**Defect Report**

Testing Technique 2 Total defects: **15**

Testing Technique 3 Total defects **WITHOUT** additional exploratory testing: **4**

Testing Technique 3 Total defects **WITH** additional exploratory testing: **5**

Usability & Exploratory Testing: **6**

**TOTAL DEFECTS FOUND: 30**

**Validation and Error Messaging Issues:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Defect ID | Title | Description | Root Cause | Technique Used |
| **DF01** | Lack of validation **and** uninformative error message for CSV file with correct headers but no data | When a CSV file with correct headers but missing data in required columns is uploaded, the system lacks **validation** to specify the required data and stop further execution. Instead, **a generic error message**—“Failed to load CSV, no numeric data to plot”—is shown. The **validation** should specify that columns ‘assessment1’ and ‘assessment2’ need valid numerical data and halt execution if these requirements are not met. | The system lacks specific **validation** checks for files with correct headers but missing required data, resulting in continued execution and a **generic error message** rather than an instructive prompt to provide the necessary data. | Testing technique 2 |
| **DF02** | Duplicate Student ID columns not detected | Uploading a CSV file with duplicate student ID columns incorrectly loads the file without detecting duplicate values. Instead of properly **validating** the column, the system allows the file to load. The **expected** error message should be: "The uploaded file is invalid. The column studentID must contain unique values." | The system lacks **validation** for unique values in the student ID column, allowing duplicates to be processed without detection. | Testing Technique 2 |
| **DF03** | Lack of validation **and** uninformative error message for CSV file with incorrect headers and no data | Uploading a CSV file with incorrect headers and no data does not trigger specific **validation** to stop the program from further execution. Instead, a **generic error message**—“Failed to load CSV file. ‘assessment1’”—is shown. The **validation** should specify that the file must have the correct headers and contain data in the required columns, and it should halt execution if these conditions are not met. | The system lacks **validation** checks for files with incorrect headers and missing required data. This results in continued execution and an **uninformative** error message instead of a clear instruction to provide the correct headers and data. | Testing technique 2 |
| **DF04** | Incorrect handling of extra unspecified header | Uploading a CSV file with an extra unspecified header loads the file without error. The system should include **validation** to check that **only** the specified columns are present, and stop execution if any extra headers are detected, preventing the file from being opened. | The system lacks **validation** to detect and handle extra, unspecified headers. This results in the unintended acceptance of files with additional columns, without stopping execution or providing an appropriate error message. | Testing technique 2 |
| **DF05** | Numeric values in string columns not flagged | Uploading a CSV file with numeric values in specific string columns (e.g., W\_or\_I, assessment2\_NS, assessment1\_NS) does not trigger the expected **validation** or **provide an error message. T**he file is loaded successfully where the system should actually **validate** the data types in these designated string columns and stop execution if a mismatch is detected. | Lack of **validation**; the system does not check for data type mismatches in designated string columns or provide an appropriate error message to stop the program from executing. | Testing technique 2 |
| **DF06** | Unspecified strings in specific string columns not flagged | Uploading a CSV file with unspecified string values in columns (e.g., W\_or\_I not containing 'W' or 'I') does not trigger the specified **validation** or an error message, and the file is loaded successfully. The system should **validate** that string columns contain only allowed values and stop execution if invalid values are detected. | Lack of **validation**; the system does not check that string columns contain only specified values and fails to provide an **appropriate error message** to stop the program from executing. | Testing technique 2 |
| **DF07** | Non-numeric data in numeric fields not flagged | Uploading a CSV file with non-numeric data in numeric fields triggers an unexpected error: "Can't multiply sequence by non-int of type float," instead of displaying a specific error message through **validation**. The system should **validate** that numeric fields contain valid numeric data and stop execution if invalid data is found, providing an appropriate error message. | The system attempts operations on invalid data types before **validating** the contents of the columns, resulting in an **unexpected error** rather than a clear validation message. | Testing technique 2 |
| **DF08** | File size limit not enforced | Uploading a large CSV file (12,500 KB) crashes the system instead of triggering a **validation** check for file size exceeding the allowed limits. The system should validate the file size before processing and display an **error message** if the file exceeds the limit. | The system lacks a file size **validation** step before processing, allowing large files to cause a crash rather than providing an appropriate **error message**. | Testing technique 2 |
| **DF09** | Null values in columns not flagged | Uploading a CSV file with only commas and no values in columns loads without triggering a specific **validation** error for missing numeric data. The system should **validate** that columns contain valid numeric data and stop execution if columns are empty or contain null values, displaying an appropriate **error message.** | The system fails to **validate** columns with null or empty values, allowing the file to load without triggering the necessary **error message**. | Testing technique 2 |
| **DF10** | Duplicate headers not flagged | Uploading a CSV file with duplicate headers and valid numeric data results in an **unexpected error,** instead of triggering a **detailed validation message**. The system should **validate** for duplicate headers before processing the data and display an appropriate **error message**, such as: "Duplicate headers detected. Please ensure each column has a unique header." | The system does not **validate** for duplicate headers before processing the file, leading to an **unexpected error** instead of a clear validation message. | Testing technique 2 |
| **DF11** | Unexpected formatting in numeric fields not flagged | Uploading a CSV file with unexpected formatting (e.g., extra spaces) in numeric fields causes an **unexpected error**, rather than triggering the intended **validation**. The system should preprocess the numeric fields to handle formatting issues, such as extra spaces, and display a **clear validation message** if the formatting is incorrect. | The system lacks preprocessing for expected formatting in numeric fields, allowing **unexpected errors** to occur instead of a proper **validation** check. | Testing technique 2 |
| **DF12** | Mixed delimiters in CSV file | Uploading a CSV file with mixed delimiters results in an "Inconsistent delimiters" error, rather than triggering the intended validation message for comma-separated values. The system should validate that the file uses consistent delimiters before parsing the data and provide an appropriate error message if inconsistencies are detected. | The system does not handle or validate inconsistent delimiters before parsing the data, leading to errors instead of a clear validation message. | Testing technique 2 |
