\_Assignment01". A POST request is made to `http://localhost:4000/graphql` with the following query:

```
mutation DeleteById($id: ID!) {
  deleteEmployeeById(id: $id) {
    success
    message
  }
}
```

The response status is 500 Internal Server Error, and the JSON body is:

```
{
  "errors": [
    {
      "message": "Invalid employee id.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "deleteEmployeeById"
      ]
    }
  ],
  "data": null
}
```

GraphQL variables shown are:

```
1 | {
2 |   "id": "699bc8638973d96118fe99"
3 |
4 |
5 |
6 |
7 }
```

## 18. Search by Designation or Department - Missing Filters Error

The screenshot shows the Postman interface with the same collection. A POST request is made to `http://localhost:4000/graphql` with the following query:

```
query {
  gender
  designation
  salary
  date_of_joining
  designation
  employee_photo
  created_at
  updated_at
}
```

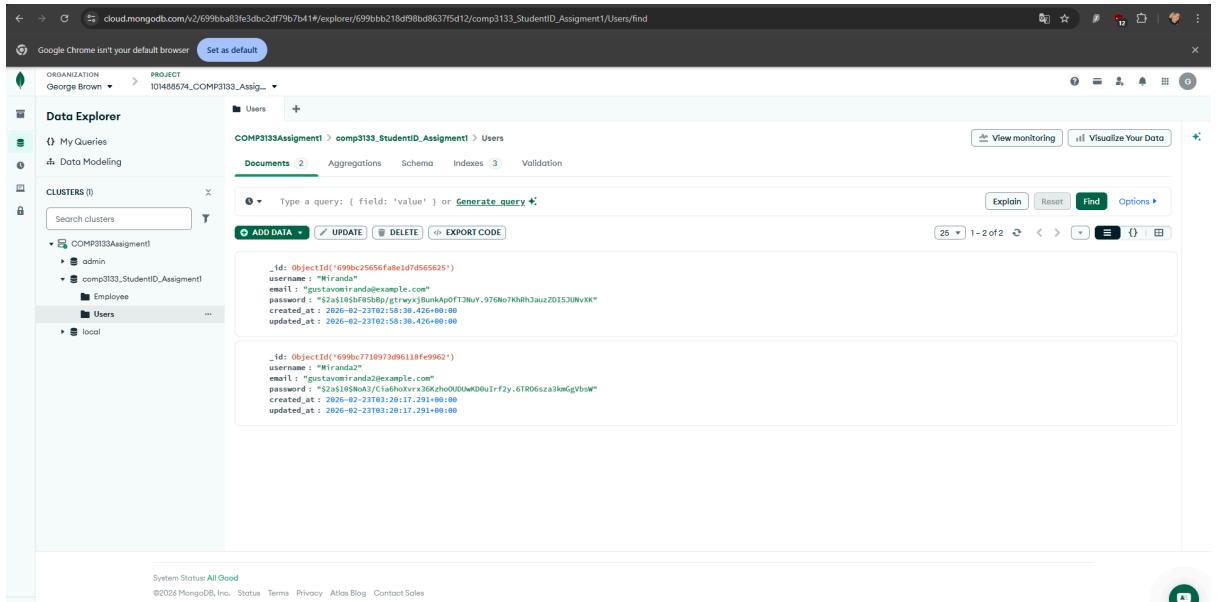
The response status is 500 Internal Server Error, and the JSON body is:

```
{
  "errors": [
    {
      "message": "Provide designation or department.",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "searchEmployeeByDesignationOrDepartment"
      ]
    }
  ],
  "data": null
}
```

GraphQL variables shown are:

```
1 | {
2 |   "designation": "",
3 |   "department": ""
4 |
5 |
6 |
7 }
```

