## **1. Testing Approach**

To ensure the functionality of both the backend (Node.js server) and the frontend (HTML page), I conducted manual testing. The testing process involved:

* **Backend Testing:**
  + Testing the /postmessage endpoint using both the frontend form and the curl command.
  + Verifying file handling (creating posts.txt if it doesn't exist and appending new posts).
  + Testing error handling for missing or invalid input.
* **Frontend Testing:**
  + Testing the form submission to ensure it sends data to the server asynchronously using the fetch API.
  + Verifying that the page updates dynamically without requiring a manual refresh when a new post is added.
  + Ensuring that the existing posts are displayed correctly when the page loads.
  + Testing responsiveness on both desktop and mobile devices.

## **2. Test Cases**

### **Backend Test Cases:**

1. **Valid Post Request (via curl):**

* **Command:**

curl -X POST http://localhost:3000/postmessage -H "Content-Type: application/json" -d '{"topic": "Test", "data": "Hello World"}'

* **Expected Result:** The server should append the post to **posts.txt** with the correct timestamp and return a **201** **Created** status.
* **Actual Result**: The post was successfully appended to **posts.txt**, and the server returned **201 Created**.

2. **Valid Post Request (via Frontend Form):**

* **Input:** { topic: "Frontend Test", data: "This is a test post from the frontend." }
* **Expected Result:** The server should append the post to **posts.txt** and the frontend should update the post list dynamically.
* **Actual Result:** The post was appended, and the frontend updated the post list without a page reload.

**3. Missing Topic or Data:**

* **Input:** { topic: "", data: "Test Data" }
* **Expected Result:** The server should return a **400 Bad Request** status with an error message.
* **Actual Result:** The server returned **400 Bad Request** with the message "Topic and data are required."

**4**. **File Handling:**

* **Scenario:** **posts.txt** does not exist.
* **Expected Result:** The server should create **posts.txt** and append the first post.
* **Actual Result:** The file was created, and the post was appended successfully.

**Frontend Test Cases:**

1. **Form Submission:**

* **Input:** Enter a topic and data in the form and submit.
* **Expected Result:** The form should send the data to the server, and the post list should update without a page reload.
* **Actual Result:** The post list updated dynamically, and the new post appeared in the list.

1. **Display Existing Posts:**

* **Scenario:** Open the page with existing posts in **posts.txt**.
* **Expected Result:** The page should display all existing posts from **posts.txt**.
* **Actual Result:** All posts were displayed correctly when the page loaded.