| **DF13** | Negative values in numeric fields not flagged | Uploading a CSV file with negative values in numeric fields loads without displaying **expected validation or error messages** for data range (0-100%). The system should **validate** that numeric fields fall within the expected range and display an **appropriate error message** if the values are outside the allowed range. | The system lacks range **validation** for numeric fields, allowing invalid values (such as negative numbers) to be processed without triggering an **error**. | Testing technique 2 |
| **DF14** | Excessively large values in numeric fields not flagged | Uploading a CSV file with large numeric values in fields intended for the 0-100% range loads without triggering a **validation** or an **error message**. The system should validate that numeric fields fall within the expected 0-100% range and display an **appropriate error message** if the values exceed the upper limit. | The system lacks **validation** for upper range limits in numeric fields, allowing values outside the 0-100% range to be processed without triggering an **error**. | Testing technique 2 |
| **DF15** | Handling of N/A values in numeric columns | Uploading a CSV file where all values are "N/A" fails to display the intended **validation or error** message for missing numeric data. | The system does not treat "N/A" values as missing data, allowing the file to load without **validating** the presence of numeric data. | Testing technique 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DF16** | Lack of validation and error message inconsistency for lower out-of-range weight values | Uploading a CSV file with out-of-range weight values (e.g., -1, 101) does not trigger the expected **validation** message. Instead of displaying an **error** like "The inputted weights are invalid. Weights must total 100% (e.g., assessment1: 40.0, assessment2: 60.0)," the system proceeds to display the overall grade calculation. The system should **validate** that the weight values are within the acceptable range and total 100%. | The system lacks providing **validation** **and an appropriate error message for** weight values that fall outside the acceptable range, allowing improper data processing and leading to the display of incorrect results. | Testing Technique 3 (Equivalence Partitioning) |
| **DF17** | Lack of validation and error message inconsistency for higher out-of-range weight values | Uploading a CSV file with out-of-range weight values (e.g., 101, -1) bypasses the expected **validation** check and proceeds to display the overall grade calculation, instead of showing the specific error message: "The inputted weights are invalid. Weights must total 100% (e.g., assessment1: 40.0, assessment2: 60.0)." The system should **validate** that weight values do not exceed the upper or lower range and trigger the appropriate error message if they do. | The system does not **validate** weight values exceeding the acceptable range, allowing invalid input to bypass validation and resulting in incorrect calculations. | Testing Technique 3 (Equivalence Partitioning) |
| **DF18** | Lack of validation and error message inconsistency for empty and null weight values | Uploading a CSV file with empty values ("", "") in the weight fields does not trigger the correct **validation or an appropriate error message.** Instead, it produces an unrelated parsing error: "Failed to load CSV file. Expecting floating-point number but got ""." The system should validate that weight fields are not empty and display a specific **validation** message if they are. | The system lacks a check for empty input values in the weight fields, causing parsing errors instead of providing clear **validation** feedback. | Testing Technique 3 (Equivalence Partitioning) |
| **DF19** | Lack of validation and error message inconsistency for invalid data types in weight fields | Uploading a CSV file with invalid data types (e.g., "Hello, World" instead of numeric values) results in a generic parsing error: “Failed to load CSV file. Expecting floating-point number but got 'hello',” rather than triggering the expected validation message: "The inputted weights are invalid. Values must be numbers between 0 and 100." The system should **validate** the data types in the weight fields before parsing and display a specific validation error for invalid types. | The system does not **validate** data types in weight fields prior to parsing, leading to an **unhelpful error message** when invalid data types are encountered. | Testing Technique 3 (Equivalence Partitioning) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DF20** | Lack of validation and error message inconsistency for foreign decimal notation in weights | Entering foreign decimal notation (e.g., "10,5; 90,0") causes a parsing error instead of displaying the expected error message: "Invalid input format. Use a decimal point (.) for numerical values." The system should **validate** the input format before parsing and **trigger a clear validation message** when non-standard decimal notation is detected. | The system does not **validate** the input format before parsing, leading to **unexpected errors** when non-standard decimal notation is used. | Testing Technique 3 (Additional Exploratory Testing) |
| **DF21** | Lack of validation and error message inconsistency for numeric words in weights | Entering numeric words (e.g., "Fifty-five") in the weight fields leads to a parsing error instead of the expected **validation** message: "The weights must be numerical values." The system should **validate** that the weight fields contain numerical values before processing and display the appropriate **validation** message for non-numeric input. | The system does not **validate** the data type or format for the weight fields before processing, allowing non-numeric input to cause **parsing errors.** | Testing Technique 3 (Additional Exploratory Testing) |
| **DF22** | Lack of validation for non-English numerals in weights | Entering non-English numerals (e.g., "५०, ५०") produces a parsing error instead of the expected message: "The weights must be numerical values between 0 and 100 in English numerals." The system should **validate** that weight fields contain numerals in the correct format (English numerals) and **display the appropriate validation** message for non-English numerals. | The system does not **validate** input language or numeral format, leading to **errors** when non-English numerals are entered. | Testing Technique 3 (Additional Exploratory Testing) |
| **DF23** | Missing weight value entry | Entering only one weight value (e.g., "40,") **does not trigger the expected validation message**: "Both weights must be entered to open CSV file." The system should **validate** that both weight values are provided before processing the CSV file and display the appropriate error message for incomplete input. | The system fails to **validate** that both weight values are entered, allowing incomplete input to be processed. | Testing Technique 3 (Additional Exploratory Testing) |
| **DF24** | Special character input in weights | Entering special characters (e.g., "@, #") in the weight fields leads to a **parsing error instead of the expected validation message:** "The inputted weights are invalid. Values must be numbers between 0 and 100." The system should **validate** that only numeric values are entered in the weight fields and display the appropriate **validation** message for invalid characters. | The system does not **validate** character types in the weight fields, resulting in an **uninformative error** when special characters are used. | Testing Technique 3 (Additional Exploratory Testing) |

