CODEBOOSTERS TECH - GRAPHQL + EXPRESS HANDS ON TRAINING

WWW.CODEBOOSTERS.IN

🍕 Example 1: Introduction to GraphQL with Express - "Pizza Ordering System"

🍕 Short Story:

We are opening an online **Pizza Shop**! 🍕🛒
Customers can **view available pizzas** and **place orders** using **GraphQL queries** and **mutations**.
Let's **setup GraphQL server with Express.js** for this!

📜 Step-by-Step Detailed Explanation:

---

1. Install Required Packages

First, we need to install the following:

Package	Purpose
express	To create the Node.js server
express-graphql	To connect Express with GraphQL

Package	Purpose
graphql	To create GraphQL schemas, types, queries, mutations

```bash
npm init -y
npm install express express-graphql graphql
```

---

2. Create `server.js` File

Now we start coding our **Pizza API**.

```javascript
// 1. Importing necessary libraries
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const {
  GraphQLSchema,
  GraphQLObjectType,
  GraphQLString,
  GraphQLList,
  GraphQLInt,
  GraphQLNonNull
} = require('graphql');

// 2. Create an Express app
const app = express();

// 3. Mock Pizza Data
const pizzas = [
  { id: 1, name: 'Margherita', size: 'Medium' },
  { id: 2, name: 'Pepperoni', size: 'Large' },
  { id: 3, name: 'Veggie Delight', size: 'Small' }
];

// 4. Define the Pizza GraphQL Type
const PizzaType = new GraphQLObjectType({
  name: 'Pizza',
```

```
  fields: {
    id: { type: GraphQLInt },
    name: { type: GraphQLString },
    size: { type: GraphQLString }
  }
});

// 5. Define the Root Query (to fetch pizzas)
const RootQuery = new GraphQLObjectType({
  name: 'Query',
  fields: {
    pizzas: {
      type: new GraphQLList(PizzaType),
      description: 'List of all pizzas',
      resolve: () => pizzas // returns pizza array
    },
    pizza: {
      type: PizzaType,
      description: 'A Single Pizza',
      args: {
        id: { type: GraphQLInt }
      },
      resolve: (parent, args) => pizzas.find(pizza => pizza.id === args.id) // find
by ID
    }
  }
});

// 6. Define Mutation (to place a pizza order)
const Mutation = new GraphQLObjectType({
  name: 'Mutation',
  fields: {
    addPizza: {
      type: PizzaType,
      description: 'Add a new pizza',
      args: {
        name: { type: new GraphQLNonNull(GraphQLString) },
        size: { type: new GraphQLNonNull(GraphQLString) }
      },
      resolve: (parent, args) => {
        const newPizza = {
          id: pizzas.length + 1, // Auto-increment ID
```

```
          name: args.name,
          size: args.size
        };
        pizzas.push(newPizza); // Add to the list
        return newPizza; // Return the new pizza
      }
    }
  }
});

// 7. Create the GraphQL Schema
const schema = new GraphQLSchema({
  query: RootQuery,
  mutation: Mutation
});

// 8. Middleware to set up GraphQL endpoint
app.use('/graphql', graphqlHTTP({
  schema: schema,
  graphiql: true // enables GraphiQL tool
}));

// 9. Start the Express server
app.listen(4000, () => {
  console.log('🍕 Pizza server running at http://localhost:4000/graphql');
});
```

🧠 What each part means:

Section	Purpose
`PizzaType`	Defines how a pizza object looks (id, name, size)
`RootQuery`	Allows clients to fetch pizzas or a pizza by id
`Mutation`	Allows clients to create a new pizza order
`graphqlHTTP`	Connects GraphQL with Express server

Section	Purpose
`app.listen`	Starts server at localhost:4000

📥 Test Inputs & Outputs

Query to get all pizzas

```graphql
{
  pizzas {
    id
    name
    size
  }
}
```

✅ Output:

```json
{
  "data": {
    "pizzas": [
      { "id": 1, "name": "Margherita", "size": "Medium" },
      { "id": 2, "name": "Pepperoni", "size": "Large" },
      { "id": 3, "name": "Veggie Delight", "size": "Small" }
    ]
  }
}
```

Query to get a single pizza

```graphql
{
  pizza(id: 2) {
    name
    size
  }
}
```

✅ Output:

```json
{
  "data": {
    "pizza": {
      "name": "Pepperoni",
      "size": "Large"
    }
  }
}
```

Mutation to add a new pizza

```graphql
mutation {
  addPizza(name: "BBQ Chicken", size: "Large") {
    id
    name
    size
  }
}
```

