FSD Lab Program 7 (Using GraphQL)

Title : Create Rest API and GraphQL the allows the user to access and manage the products.

Problem description:. Set up a basic server that responds with "Hello, Express!" at the root endpoint (GET /). Implement endpoints for a Product resource: GET : Returns a list of products. POST Adds a new product. GET /:id: Returns details of a specific product. PUT /:id: Updates an existing product. DELETE /:id: Deletes a product. Add middleware to log requests to the console. Use express.json() to parse incoming JSON payloads.

Method: Install Express (npm install express). Use apollo server for GraphQL API.
Write endpoints and test the API using Postman.(Download postman)

Step-1 Create a Project Folder (In cmd Prompt Run the following Commands) and install express

```
mkdir product-graphql
cd product-graphql
npm init –y
npm install express @apollo/server graphql graphql-tag body-parser cors
code . (it will open folder in VS Code)
```

Step-2 Create File product.js in product-graphql

product.js

```
let products = [
  { id: 1, name: "Laptop", price: 1000 },
  { id: 2, name: "Phone", price: 500 }
];

export default products;
```

Step-3 Create a File server.js

Server.js

```
import express from "express";
import { ApolloServer } from "@apollo/server";
import { expressMiddleware } from "@apollo/server/express4";
import { ApolloServerPluginLandingPageLocalDefault } from
"@apollo/server/plugin/landingPage/default";
import cors from "cors";
import bodyParser from "body-parser";
import { gql } from "graphql-tag";
import products from "./product.js";

const typeDefs = gql`
  type Product { id: ID!, name: String!, price: Float! }
  type Query { hello: String, products: [Product], product(id: ID!): Product }
  type Mutation {
    addProduct(name: String!, price: Float!): Product
    updateProduct(id: ID!, name: String, price: Float): Product
    deleteProduct(id: ID!): String
  }
`;

const resolvers = {
  Query: {
    hello: () => "Hello, GraphQL!",
    products: () => products,
    product: (_, { id }) => products.find(p => p.id == id),
  },
  Mutation: {

    addProduct: (_, { name, price }) => {
      const p = { id: Date.now(), name, price };
      products.push(p);
      return p;
    },

    updateProduct: (_, { id, name, price }) => {
      const p = products.find(p => p.id == id);
      if (!p) throw new Error("Product not found");
      if (name) p.name = name;
      if (price) p.price = price;
      return p;
    },

    deleteProduct: (_, { id }) => {
      const i = products.findIndex(p => p.id == id);
      if (i === -1) throw new Error("Product not found");
      products.splice(i, 1);
      return `Product ${id} deleted`;
    },
  },
};

const app = express();
app.use(cors(), bodyParser.json());

const server = new ApolloServer({
  typeDefs,
  resolvers,
  plugins: [ApolloServerPluginLandingPageLocalDefault({ embed: true })],
});

await server.start();
app.use("/graphql", expressMiddleware(server));

app.listen(5000, () => {
  console.log(" GraphQL server ready at http://localhost:5000/graphql");
});
```

Step -4 Edit package.json

Package.json

```
{
  "name": "product-graphql",
  "version": "1.0.0",
  "main": "server.js",
  "type": "module",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "@apollo/server": "^4.11.0",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.21.2",
    "graphql": "^16.11.0",
    "graphql-tag": "^2.12.6"
  }
}
```

Step-5 Run the Server

```
npm install
npm start
```

o/p: GraphQL at http://localhost:5000/graphql

Step-6 Open the Browser go to http://localhost:5000/graphql and run GraphQL queries

Action	Type	Query Name	Execute this query in GraphQL Interface
Fetch all	Query	products	query { products { id name price } }
Fetch one	Query	product(id: ID!)	query { product(id:1){name price} }
Add new	Mutation	addProduct(name:String!, price:Float!)	mutation { addProduct(name:"Tablet", price:700){id name price} }
Update	Mutation	updateProduct(id:ID!, name:String, price:Float)	mutation { updateProduct(id:1, name:"Laptop", price:1200){id name price} }
Delete	Mutation	deleteProduct(id:ID!)	mutation { deleteProduct(id:2) }