CSV Object Importer Task

Your task is to create a small PHP powered tool which can take a CSV file as an input, parse the columns and rows into an object, sort the objects, ensure the Transaction Code is valid and then return the objects in a table format.

Please use the CSV file attached to the email.

The object: BankTransaction

Attributes:

- Date (DateTime)
- Transaction Code (String)
- Customer Number (Int)
- Reference (String)
- Amount (Currency) saved in cents

Please include:

- Use Object Oriented PHP for your solution. There is no need to save anything to a database.
- The sort should be by the Date and Time.
- You should ensure the figures from the CSV are parsed as the correct type above
- **The Transaction Code can be verified as correct by using the algorithm on the second page of this document. The table should display if the transaction code is valid or not.**

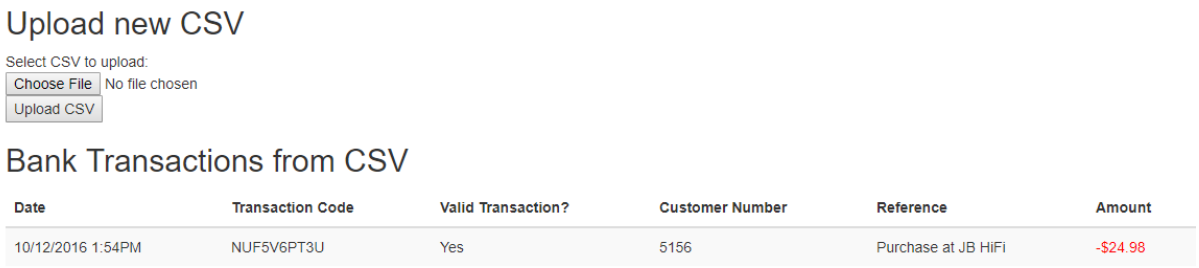
Things to note:

- You can assume the CSV will always be in the correct format
- Your code will need to decide if it's a debit or credit per the amount

Submission requirements:

- Provide a small explanation outlining reasons why you selected your approach and why you decided against any other solutions.
- You must use PHP 7+ for your implementation
- Ensure your code is appropriately documented and formatted using PSR-2 Style Guide (https://www.php-fig.org/psr/psr-2/)

Sample screen output:



Transaction Code Check Character Algorithm

The Transaction Code check character is calculated using the algorithm below. The character weighting and algorithm pseudo code for calculating it is shown below:

```
char[] validChars = {'2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C',
'D', 'E', 'F', 'G', 'H', 'J', 'K','L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',
'U', 'V', 'W', 'X', 'Y', 'Z'}


bool VerifyKey(string key)
{
    if (key.Length != 10)
        return false;
    char checkDigit = GenerateCheckCharacter(key.ToUpper().Substring(0,
9));
    return key[9] == checkDigit;
}

// Implementation of algorithm for check digit.
char GenerateCheckCharacter(string input)
{
    int factor = 2;
    int sum = 0;
    int n = validChars.Length;
    // Starting from the right and working leftwards is easier since
    // the initial "factor" will always be "2"
    for (int i = input.Length - 1; i >= 0; i--)
    {
        int codePoint = Array.IndexOf(validChars, input[i]);
        int addend = factor * codePoint;
        // Alternate the "factor" that each "codePoint" is multiplied by
        factor = (factor == 2) ? 1 : 2;
        // Sum the digits of the "addend" as expressed in base "n"
        addend = (addend / n) + (addend % n);
        sum += addend;
    }
    // Calculate the number that must be added to the "sum"
    // to make it divisible by "n"
    int remainder = sum % n;
    int checkCodePoint = (n - remainder) % n;
    return validChars[checkCodePoint];
}
```