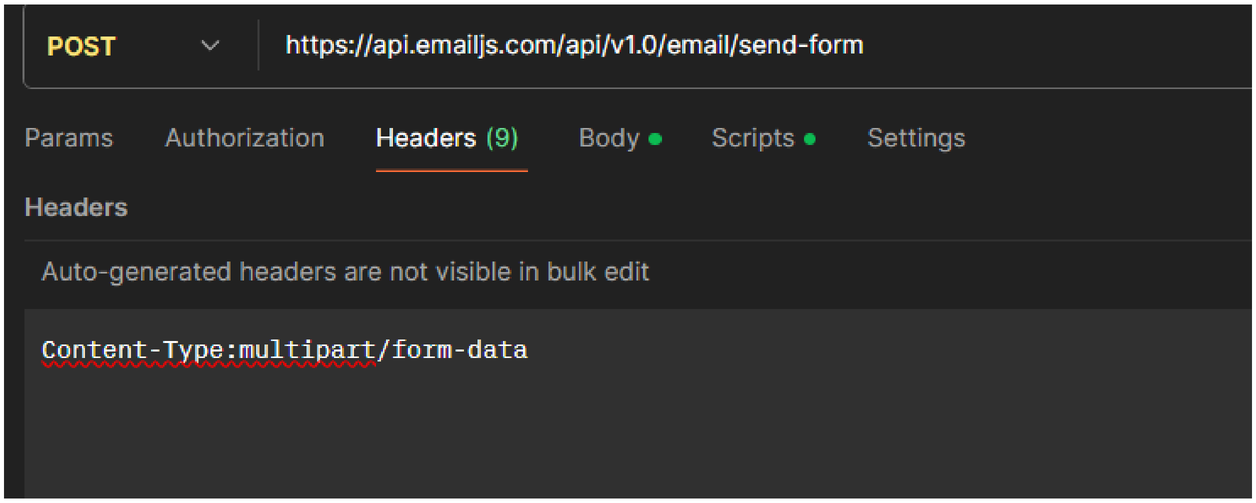
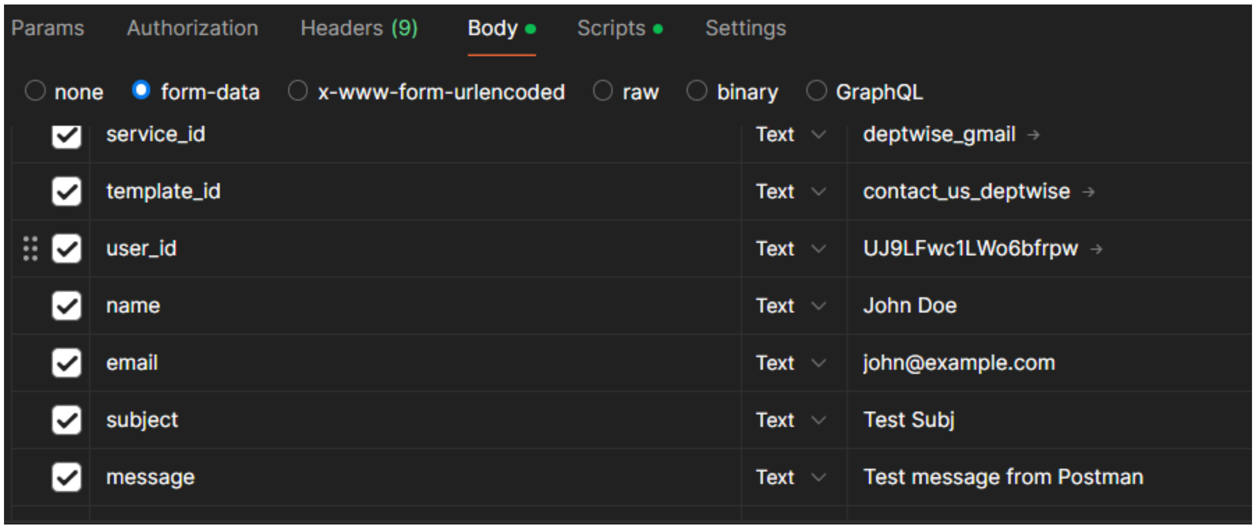

POST https://api.emailjs.com/api/v1.0/email/send-form

```
Content-Type:multipart/form-data
```



( ) none (●) form-data ( ) x-www-form-urlencoded ( ) raw ( ) binary ( ) GraphQL

| | | | |
|---|---|---|---|
| ✓ | service_id | Text | deptwise_gmail |
| ✓ | template_id | Text | contact_us_deptwise |
| ✓ | user_id | Text | UJ9LFwc1LWo6bfrpw |
| ✓ | name | Text | John Doe |
| ✓ | email | Text | john@example.com |
| ✓ | subject | Text | Test Subj |
| ✓ | message | Text | Test message from Postman |

