To connect your React Native application to a MySQL database using a Spring Boot backend, you'll need to follow these general steps:

1. **Set up your Spring Boot backend:**
   * Create a REST API that interacts with your MySQL database. This will involve creating controllers, services, and repositories, as well as configuring your database connection.
2. **Send requests from your React Native application:**
   * Install a library for sending HTTP requests, such as axios or fetch.
   * In your React Native code, when the user submits the form, send a request to your Spring Boot API. The request should include the user's input as data.

To set up a Spring Boot application with MySQL and test it with Postman, follow these steps:

1. **Set up your Spring Boot application:**
   * Add the necessary dependencies in your pom.xml or build.gradle file. You'll need spring-boot-starter-data-jpa, spring-boot-starter-web, and mysql-connector-java.
   * Configure your MySQL database connection in application.properties or application.yml.
   * Create your entity classes, which represent your database tables.
   * Create your repository interfaces, which extend JpaRepository or CrudRepository.
   * Create your service classes, which contain your business logic.
   * Create your controller classes, which handle HTTP requests and responses.
2. **Test your application with Postman:**
   * Start your Spring Boot application.
   * Open Postman.
   * Create a new request in Postman. Set the HTTP method (GET, POST, PUT, DELETE, etc.) and the request URL to match one of your endpoints.
   * If necessary, set the request headers and body. For example, if you're testing a POST request, you might need to set the Content-Type header to application/json and provide a JSON object in the request body.
   * Send the request and check the response.

Remember, this is a high-level overview. The exact steps and code will depend on your specific requirements. If you need help with a specific step or piece of code, feel free to ask!

so we use postman api only to test right?

Yes, that's correct. Postman is a tool that is commonly used for testing APIs. It allows you to send HTTP requests to your API endpoints and view the responses. This makes it easier to test and debug your APIs, as you can manually trigger requests and inspect the results.

With Postman, you can:

* Send requests with any HTTP method (GET, POST, PUT, DELETE, etc.)
* Set request headers and bodies
* View response status codes, headers, and bodies
* Save requests for later use
* Organize requests into collections
* Write tests for your requests
* Automate tests using the collection runner or Newman

However, Postman is not part of your application's production environment. It's a separate tool that you use during development and testing. In production, your API will be called by other applications or services, not by Postman.

Yes, MySQL server typically runs on port 3306 by default. Your Spring Boot application connects to it using the connection details you provide in your application.properties or application.yml file.

Here's an example of how you might configure your MySQL connection in application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/db\_name

spring.datasource.username=db\_user

spring.datasource.password=db\_password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

In this configuration:

* spring.datasource.url is the URL of your MySQL server. localhost:3306 is the address and port, and db\_name is the name of your database.
* spring.datasource.username and spring.datasource.password are your MySQL username and password.
* spring.datasource.driver-class-name is the driver class name for MySQL.

With this configuration, Spring Boot will automatically set up a connection pool to your MySQL server when it starts up. You can then use Spring Data JPA to interact with your database.

Jpa – connects java objects to the relational databases,

Or

Spring Boot JPA is a Java specification for managing relational data in Java applications. It allows us to access and persist data between Java object/ class and relational database. JPA follows Object-Relation Mapping (ORM).

Axios.get --. Searches the data through database- data retrieval.

Axios.put – update the database by putting new data.

Axios.delete – deletes the data in the database.

Axios.post – insert/ add data to the database.

`@PostMapping` is an annotation in Spring Framework, which is used to map HTTP POST requests onto specific handler methods. It is a composed annotation that acts as a shortcut for `@RequestMapping(method = RequestMethod.POST)`.

HTTP POST is a request method supported by HTTP used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request.

In the context of a REST API, POST methods are often used to create new resources. For example, in a blog application, a POST request might be used to create a new blog post, with the content of the post included in the request body.

Here's an example of a `@PostMapping` in a Spring controller:

```java

@PostMapping("/books")

public Book newBook(@RequestBody Book newBook) {

return bookRepository.save(newBook);

}

```

In this example, a POST request to `/books` will trigger the `newBook` method, creating a new book in the database with the details provided in the request body.

Yes, in this context, your development machine is your laptop where your server (Spring Boot application) is running.

IPv4 stands for Internet Protocol version 4. It is the fourth version of the Internet Protocol (IP) and it is the core protocol that routes most of the traffic on the Internet. An IPv4 address is a numerical label assigned to each device participating in a computer network that uses the Internet Protocol for communication.

When you're running a server on your laptop, it has an IP address that other devices on the same network can use to access it. This is known as its local IP address. If you're running your React Native app on a physical mobile device, you need to use this local IP address to make requests to your server.

You can usually find your local IP address in your network settings or by using a command in your terminal. It will typically look something like `192.168.1.5`.