Bug Report: core-utils.js Missing Content-Type Header for POST Requests

Date: 2025-11-06

Affected Component: core-utils.js (Specifically the fetchWithGuard function)

Severity: High (Prevents successful API submission of all JSON payloads)

Description:

The fetchWithGuard function, used for all API communication, is failing to correctly set the **Content-Type: application/json** HTTP header when performing POST or PUT requests that contain a JavaScript object body (which is intended to be serialized as JSON).

Without this header, the server-side framework (e.g., Flask/Werkzeug) is unable to automatically parse the request body as JSON. This results in the server failing to find the required JSON fields (job_title_name, date_applied, etc.), rejecting the request with a **400 Bad Request** status, and often returning a non-JSON formatted error body.

Observed Console Output:

```
POST [http://192.168.56.4/api/applications](http://192.168.56.4/api/applications) 400 (BAD REQUEST)
[API] Fatal Error after 3 attempts for [object Object]: Error: Non-JSON API Error (400): [object Object] failed. Response was not JSON.
Application creation failed: Error: Failed to perform... Technical details: Non-JSON API Error (400): [object Object] failed. Response was not JSON.
```

Steps to Reproduce:

1. Attempt to submit the application form via application_create.js.
2. The handleSubmit function calls fetchWithGuard with the applicationData object.
3. The request is sent to /api/applications.
4. The server returns 400 BAD REQUEST due to missing or unparsable fields.

Proposed Solution (Fix for fetchWithGuard):

Inside fetchWithGuard, before the fetch call, logic must be added to check if the body argument is a plain JavaScript object and, if so, ensure the following is applied to fetchOptions:

1. Set the request body: fetchOptions.body = JSON.stringify(body);
2. Set the header: fetchOptions.headers['Content-Type'] = 'application/json';

This is essential for the API server to correctly interpret the payload.