This pre-request script in Postman generates unique test data (email/username with timestamps) to prevent duplicate conflicts, sets default EmailJS configuration values (service ID, template ID, and public key), but your error ("Invalid Public Key") occurs because the placeholder UJ9LFwc1LWo6bfrpw isn't a valid EmailJS key – replace it with your actual public key (starting with "user_") from the EmailJS Dashboard under API Keys, while ensuring your service_id and template_id match your configured EmailJS service and template.

```
// Generate timestamp for unique messages
const timestamp = new Date().getTime();

// Set dynamic values using faker.js (built-in Postman)
pm.environment.set("dynamicEmail", `testuser_${timestamp}@example.com`);
pm.environment.set("dynamicName", `User_${timestamp}`);

// Set default template values
```

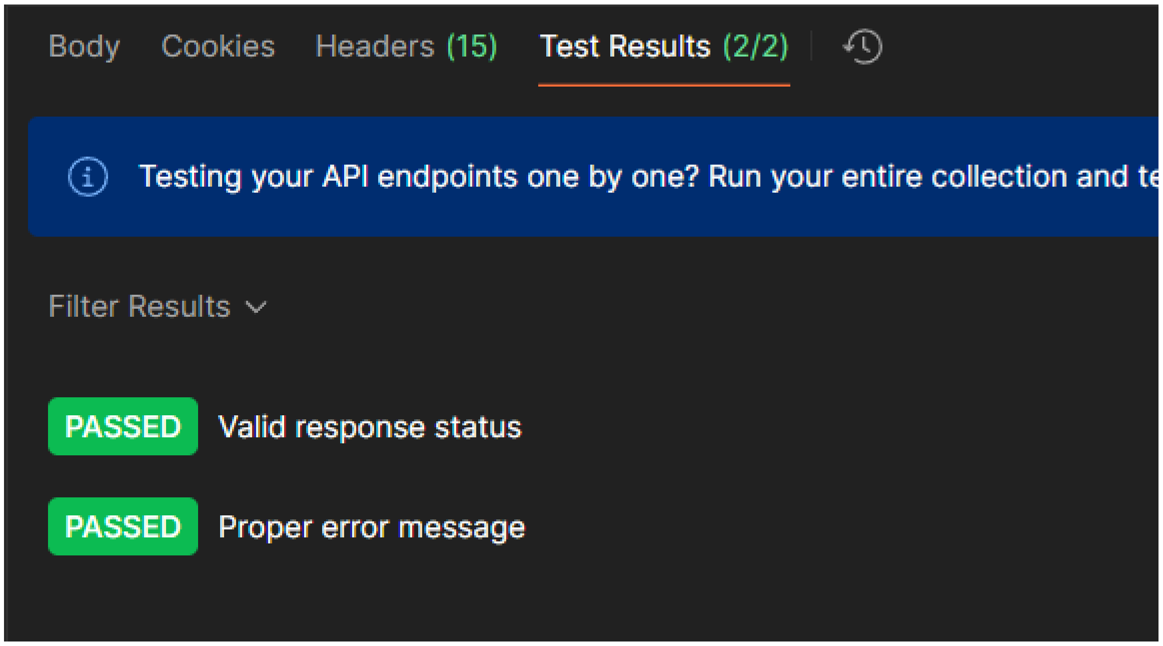
```
pm.environment.set("service_id", "deptwise_gmail");
pm.environment.set("template_id", "contact_us_deptwise");
pm.environment.set("user_id", "UJ9LFwc1LWo6bfrpw");
```

Post request

```
// Check for both success and failure cases
pm.test("Valid response status", () => {
  pm.expect(pm.response.code).to.be.oneOf([200, 400]);
});

pm.test("Proper error message", () => {
  const responseText = pm.response.text();
  if(pm.response.code === 400) {
    pm.expect(responseText).to.include("Public Key");
  } else {
    pm.expect(responseText).to.eql("OK");
  }
});
```

This Postman test script validates both successful (200 OK) and error (400 Bad Request) responses from the EmailJS API: the first test checks if the status code is either 200 (success) or 400 (invalid public key error), while the second test verifies the response text contains "Public Key" for errors (to detect authentication issues) or matches "OK" for successful email submissions, ensuring proper handling of both expected scenarios in your contact form integration.



Testing your API endpoints one by one? Run your entire collection and te

PASSED Valid response status

PASSED Proper error message