1. **Error Handling:**

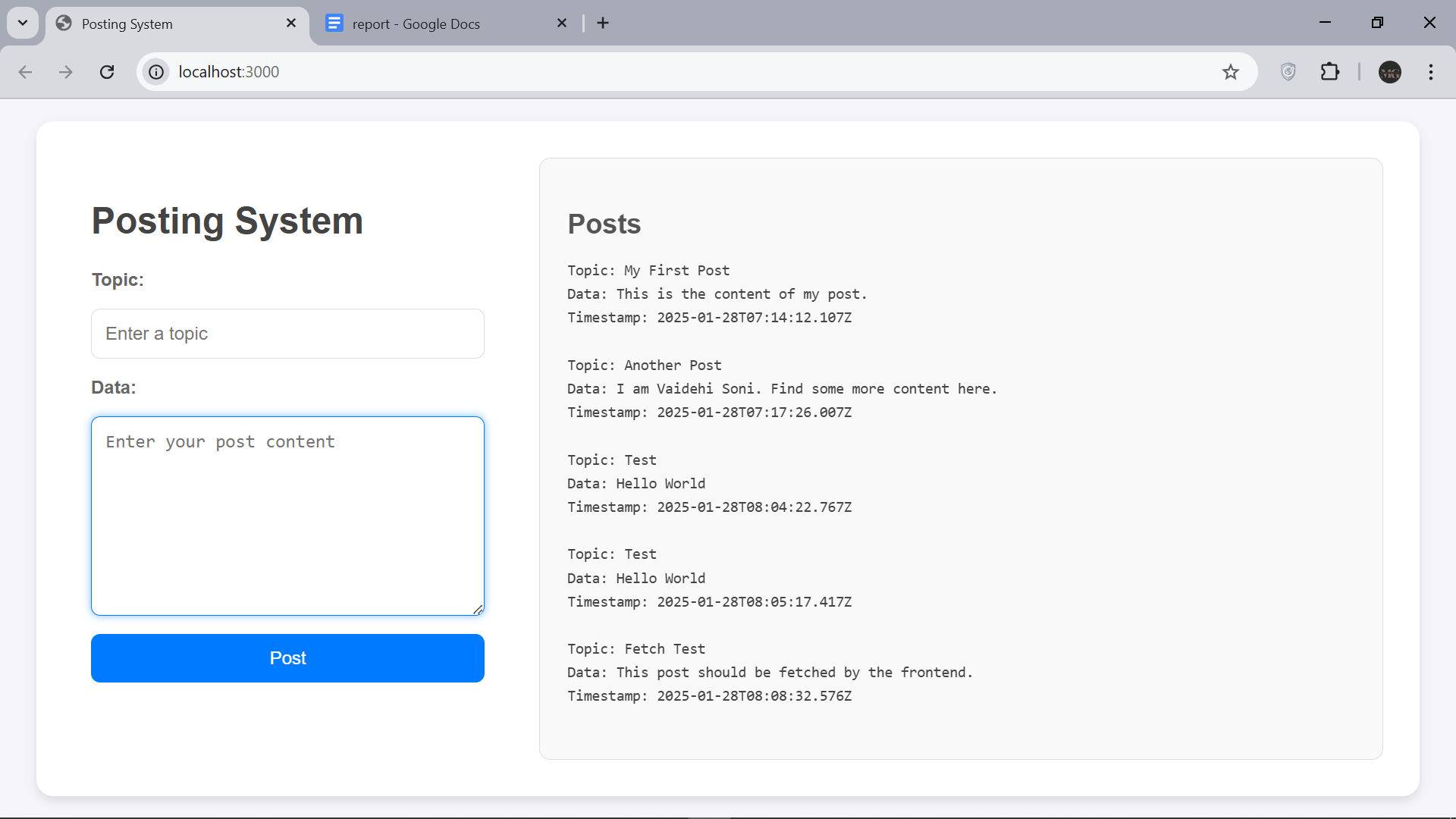
* **Scenario:** Submit the form without entering a topic or data.
* **Expected Result:** The server should return an error, and the frontend should handle it gracefully.
* **Actual Result:** The server returned a **400 Bad Request**, and the frontend did not update the post list.

**5. Responsiveness:**

* **Scenario:** Resize the browser window to simulate mobile and desktop views.
* **Expected Result:** The layout should adjust seamlessly, with the form and posts stacking vertically on mobile.
* **Actual Result:** The layout adjusted correctly, providing a great user experience on both desktop and mobile.

