

COMP 3133 – FULL STACK DEVELOPMENT 2

ASSIGNMENT 01 SNAPSHOTS

GITHUB REPO LINK: https://github.com/inishimehta/101514172_COMP3133_Assignment01

MONGODB COLLECTIONS

users

The screenshot shows the MongoDB Compass interface for the 'users' collection. The interface includes tabs for Documents, Aggregations, Schema, Indexes, and Validation. The Documents tab is active, showing 3 documents. The query bar contains a query: { field: 'value' }. The documents are as follows:

```
{
  "_id": ObjectId('6985b334e3f135b37441cdc0'),
  "username": "nishil",
  "email": "nishil@test.com",
  "password": "$2b$12$9HlKwEC5wSobcoKu3k/064rTven1lEYrptHr0Q7.WnWbLACB581",
  "created_at": 2026-02-06T09:24:04.340+00:00,
  "updated_at": 2026-02-06T09:24:04.660+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('698c531b462b9d18c7611a63'),
  "username": "nishil",
  "email": "nishil@test.com",
  "password": "$2b$12$gRl1lomo/0k8V.8q7bTU609fPa272IkSHsCTOCE3jFeEFH0WRrX25",
  "created_at": 2026-02-11T09:59:55.435+00:00,
  "updated_at": 2026-02-11T09:59:55.894+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('698c63e69ee324c8c0b0d5ff'),
  "username": "testuser1",
  "email": "testuser1@test.com",
  "password": "$2b$10$2J1sF0D5Q012WwCEKyWU.u1dcfnt5T90sqwAJXdsfrLTnNe5.z3IW",
  "created_at": 2026-02-11T11:11:34.657+00:00,
  "updated_at": 2026-02-11T11:11:34.657+00:00,
  "__v": 0
}
```

employees

The screenshot shows the MongoDB Compass interface for the 'employees' collection. The interface includes tabs for Documents, Aggregations, Schema, Indexes, and Validation. The Documents tab is active, showing 2 documents. The query bar contains a query: { field: 'value' }. The documents are as follows:

```
{
  "_id": ObjectId('698c6e799ee324c8c0b0d703'),
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.doe3@test.com",
  "gender": "Male",
  "designation": "Senior Developer",
  "salary": 2000,
  "date_of_joining": 2026-02-11T00:00:00.000+00:00,
  "department": "IT",
  "employee_photo": "https://res.cloudinary.com/dnjkoou9/image/upload/v1770811000/comp3133-",
  "created_at": 2026-02-11T11:56:41.134+00:00,
  "updated_at": 2026-02-11T12:29:57.809+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId('698c78e29ee324c8c0b0d713'),
  "first_name": "Jane",
  "last_name": "Smith",
  "email": "jane.smith1@test.com",
  "gender": "Female",
  "designation": "QA",
  "salary": 1400,
  "date_of_joining": 2026-02-11T00:00:00.000+00:00,
  "department": "Quality",
  "employee_photo": "https://res.cloudinary.com/dnjkoou9/image/upload/v1770813666/comp3133-",
  "created_at": 2026-02-11T12:41:06.269+00:00,
  "updated_at": 2026-02-11T12:41:06.271+00:00,
  "__v": 0
}
```

SAMPLE USER DETAILS FOR LOGIN/TESTING

```
1  {
2    "query": "mutation($u:String!,$e:String!,$p:String!){ signup(username:$u,email:$e,password:$p){ token user{ _id username email } } }",
3    "variables": {
4      "u": "testuser1",
5      "e": "testuser1@test.com",
6      "p": "Pass1234!"
7    }
8  }
```