✅ Output:

```json
```

```
{
  "data": {
    "addPizza": {
      "id": 4,
      "name": "BBQ Chicken",
      "size": "Large"
    }
  }
}
```

✅ Full Working Code (Copy-Paste Ready)

```javascript
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const {
  GraphQLSchema,
  GraphQLObjectType,
  GraphQLString,
  GraphQLList,
  GraphQLInt,
  GraphQLNonNull
} = require('graphql');

const app = express();

const pizzas = [
  { id: 1, name: 'Margherita', size: 'Medium' },
  { id: 2, name: 'Pepperoni', size: 'Large' },
  { id: 3, name: 'Veggie Delight', size: 'Small' }
];

const PizzaType = new GraphQLObjectType({
  name: 'Pizza',
  fields: {
```

```
    id: { type: GraphQLInt },
    name: { type: GraphQLString },
    size: { type: GraphQLString }
  }
});

const RootQuery = new GraphQLObjectType({
  name: 'Query',
  fields: {
    pizzas: {
      type: new GraphQLList(PizzaType),
      description: 'List of all pizzas',
      resolve: () => pizzas
    },
    pizza: {
      type: PizzaType,
      description: 'A Single Pizza',
      args: {
        id: { type: GraphQLInt }
      },
      resolve: (parent, args) => pizzas.find(pizza => pizza.id === args.id)
    }
  }
});

const Mutation = new GraphQLObjectType({
  name: 'Mutation',
  fields: {
    addPizza: {
      type: PizzaType,
      description: 'Add a new pizza',
      args: {
        name: { type: new GraphQLNonNull(GraphQLString) },
        size: { type: new GraphQLNonNull(GraphQLString) }
      },
      resolve: (parent, args) => {
        const newPizza = {
          id: pizzas.length + 1,
          name: args.name,
          size: args.size
        };
        pizzas.push(newPizza);
```

```
        return newPizza;
      }
    }
  }
});

const schema = new GraphQLSchema({
  query: RootQuery,
  mutation: Mutation
});

app.use('/graphql', graphqlHTTP({
  schema: schema,
  graphiql: true
}));

app.listen(4000, () => {
  console.log('🍕 Pizza server running at http://localhost:4000/graphql');
});
```

🎬 Example 2: Setting up GraphQL with Express - "Movie Review System" 🍿

🎥 Short Story:

Imagine we are building a **Movie Review App**! 🍿🎬
Users can **view available movies** and **submit new movie reviews** using **GraphQL Queries and Mutations**.
We will set up a **GraphQL server with Express** and **use mock data** for movies.

📜 Step-by-Step Line-by-Line Explanation:

---

1. Install Required Packages

We only need **Express**, **GraphQL**, and **express-graphql** (same as before).

If not installed yet:

```bash
npm install express express-graphql graphql
```

---

2. Create a new `movieServer.js` File

Start coding!

```javascript
// 1. Import express and graphql modules
const express = require('express');
```

```
const { graphqlHTTP } = require('express-graphql');
const {
  GraphQLSchema,
  GraphQLObjectType,
  GraphQLString,
  GraphQLList,
  GraphQLInt,
  GraphQLNonNull
} = require('graphql');

// 2. Create express app
const app = express();

// 3. Create Mock Movie Data 🎥
const movies = [
  { id: 1, title: 'Interstellar', director: 'Christopher Nolan', rating: 9 },
  { id: 2, title: 'Inception', director: 'Christopher Nolan', rating: 8 },
  { id: 3, title: 'Coco', director: 'Lee Unkrich', rating: 9 }
];

// 4. Define Movie Type
const MovieType = new GraphQLObjectType({
  name: 'Movie',
  fields: {
    id: { type: GraphQLInt },
    title: { type: GraphQLString },
    director: { type: GraphQLString },
    rating: { type: GraphQLInt }
  }
});

// 5. Define Root Query
const RootQuery = new GraphQLObjectType({
  name: 'Query',
  fields: {
    movies: {
      type: new GraphQLList(MovieType),
      description: 'List of All Movies',
      resolve: () => movies // Return all movies
    },
    movie: {
      type: MovieType,
```

```
      description: 'Single Movie by ID',
      args: {
        id: { type: GraphQLInt }
      },
      resolve: (parent, args) => movies.find(movie => movie.id === args.id) // Find
movie by ID
    }
  }
});

// 6. Define Mutations
const Mutation = new GraphQLObjectType({
  name: 'Mutation',
  fields: {
    addMovie: {
      type: MovieType,
      description: 'Add a New Movie',
      args: {
        title: { type: new GraphQLNonNull(GraphQLString) },
        director: { type: new GraphQLNonNull(GraphQLString) },
        rating: { type: new GraphQLNonNull(GraphQLInt) }
      },
      resolve: (parent, args) => {
        const newMovie = {
          id: movies.length + 1,
          title: args.title,
          director: args.director,
          rating: args.rating
        };
        movies.push(newMovie); // Add movie to mock array
        return newMovie;
      }
    }
  }
});

// 7. Create the GraphQL Schema
const schema = new GraphQLSchema({
  query: RootQuery,
  mutation: Mutation
});
```

```
// 8. Set up GraphQL middleware
app.use('/graphql', graphqlHTTP({
  schema: schema,
  graphiql: true
}));

// 9. Start the Server
app.listen(5000, () => {
  console.log('🎥 Movie server running at http://localhost:5000/graphql');
});
```