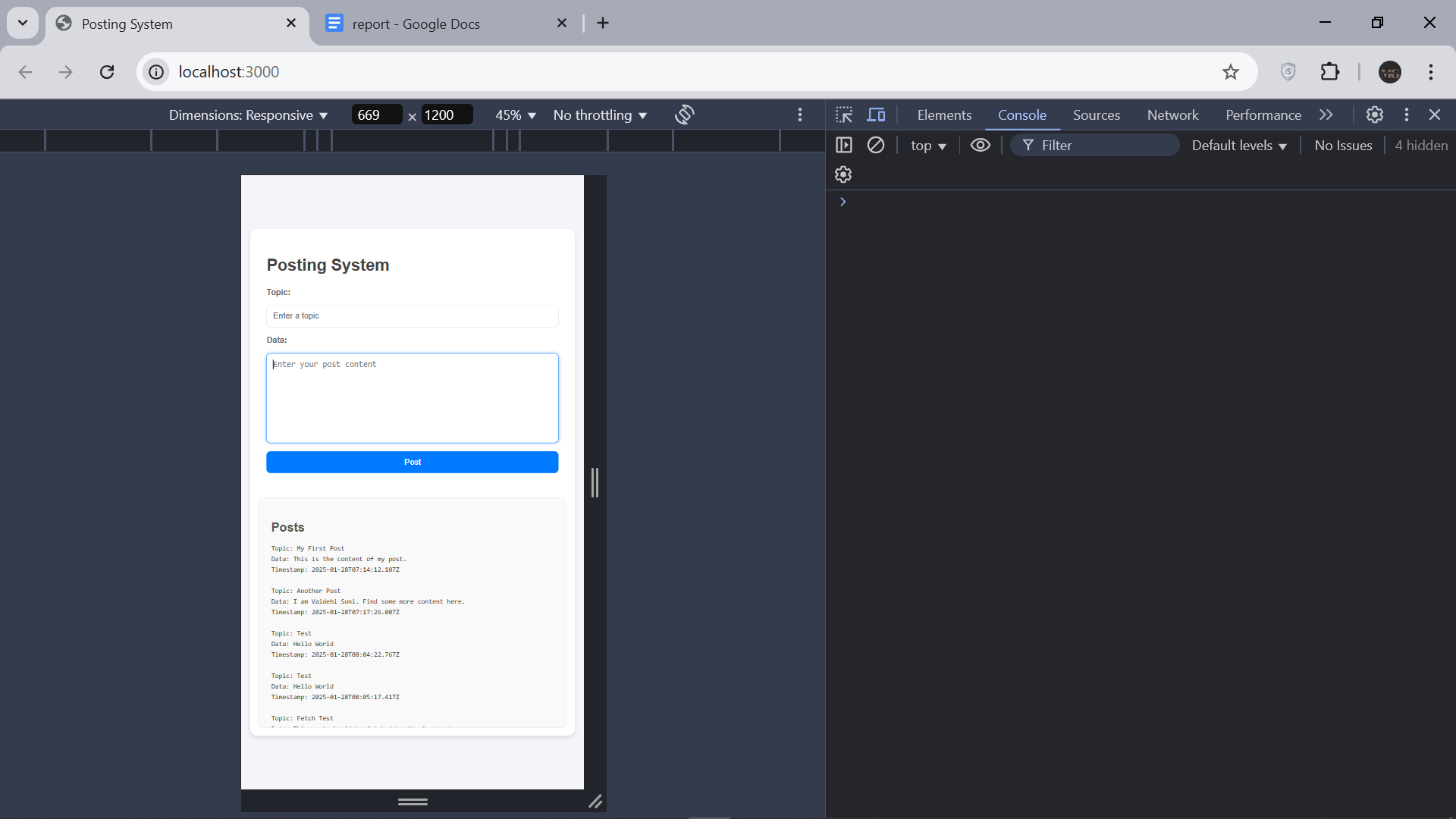
**6. Screenshots**

* **Desktop View:**



