**Assignment 10: GraphQL and Apollo Client**

**Learn the basics of GraphQL.**

**Integrate GraphQL with your MERN stack using Apollo Client.**

**GraphQL is a query language for APIs that enables clients to request only the data they need. Apollo Client is a powerful GraphQL client library that helps you manage state and interact with GraphQL APIs in your front-end application. In this assignment, I'll guide you through the process of integrating GraphQL with your MERN stack using Apollo Client.**

**Step 1: Set Up a GraphQL Server**

**Create a GraphQL server using Apollo Server in your Express.js app.**

**Step 1.1: Install Dependencies**

npm install apollo-server-express graphql

Step 1.2: Create a GraphQL Schema and Resolvers

Create a file graphql/schema.js:

| // graphql/schema.js  const { gql } = require('apollo-server-express');  const typeDefs = gql`  type Query {  getData: [Data]  }  type Data {  id: ID!  title: String!  }  `;  const data = [  { id: '1', title: 'Item 1' },  { id: '2', title: 'Item 2' },  { id: '3', title: 'Item 3' },  ];  const resolvers = {  Query: {  getData: () => data,  },  };  module.exports = { typeDefs, resolvers }; |
| --- |

Step 1.3: Integrate Apollo Server in Express.js

Update express-api/app.js to include Apollo Server:

| // express-api/app.js  const express = require('express');  const { ApolloServer } = require('apollo-server-express');  const { typeDefs, resolvers } = require('./graphql/schema');  const app = express();  const server = new ApolloServer({ typeDefs, resolvers });  server.applyMiddleware({ app });  const port = 3001;  app.listen(port, () => {  console.log(`Express API with GraphQL is listening at http://localhost:${port}`);  }); |
| --- |

Step 2: Set Up Apollo Client in React

Step 2.1: Install Dependencies

npm install @apollo/client graphql

Step 2.2: Create an Apollo Client

Update src/index.js to include Apollo Client:

| // src/index.js  import React from 'react';  import ReactDOM from 'react-dom';  import { ApolloProvider } from '@apollo/client';  import { ApolloClient, InMemoryCache } from '@apollo/client';  import './index.css';  import App from './App';  import reportWebVitals from './reportWebVitals';  const client = new ApolloClient({  uri: 'http://localhost:3001/graphql', // Adjust the URL based on your server configuration  cache: new InMemoryCache(),  });  ReactDOM.render(  <ApolloProvider client={client}>  <React.StrictMode>  <App />  </React.StrictMode>  </ApolloProvider>,  document.getElementById('root')  );  reportWebVitals(); |
| --- |

Step 3: Fetch Data Using Apollo Client in React

Update your React component (src/DataFetching.js) to fetch data using Apollo Client:

| // src/DataFetching.js  import React from 'react';  import { useQuery, gql } from '@apollo/client';  const GET\_DATA = gql`  query {  getData {  id  title  }  }  `;  const DataFetching = () => {  const { loading, error, data } = useQuery(GET\_DATA);  if (loading) return <p>Loading...</p>;  if (error) return <p>Error: {error.message}</p>;  return (  <div>  <h2>Data Fetching with Apollo Client</h2>  <ul>  {data.getData.map((item) => (  <li key={item.id}>{item.title}</li>  ))}  </ul>  </div>  );  };  export default DataFetching; |
| --- |

Step 4: Run Your MERN App

Start both your Express.js server and React development server:

| # Start Express.js server  node express-api/app.js  # Start React development server  npm start |
| --- |

Visit http://localhost:3000 in your browser. You should see your React app fetching and displaying data from the GraphQL server using Apollo Client.

This is a basic example of integrating GraphQL with a MERN app using Apollo Client. You can further extend this by implementing mutations, handling variables, and incorporating more complex GraphQL features based on your application's requirements.