**UI Inconsistencies/errors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Defect ID | Title | Description | Root Cause | Technique Used |
| **DF25** | Handling of NAN values | NAN values are displayed as "NaN", which is not user-friendly. They should be replaced with more accessible terms like "No number" or "N/A". | The application displays NAN values without considering user experience or accessibility. | Usability & Exploratory Testing |
| **DF26** | Non-rounded grades | Grades are displayed with excessive decimal places, like ’62.8799999999999995’, making them visually unappealing and confusing for users. Rounding should be applied. | The system does not round grades properly, leading to inconsistent visual presentation. | Usability & Exploratory Testing |
| **DF27** | No plot adjustments for invalid values | When users input invalid values, plots do not adjust to reflect this, leading to confusion and a poor user experience. | The system fails to dynamically update plots in response to invalid input. | Usability & Exploratory Testing |
| **DF28** | Inconsistent weight adjustments | When users enter values from right to left, they expect the weights to adjust in the same way as when entered left to right, but this is not the case. | Weight adjustment logic is not properly implemented to handle different input directions. | Usability & Exploratory Testing |
| **DF29** | Unresponsive design on smaller devices | The application does not adjust properly when the window size is reduced, causing elements to be cut off on smaller screens. | The app's responsive design is not effectively implemented for smaller devices. | Usability & Exploratory Testing |
| **DF30** | Ambiguous error messages | Error messages are unclear and do not provide sufficient information for users to understand the problem or how to resolve it. | Error handling messages lack clarity and actionable instructions. | Usability & Exploratory Testing |