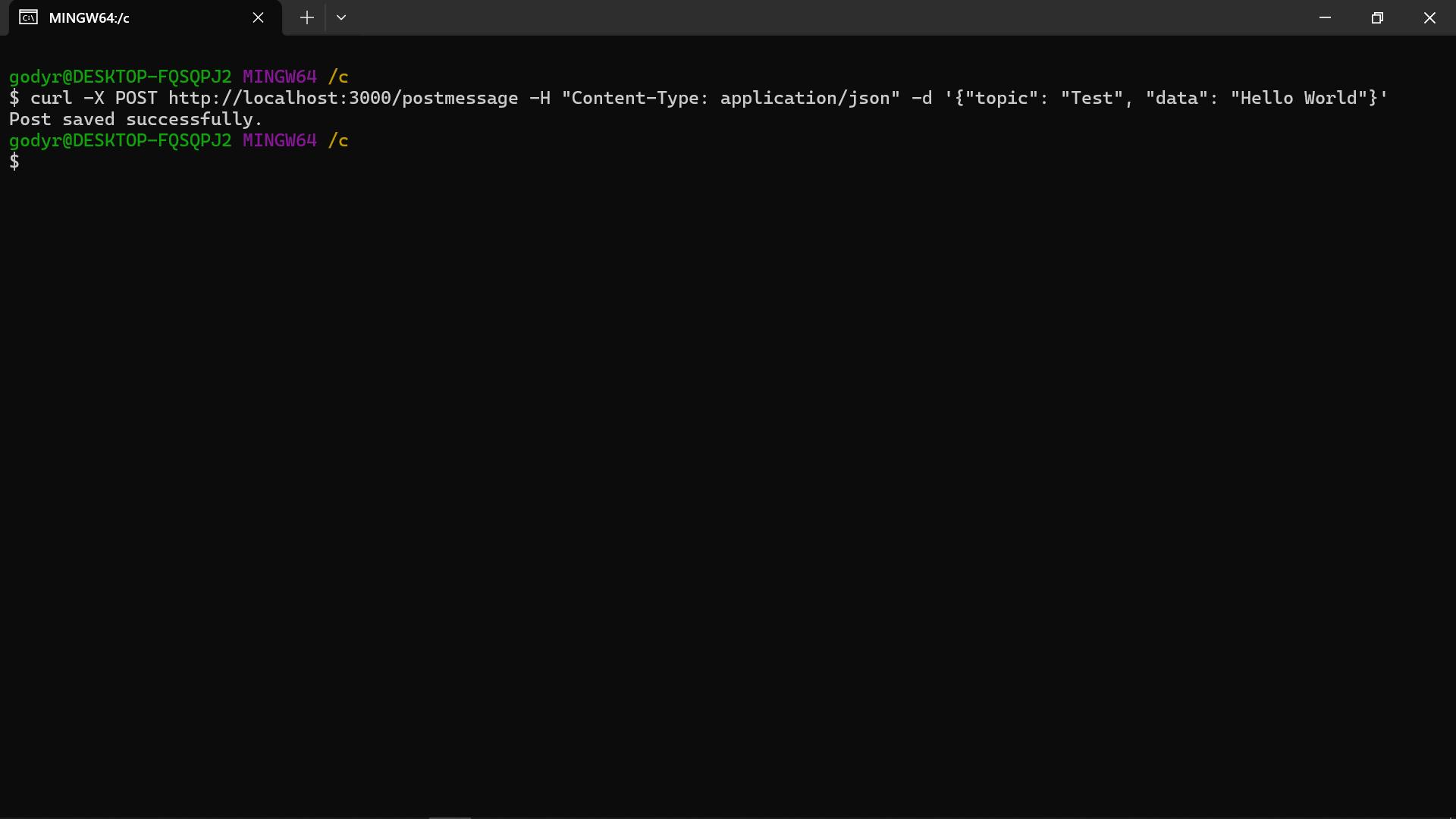
Screenshot of the desktop layout with the form on the left and posts on the right.