API TESTING SCREENSHOTS

1. SignUp (Mutation)

The screenshot shows a REST client interface with a tab for 'ASSIGNMENT_01 / Signup'. The method is 'POST' and the URL is 'http://localhost:4000/graphql'. The request body is a GraphQL mutation to sign up a user with the username 'testuser1', email 'testuser1@test.com', and password 'Pass1234!'. The response is a 200 OK status with a response time of 282 ms and a body size of 597 B. The response body is a JSON object containing a token and user information.

```
1 {
2   "query": "mutation($u:String!, $e:String!, $p:String!) { signup(username:$u, email:$e, password:$p) { token user { _id username email } } }",
3   "variables": { "u": "testuser1", "e": "testuser1@test.com", "p": "Pass1234!" }
4 }
5
```

```
1 {
2   "data": {
3     "signup": {
4       "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI20ThjNjNlNjllZTM5NGM4YzBiMGQ2ZmYiLCJpYXQiOiJlY3NzA4MDgyOTQsImV4cCI6MTc3MDg5NDY5NH0.-eXmkhRdkXBtBDoM00cKE3Q3F3hE8x_qiefm1DYVoIg",
5       "user": {
6         "_id": "698c63e69ee324c8c0b0d6ff",
7         "username": "testuser1",
8         "email": "testuser1@test.com"
9       }
10    }
11  }
12 }
```

