## Network Debugging

The 'Network' tab allows you to inspect the traffic between the client and the server, which can help pinpoint any issues related to failed form submissions, whether they are related to client-side validation, server errors, or network issues. Alongside, ‘Console’ tab will help you see any errors / warning being produced on the browser side.

## Step-by-Step Instructions

1. Open the Developer Tools: To begin debugging, right-click anywhere on the page, and select 'Inspect' or press `Ctrl + Shift + I` (Windows/Linux) or `Cmd + Option + I` (Mac). This will open the Developer Tools.

2. Go to the 'Network' Tab: Inside Developer Tools, click on the 'Network' tab. This is where you can see all network requests made by the page.

3. Clear Existing Logs: To ensure you're only tracking the form submission, click the 'Clear' button (the circle icon) in the top-left corner of the 'Network' tab.

4. Open the Form Submission Page: Go to the page that contains the form. This ensures that all network traffic generated during the form submission process will be visible in the 'Network' tab.

5. Fill Out the Form: Enter data into the form fields as you normally would.

6. Submit the Form: Click the 'Submit' button to trigger the form submission.

7. Monitor Network Traffic: Watch the 'Network' tab as the form is submitted. You will see new requests appear as the form is processed. Look for a request that relates to the form submission, usually a POST request.

8. Inspect the Request: Click on the form submission request (in most cases it is a 'POST' request). This will display detailed information about the request and response.

9. Analyze the Response: In the 'Response' tab, check for any error messages returned by the server. You can also inspect the 'Headers' tab to check if the request payload (the data you submitted) is correct.

## Common HTTP Error Codes

1. 400 Bad Request: This error indicates that the server cannot process the form submission because the request was malformed. This could be due to invalid form data (e.g., incorrect input types, missing required fields). In the Network tab, check the request payload to ensure all required fields are included and properly formatted.

2. 401 Unauthorized: The server requires authentication, but the request did not include valid credentials. Check if the request is missing an authentication token or login session, and ensure the user is authenticated.

3. 403 Forbidden: This error occurs when the server understands the request but refuses to authorize it. Check if the user has the necessary permissions to submit the form. Inspect headers for any authorization tokens or session IDs.

4. 404 Not Found: The requested URL (such as the form action URL) could not be found on the server. Verify the URL specified in the form action attribute is correct and points to an existing resource on the server.

5. 405 Method Not Allowed: The server does not allow the HTTP method (e.g., POST) for the given resource. Ensure the form's method matches the server's expected method for the action URL (typically POST for submitting forms).

6. 500 Internal Server Error: This error indicates a problem on the server side, usually due to a misconfiguration or bug. In this case, you won't be able to resolve the issue from the client side. Check the server logs for more details.

7. 502 Bad Gateway: This error suggests a problem with the server acting as a gateway or proxy. Check the server's configuration and ensure that all upstream services are functioning properly.

8. 503 Service Unavailable: The server is temporarily unavailable, often due to maintenance or heavy load. Try again later.

9. 504 Gateway Timeout: This error occurs when the server, acting as a gateway, does not receive a timely response from an upstream server. It could indicate issues with network connectivity or a slow backend service.

# Further Debugging Tips

If none of the above steps or errors point to the issue, here are some additional debugging tips:

1. Check the Console for JavaScript Errors: In the 'Console' tab of the developer tools, check for any JavaScript errors or warnings that could be preventing the form submission. Pay attention to uncaught exceptions or network request failures.

2. Inspect the Form Submission Data: Sometimes, the issue lies with the form data itself. Use the 'Network' tab to inspect the form data being submitted in the request payload. Check for missing or malformed fields that may be causing issues.

3. Check for CORS Issues: Cross-Origin Resource Sharing (CORS) issues can prevent form submissions from reaching the server. Check the 'Network' tab to see if any CORS-related errors are returned by the server, such as 'No 'Access-Control-Allow-Origin' header.'

4. Test with Different Data If the form is not submitting, try filling out the form with different data, including edge cases, such as extremely long strings, special characters, or empty fields, to see if the issue is related to specific inputs.

5. Check Server Logs: If the issue persists on the server-side, access the server logs (if available) to look for clues about what went wrong.

6. Network Connection: Sometimes, issues like timeouts or loss of connection between the client and the server can affect form submissions. Check if the network connection is stable.