* **Mobile View:**

****

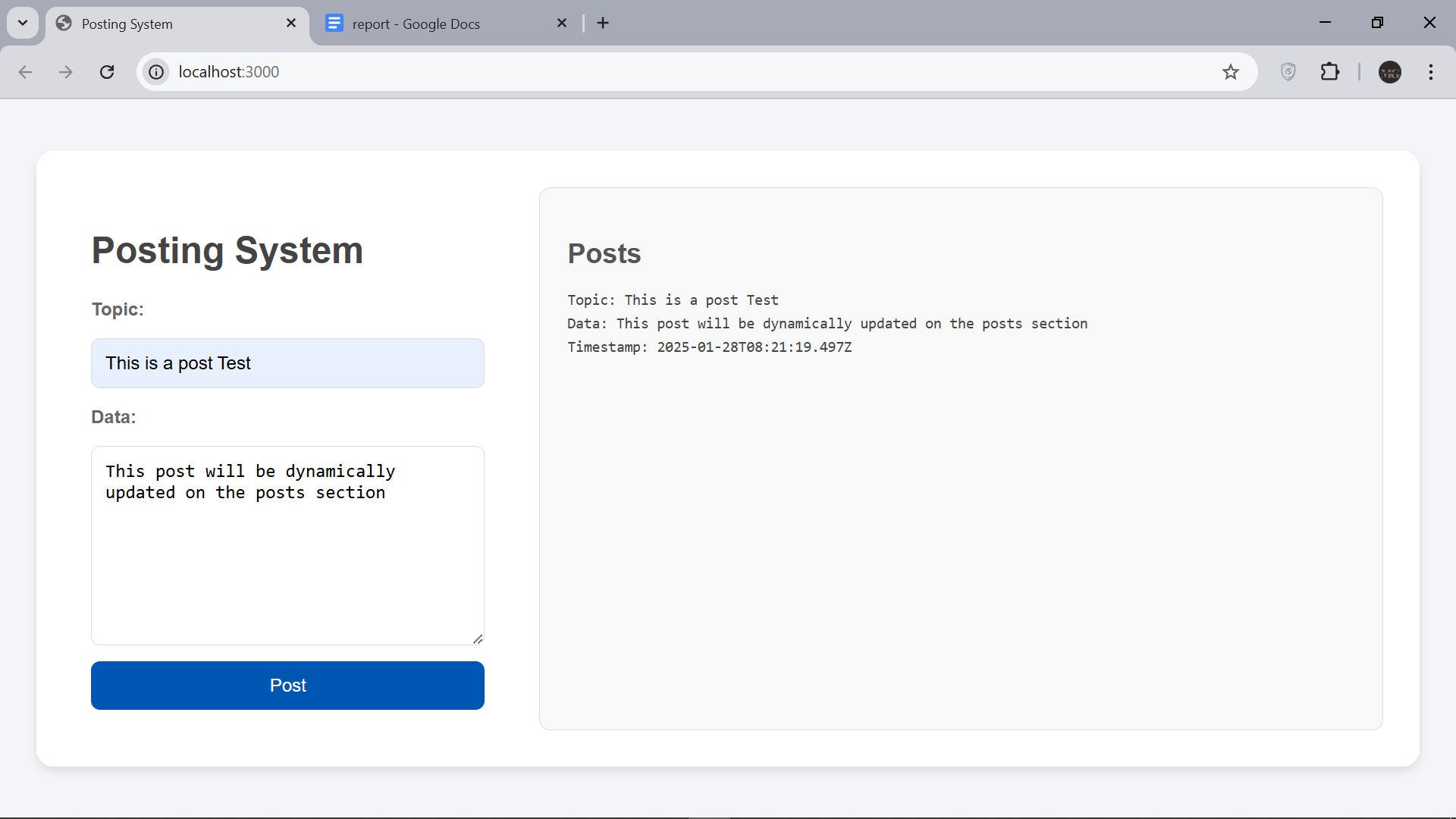
Screenshot of the mobile layout with the form and posts stacked vertically.