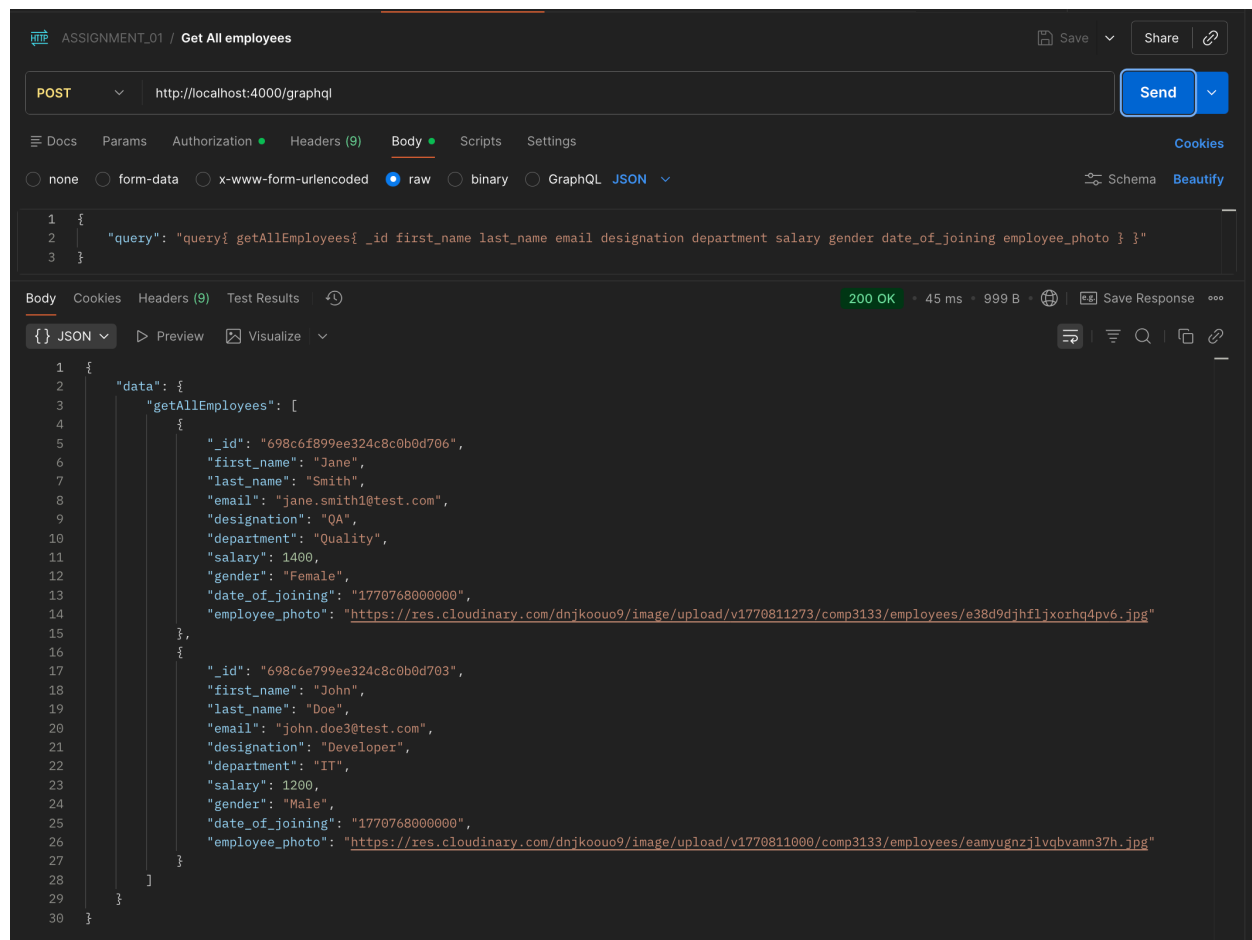
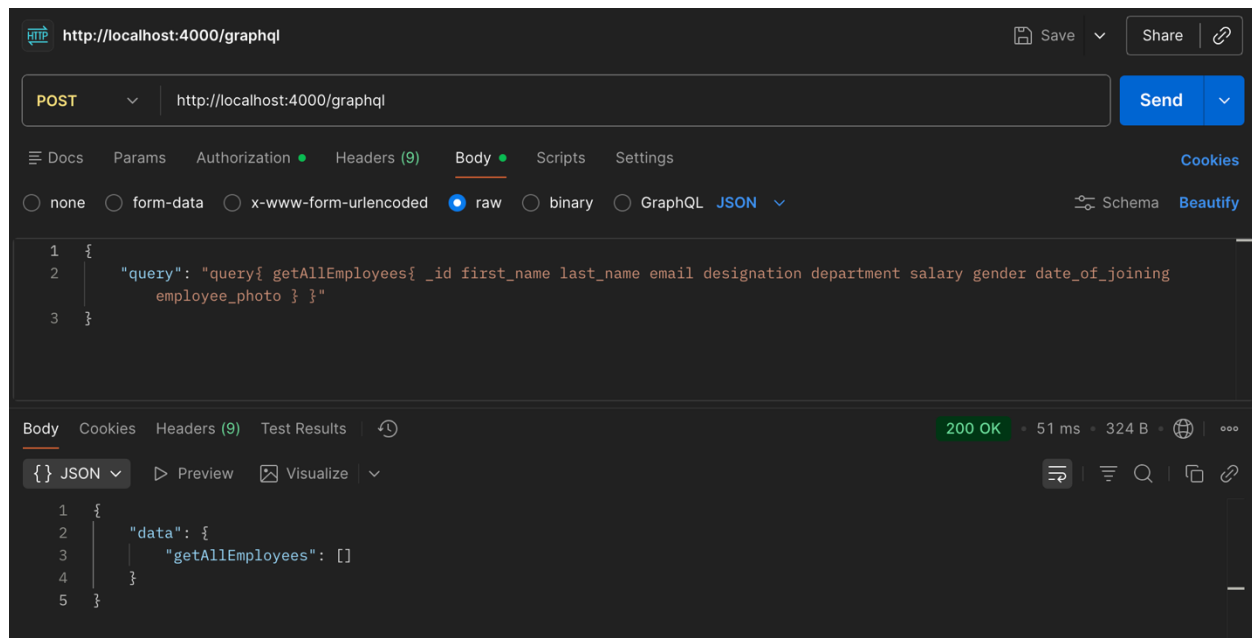
2. Login (Query)