🧠 Explanation of Each Part:

Section	Purpose
MovieType	Defines a movie (id, title, director, rating)
RootQuery	Allows users to fetch all movies or one by id
Mutation	Allows users to add a new movie review
graphqlHTTP	Connects GraphQL and Express together
app.listen	Starts our server

📥 Test Inputs & Outputs

🧪 Query: Get All Movies

```graphql
```

```
{
  movies {
    id
    title
    director
    rating
  }
}
```

✅ Output:

```json
{
  "data": {
    "movies": [
      { "id": 1, "title": "Interstellar", "director": "Christopher Nolan", "rating": 9 },
      { "id": 2, "title": "Inception", "director": "Christopher Nolan", "rating": 8 },
      { "id": 3, "title": "Coco", "director": "Lee Unkrich", "rating": 9 }
    ]
  }
}
```

🧪 Query: Get Single Movie

```graphql
{
  movie(id: 1) {
    title
    director
    rating
  }
}
```

✅ Output:

```json
{
  "data": {
    "movie": {
      "title": "Interstellar",
      "director": "Christopher Nolan",
      "rating": 9
    }
  }
}
```

🧪 Mutation: Add New Movie

```graphql
mutation {
  addMovie(title: "Soul", director: "Pete Docter", rating: 8) {
    id
    title
    director
    rating
  }
}
```

✅ Output:

```json
{
  "data": {
    "addMovie": {
      "id": 4,
      "title": "Soul",
      "director": "Pete Docter",
      "rating": 8
    }
  }
}
```

✅ Full Working Code (Copy-Paste Ready)

```javascript
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const {
  GraphQLSchema,
  GraphQLObjectType,
  GraphQLString,
  GraphQLList,
  GraphQLInt,
  GraphQLNonNull
} = require('graphql');

const app = express();

const movies = [
  { id: 1, title: 'Interstellar', director: 'Christopher Nolan', rating: 9 },
  { id: 2, title: 'Inception', director: 'Christopher Nolan', rating: 8 },
  { id: 3, title: 'Coco', director: 'Lee Unkrich', rating: 9 }
];

const MovieType = new GraphQLObjectType({
  name: 'Movie',
  fields: {
    id: { type: GraphQLInt },
    title: { type: GraphQLString },
    director: { type: GraphQLString },
    rating: { type: GraphQLInt }
  }
});

const RootQuery = new GraphQLObjectType({
  name: 'Query',
  fields: {
    movies: {
      type: new GraphQLList(MovieType),
```

```
      description: 'List of All Movies',
      resolve: () => movies
    },
    movie: {
      type: MovieType,
      description: 'Single Movie by ID',
      args: {
        id: { type: GraphQLInt }
      },
      resolve: (parent, args) => movies.find(movie => movie.id === args.id)
    }
  }
});

const Mutation = new GraphQLObjectType({
  name: 'Mutation',
  fields: {
    addMovie: {
      type: MovieType,
      description: 'Add a New Movie',
      args: {
        title: { type: new GraphQLNonNull(GraphQLString) },
        director: { type: new GraphQLNonNull(GraphQLString) },
        rating: { type: new GraphQLNonNull(GraphQLInt) }
      },
      resolve: (parent, args) => {
        const newMovie = {
          id: movies.length + 1,
          title: args.title,
          director: args.director,
          rating: args.rating
        };
        movies.push(newMovie);
        return newMovie;
      }
    }
  }
});

const schema = new GraphQLSchema({
  query: RootQuery,
  mutation: Mutation
```

```
});

app.use('/graphql', graphqlHTTP({
  schema: schema,
  graphiql: true
}));

app.listen(5000, () => {
  console.log('🎥 Movie server running at http://localhost:5000/graphql');
});
```