* **Backend Success:**



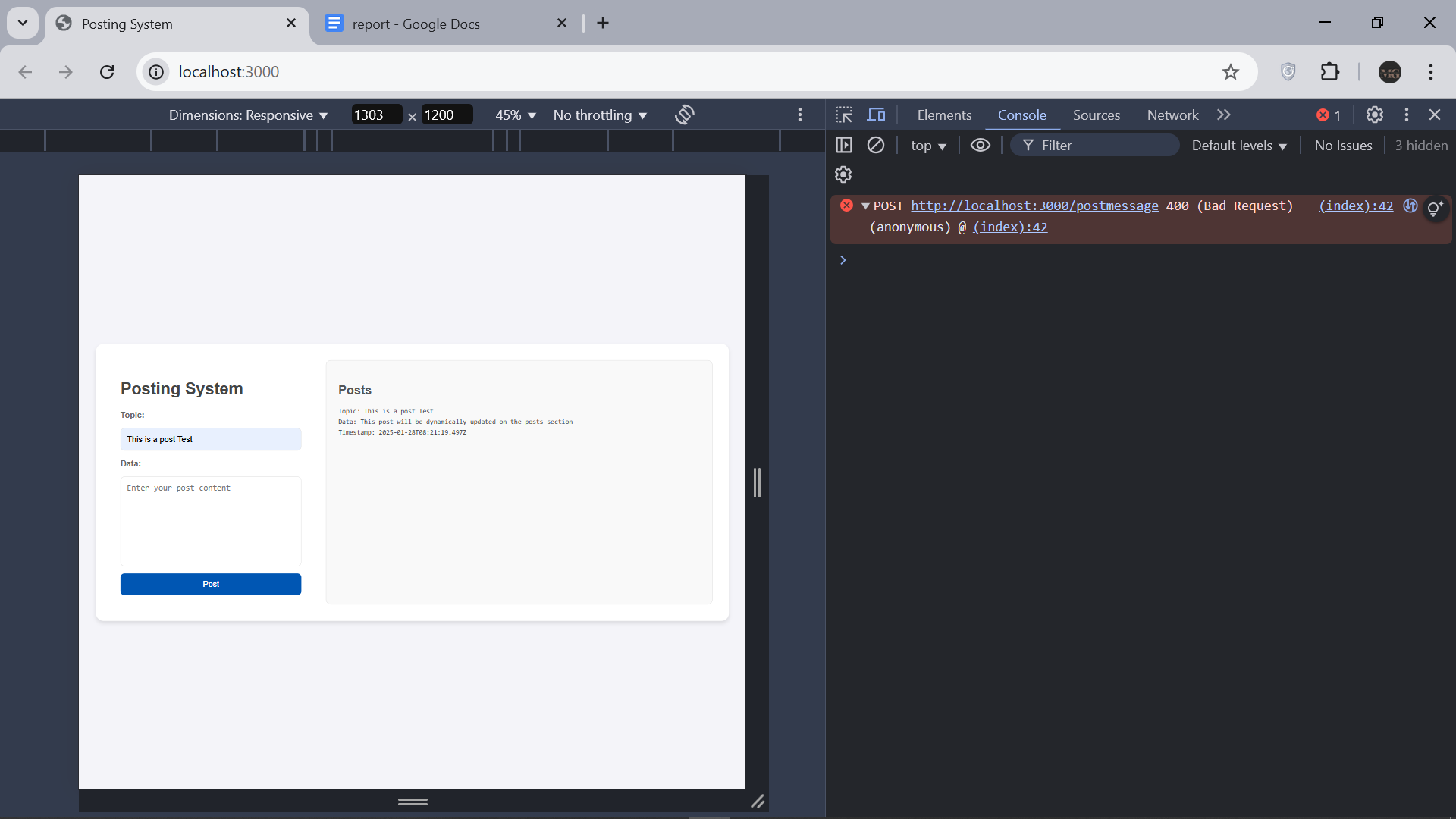
Screenshot of the terminal showing the server running and successfully handling a POST request via curl.

