

# Bug Fix Documentation – Note Taking Application

## 1. Project Overview

This project is a simple Flask-based Note Taking Application. The purpose of this application is to allow users to enter notes through a text input field and display all added notes as an unordered list on the same home page.

Initially, the application contained several issues that prevented it from working correctly. The task was to identify these bugs, refactor the code, and make the application functional as per the requirements.

## 2. Application Requirements

The expected functionality of the application is:

- The home page should contain a text field and an “Add Note” button.
- Users should be able to enter a note and submit it.
- All submitted notes should be displayed below the input field as an unordered list.
- The application should work using a single home route.

## 3. Bugs Identified and Fixes Applied

### Bug 1: Missing Form Submission Method

#### Problem Description:

The HTML form did not specify a request method. By default, the browser sends form data using the GET method. However, the Flask backend was not designed to handle GET requests for form submission, which caused the note input to not be processed correctly.

#### Solution Implemented:

The form method was changed to POST so that the data could be securely and correctly sent to the server.

#### Result:

After this fix, the form started sending data properly to the backend.

### Bug 2: Submit Button Type Not Defined

#### Problem Description:

The button inside the form did not explicitly define its type. This can cause inconsistent behavior across browsers and may prevent proper form submission.

#### Solution Implemented:

The button type was explicitly set as “submit” to ensure correct form submission behavior.

#### Result:

The button now works reliably and triggers the form submission as expected.

### **Bug 3: Backend Route Not Handling POST Requests**

**Problem Description:**

The Flask home route was only configured to accept GET requests. Since the form was sending data using POST, the server was ignoring submitted notes.

**Solution Implemented:**

The home route was updated to accept both GET and POST methods. This allowed the application to process user-submitted notes.

**Result:**

The backend now correctly receives and processes form data.

### **Bug 4: Notes Were Not Stored Anywhere**

**Problem Description:**

The application did not have any data structure to store notes. As a result, even if the input was received, the notes could not be displayed.

**Solution Implemented:**

A Python list was created to store the notes during runtime. Each submitted note is appended to this list.

**Result:**

All added notes are now stored and displayed dynamically on the web page.

### **Bug 5: Empty Note Submission Allowed**

**Problem Description:**

The application allowed users to submit empty notes. This resulted in blank entries being displayed in the notes list.

**Solution Implemented:**

Input validation was added to ensure that empty or whitespace-only notes are not accepted.

**Result:**

The application now accepts only valid notes and prevents blank submissions.

## **4. Outcome**

After fixing all identified issues, the Note Taking Application is now fully functional. Users can successfully add notes using the home page form and view all submitted notes as an unordered list. The application now meets all the requirements specified in the task.