The screenshot shows a REST client interface with a tab for 'ASSIGNMENT_01 / Login'. The method is 'POST' and the URL is 'http://localhost:4000/graphql'. The request body is a GraphQL query to login a user with the username 'testuser1' and password 'Pass1234!'. The response is a 200 OK status with a response time of 177 ms and a body size of 596 B. The response body is a JSON object containing a token and user information.

```
1 {
2   "query": "query($x:String!, $p:String!) { login(usernameOrEmail:$x, password:$p) { token user { _id username email } } }",
3   "variables": { "x": "testuser1", "p": "Pass1234!" }
4 }
5
```

```
1 {
2   "data": {
3     "login": {
4       "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI20ThjNjNlNjllZTM5NGM4YzBiMGQ2ZmYiLCJpYXQiOiJlY3NzA4MDg5MjMsImV4cCI6MTc3MDg5NTMyM30.iaE0K7piiuTfcgMG_3e8EwxAAhMxrTaz81TX6J6FALs",
5       "user": {
6         "_id": "698c63e69ee324c8c0b0d6ff",
7         "username": "testuser1",
8         "email": "testuser1@test.com"
9       }
10    }
11  }
12 }
```

3. Get All Employees (Query)



4. Add New Employee (Mutation)

The screenshot displays a REST client interface with a tab for 'POST Add New Employee'. The URL is 'http://localhost:4000/graphql'. The request body is a GraphQL mutation with variables for a new employee. The response is a 200 OK status with a JSON body containing the employee details.

Request:

```
1 {
2   "query": "mutation($in:EmployeeInput!){ addNewEmployee(input:$in){ _id first_name last_name email employee_photo designation department salary
3     gender date_of_joining } }",
4   "variables": {
5     "in": {
6       "first_name": "John",
7       "last_name": "Doe",
8       "email": "john.doe3@test.com",
9       "gender": "Male",
10      "designation": "Developer",
11      "salary": 1200,
12      "date_of_joining": "2026-02-11",
13      "department": "IT",
14      "employee_photo": "https://plus.unsplash.com/premium_photo-1739786996022-5ed5b56834e2?q=80&w=1480&auto=format&fit=crop&ixlib=rb-4.1.0&
15      ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVuLWVufDB8fHx8fA%3D%3D"
16    }
17  }
18 }
```

Response:

```
1 {
2   "data": {
3     "addNewEmployee": {
4       "_id": "698c6e799ee324c8c0b0d703",
5       "first_name": "John",
6       "last_name": "Doe",
7       "email": "john.doe3@test.com",
8       "employee_photo": "https://res.cloudinary.com/dnjkoouo9/image/upload/v1770811000/comp3133/employees/eamyugnzjlvqbvamn37h.jpg",
9       "designation": "Developer",
10      "department": "IT",
11      "salary": 1200,
12      "gender": "Male",
13      "date_of_joining": "1770768000000"
14    }
15  }
16 }
```

5. Search Employee with Id (Query)

The screenshot shows the GraphQL Playground interface. The query is a POST request to `http://localhost:4000/graphql`. The query body is:

```
1 {
2   "query": "query($id:ID!){ searchEmployeeByEid(eid:$id){ _id first_name last_name email designation department salary gender date_of_joining employee_photo } }",
3   "variables": {
4     "id": "698c6e799ee324c8c0b0d703"
5   }
6 }
```

The response is a 200 OK status, 50 ms, 662 B. The JSON response body is:

```
1 {
2   "data": {
3     "searchEmployeeByEid": {
4       "_id": "698c6e799ee324c8c0b0d703",
5       "first_name": "John",
6       "last_name": "Doe",
7       "email": "john.doe3@test.com",
8       "designation": "Developer",
9       "department": "IT",
10      "salary": 1200,
11      "gender": "Male",
12      "date_of_joining": "1770768000000",
13      "employee_photo": "https://res.cloudinary.com/dnjkoouo9/image/upload/v1770811000/comp3133/employees/eamyugnzjlvgbvamn37h.jpg"
14    }
15  }
16 }
```