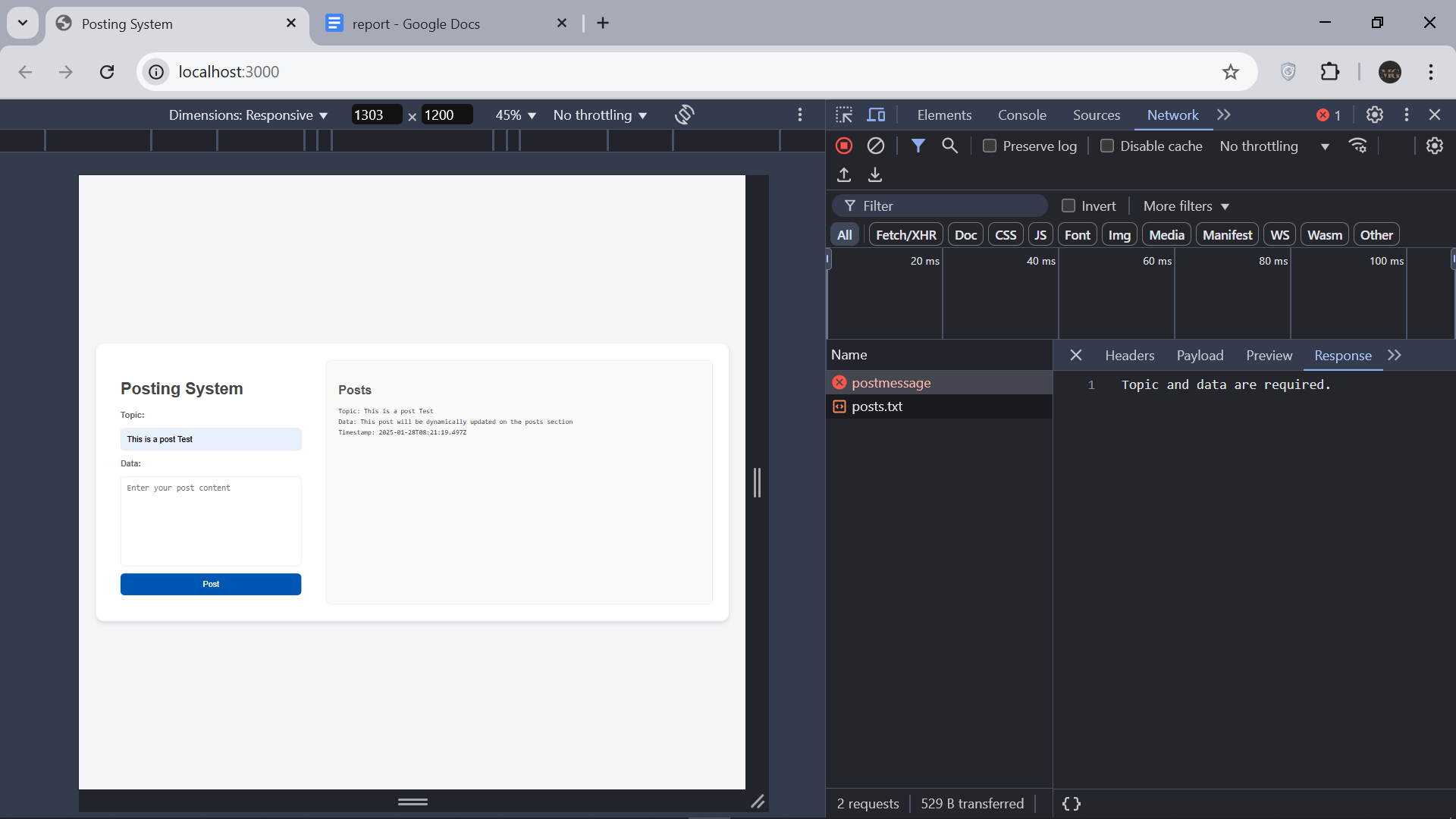
* **Frontend Success:**



Screenshot of the browser showing the form, submitted posts, and dynamically updated post list.

* **Error Handling:**





Screenshots of the browser console showing the error message when submitting an incomplete form.

**Challenges and Solutions**

* **Challenge:** Initially, the frontend did not update dynamically after a new post was added.
* **Solution:** I added a call to **fetchPosts()** after the form submission to refresh the post list.
* **Challenge:** The server did not handle missing **posts.txt** correctly.
* **Solution:** I modified the file handling logic to create **posts.txt** if it doesn't exist before appending the post.
* **Challenge:** Making the layout responsive for both desktop and mobile.
* **Solution:** I used CSS Flexbox and media queries to create a responsive design that adapts to different screen sizes.