query GetStory($storiesId: Int!) {

getStory(storiesId: $storiesId) {

id

posterId

title

content

status

created

}

}

This is a GraphQL query called GetStory that accepts a single variable called $storiesId, which should be an integer. Here's an example of how you could use this query to retrieve the details of a specific story:

graphql

Copy code

{

"storiesId": 123

}

In this example, we're setting the $storiesId variable to 123, which is the ID of the story that we want to retrieve.

The query itself requests several fields from the getStory endpoint, including the id of the story, the posterId of the user who posted it, the title and content of the story itself, its status (e.g. whether it's published or still a draft), and the created timestamp for when the story was originally created.

To execute this query, you would typically send it to a GraphQL API endpoint along with any necessary authentication tokens or headers, and then parse the response to extract the data that you need.

To integrate this GraphQL query into a frontend application, you can use a GraphQL client library such as Apollo Client or Relay.

Here's an example of how you could use Apollo Client to execute the GetStory query and display the results in a React component:

jsx

Copy code

import { useQuery, gql } from '@apollo/client';

const GET\_STORY = gql`

query GetStory($storiesId: Int!) {

getStory(storiesId: $storiesId) {

id

posterId

title

content

status

created

}

}

`;

function StoryDetails({ storiesId }) {

const { loading, error, data } = useQuery(GET\_STORY, {

variables: { storiesId },

});

if (loading) return <p>Loading...</p>;

if (error) return <p>Error :(</p>;

return (

<div>

<h1>{data.getStory.title}</h1>

<p>{data.getStory.content}</p>

<p>Posted by {data.getStory.posterId}</p>

<p>Status: {data.getStory.status}</p>

<p>Created on {data.getStory.created}</p>

</div>

);

}

In this example, we're defining the GET\_STORY query using the gql tag from Apollo Client. We're also using the useQuery hook to execute the query and retrieve the data.

We're passing the storiesId variable to the query using the variables option in useQuery. This value could be passed in as a prop from a parent component, or retrieved from the URL params using a library like React Router.

Once the data is loaded, we're displaying various fields from the getStory response in the component's UI. We're also handling loading and error states using conditional rendering.