6. Update Employee with Id (Mutation)

The screenshot shows the GraphQL Playground interface. The query is a POST request to `http://localhost:4000/graphql`. The query body is:

```
1 {
2   "query": "mutation($id:ID!,$in:EmployeeUpdateInput!){ updateEmployeeByEid(eid:$id,input:$in){ _id salary designation department updated_at employee_photo } }",
3   "variables": {
4     "id": "698c6e799ee324c8c0b0d703",
5     "in": {
6       "salary": 2000,
7       "designation": "Senior Developer"
8     }
9   }
10 }
```

The response is a 200 OK status, 105 ms, 581 B. The JSON response body is:

```
1 {
2   "data": {
3     "updateEmployeeByEid": {
4       "_id": "698c6e799ee324c8c0b0d703",
5       "salary": 2000,
6       "designation": "Senior Developer",
7       "department": "IT",
8       "updated_at": "1770812050293",
9       "employee_photo": "https://res.cloudinary.com/dnjkoouo9/image/upload/v1770811000/comp3133/employees/eamyugnzjlvgbvamn37h.jpg"
10     }
11   }
12 }
```

8.

8. Search Employee with Designation or Department (Query)

The screenshot shows the GraphQL Playground interface. The query is a POST request to `http://localhost:4000/graphql`. The query body is:

```
1 {
2   "query": "query($d:String,$dep:String){ searchEmployeeByDesignationOrDepartment(designation:$d,department:$dep){ _id first_name designation department } }",
3   "variables": { "d": "Senior Developer", "dep": null }
4 }
```

The response is a 200 OK status with a response time of 66 ms and a body size of 454 B. The response body is JSON:

```
1 {
2   "data": {
3     "searchEmployeeByDesignationOrDepartment": [
4       {
5         "_id": "698c6e799ee324c8c0b0d703",
6         "first_name": "John",
7         "designation": "Senior Developer",
8         "department": "IT"
9       }
10    ]
11  }
12 }
```

7. Delete Employee by Id (Mutation)

The screenshot shows the GraphQL Playground interface. The query is a POST request to `http://localhost:4000/graphql`. The query body is:

```
1 {
2   "query": "mutation($id:ID!){ deleteEmployeeByEid(eid:$id) }",
3   "variables": { "id": "698c6f899ee324c8c0b0d706" }
4 }
```

The response is a 200 OK status with a response time of 59 ms and a body size of 330 B. The response body is JSON:

```
1 {
2   "data": {
3     "deleteEmployeeByEid": true
4   }
5 }
```

Employees after updating and deleting (Query)

The screenshot displays a REST client interface with a dark theme. At the top, there are tabs for different API endpoints: "POST Delete Employee with", "POST Get All employees" (which is selected), "POST Update Employee with", and "POST Search Employee with". The selected tab shows the URL "http://localhost:4000/graphql" and a "Send" button. Below the URL bar, there are tabs for "Docs", "Params", "Authorization", "Headers (9)", "Body" (selected), "Scripts", and "Settings". The "Body" tab shows a GraphQL query:

```
1 {
2   "query": "query{ getAllEmployees{ _id first_name last_name email designation department salary gender date_of_joining employee_photo } }"
3 }
```

 Below the query, there are tabs for "Body", "Cookies", "Headers (9)", and "Test Results". The "Body" tab shows the response in JSON format:

```
1 {
2   "data": {
3     "getAllEmployees": [
4       {
5         "_id": "698c6e799ee324c8c0b0d703",
6         "first_name": "John",
7         "last_name": "Doe",
8         "email": "john.doe3@test.com",
9         "designation": "Senior Developer",
10        "department": "IT",
11        "salary": 2000,
12        "gender": "Male",
13        "date_of_joining": "1770768000000",
14        "employee_photo": "https://res.cloudinary.com/dnjkoouo9/image/upload/v1770811000/comp3133/employees/eamyugnzjlvqbvamn37h.jpg"
15      }
16    ]
17  }
18 }
```

 The response status is "200 OK" with a response time of "38 ms" and a size of "667 B". There are also buttons for "Save Response" and "Beautify".