## 19. MongoDB Atlas - Database and Collections

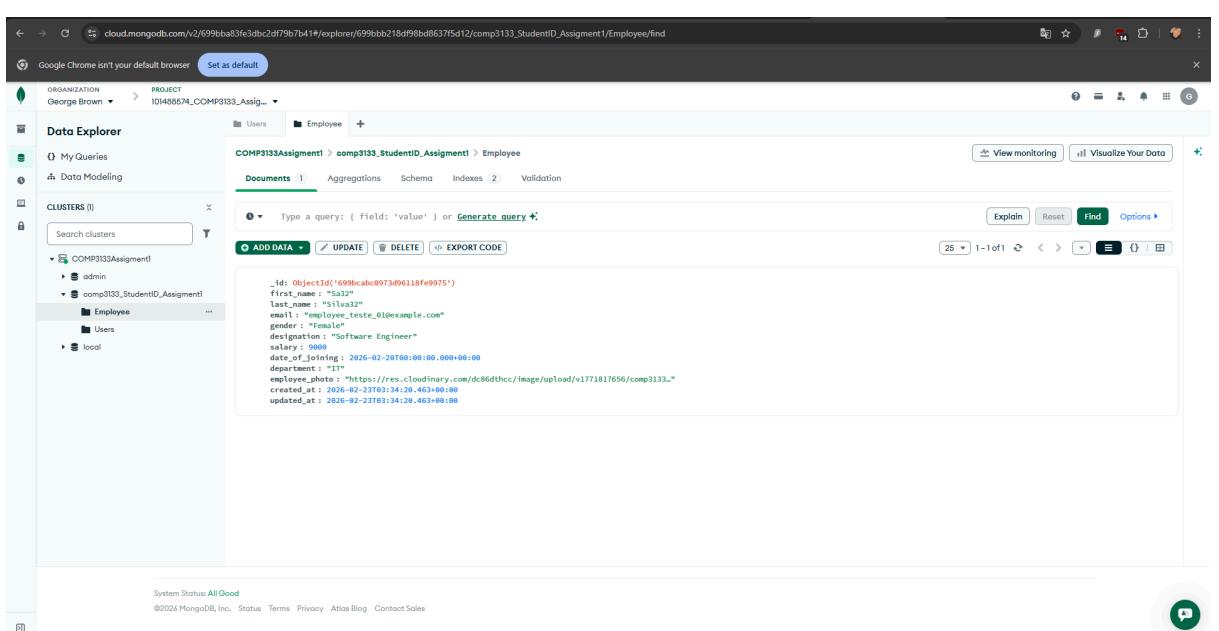


The screenshot shows the MongoDB Cloud interface for the 'Users' collection in the 'comp3133\_assignment' database. The left sidebar shows the project structure with 'COMP3133Assignment' selected, containing 'admin', 'Employee', 'Users', and 'local' clusters. The main panel displays two documents:

```
_id: ObjectId('699bc25656fafe1d7d565625')
username: "Miranda"
email: "gustavomiranda@example.com"
password: "$2as$08jN6A3/CiahoVrx36Kchn0UJ0wKOu1rF2y.GTR06sza3kmGpbwsW"
created_at: 2026-02-23T03:28:13.425+00:00
updated_at: 2026-02-23T03:28:13.425+00:00

_id: ObjectId('699bc2716973d96118f9962')
username: "Miranda"
email: "gustavomiranda@example.com"
password: "$2as$08jN6A3/CiahoVrx36Kchn0UJ0wKOu1rF2y.GTR06sza3kmGpbwsW"
created_at: 2026-02-23T03:28:17.291+00:00
updated_at: 2026-02-23T03:28:17.291+00:00
```

System Status: All Good



The screenshot shows the MongoDB Cloud interface for the 'Employee' collection in the 'comp3133\_assignment' database. The left sidebar shows the project structure with 'COMP3133Assignment' selected, containing 'admin', 'Employee', 'Users', and 'local' clusters. The main panel displays one document:

```
_id: ObjectId('699bcabc0973d96118f9975')
first_name: "Silva"
last_name: "Silva"
email: "employee teste_0@example.com"
gender: "Female"
designation: "Software Engineer"
salary: 9000
date_of_joining: 2026-02-28T00:00:00.000+00:00
department: "IT"
profile_photo: "https://res.cloudinary.com/dc86dthcc/image/upload/v1771817656/comp3133_"
created_at: 2026-02-23T03:34:20.463+00:00
updated_at: 2026-02-23T03:34:20.463+00:00
```

System Status: All Good