1. Axios vs Fetch API

Both Axios and Fetch API are used to make HTTP requests from the frontend.

Feature	Axios	Fetch API
Browser Support	IE11+ and modern browsers	Modern browsers (IE not supported without polyfill)
Syntax	`axios.get/post(url, data)`	`fetch(url, { method, body })`
Response Handling	Automatically parses JSON	Must manually parse JSON (`res.json()`)
Error Handling	Handles HTTP errors automatically	Only network errors are caught; HTTP errors must be handled manually
Request Timeout	Built-in support	Must implement manually
Intercept Requests/Responses	Yes, easy	No, need to wrap manually
File Upload	Easy via FormData	Supported but more verbose

2. Frontend (React) Examples

Using Axios

```
import React, { useState } from "react";
import axios from "axios";

export default function UserForm() {
  const [data, setData] = useState({ name: "", email: "" });

  const handleChange = e => setData({ ...data, [e.target.name]:
e.target.value });

  const handleSubmit = async e => {
    e.preventDefault();
    try {
      const response = await
axios.post("http://127.0.0.1:8000/api/users", data);
      console.log("Success:", response.data);
    } catch (error) {
      console.error("Error:", error.response?.data ||
error.message);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="name" placeholder="Name" onChange={handleChange}
/>
      <input name="email" placeholder="Email"
onChange={handleChange} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

Using Fetch API

```
import React, { useState } from "react";

export default function UserForm() {
  const [data, setData] = useState({ name: "", email: "" });

  const handleChange = e => setData({ ...data, [e.target.name]:
e.target.value });

  const handleSubmit = async e => {
    e.preventDefault();
    try {
      const response = await
fetch("http://127.0.0.1:8000/api/users", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(data)
      });

      if (!response.ok) throw new Error(`HTTP error! status:
${response.status}`);
      const result = await response.json();
      console.log("Success:", result);
    } catch (error) {
      console.error("Error:", error.message);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="name" placeholder="Name" onChange={handleChange}
/>
      <input name="email" placeholder="Email"
onChange={handleChange} />
      <button type="submit">Submit</button>
    </form>
  );
}
```

3. Backend (PHP Laravel)

Step 1: Create Route

`routes/api.php`:

```
use App\Http\Controllers\UserController;

Route::post('/users', [UserController::class, 'store']);
```

Step 2: Create Controller

```
php artisan make:controller UserController
```

`app/Http/Controllers/UserController.php`:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```php
use App\Models\User;

class UserController extends Controller
{
    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|unique:users',
        ]);

        $user = User::create($validated);

        return response()->json([
            'message' => 'User created successfully',
            'data' => $user
        ], 201);
    }
}
```

Step 3: Model Fillable

`app/Models/User.php`:

```php
protected $fillable = ['name', 'email'];
```

Step 4: Enable CORS

`config/cors.php`:

```php
'paths' => ['api/*'],
'allowed_methods' => ['*'],
'allowed_origins' => ['http://localhost:3000'],
'allowed_headers' => ['*'],
```

4. Tips & Best Practices

- **Axios**

    - Best for larger apps due to interceptors and automatic JSON parsing.
    - Easy to cancel requests (`axios.CancelToken`).
- **Fetch API**
    - Native to the browser, no extra dependency.
    - Needs manual JSON parsing and error handling.
- **Laravel**
    - Always validate incoming requests.
    - Use `api.php` routes for REST APIs.
    - Ensure CORS is configured correctly to avoid frontend errors.