# CMPT473 E2 report

# Specification of the Program Under Test

For this exercise, we decided to test a CLI application written in Javascript that takes a CSV file as input and converts it to produce a JSON output.

## CSV specification:

Files with .csv extension contain records of data with comma separated values. Each line in a CSV file is a new record from the set of records contained in the file.

- Each record is located on a separate line, delimited by a line break (CRLF). For example:
  - aaa,bbb,ccc CRLF
  - zzz,yyy,xxx CRLF
- The last record in the file may or may not have an ending line break. For example:
  - aaa,bbb,ccc CRLF
  - zzz,yyy,xxx
- There may be an optional header line appearing as the first line of the file with the same format as normal record lines. This header will contain names corresponding to the fields in the file and should contain the same number of fields as the records in the rest of the file. For example:
  - field_name,field_name,field_name CRLF
  - aaa,bbb,ccc CRLF
  - zzz,yyy,xxx CRLF
- Within the header and each record, there may be one or more fields, separated by commas. Each line should contain the same number of fields throughout the file. Spaces are considered part of a field and should not be ignored. The last field in the record must not be followed by a comma. For example:
  - aaa,bbb,ccc
- Each field may or may not be enclosed in double quotes. If fields are not enclosed with double quotes, then double quotes may not appear inside the fields. For example:
  - “aaa”,“bbb”,“ccc” CRLF
  - zzz,yyy,xxx
- Fields containing line breaks (CRLF), double quotes, and commas should be enclosed in double-quotes. For example:
  - “aaa”,“b CRLF
  - bb”,“ccc” CRLF
  - zzz,yyy,xxx
- If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be escaped by preceding it with another double quote. For example:
  - “aaa”,“b""bb”,“ccc”

However, the delimiter is not limited to comma only and can be semicolon, tab or spaces as well.

## CSV file format:

file = [header CRLF] record *(CRLF record) [CRLF]
header = name *(COMMA name)
record = field *(COMMA field)
name = field
field = (escaped / non-escaped)
escaped = DQUOTE *(TEXTDATA / COMMA / CR / LF / 2DQUOTE) DQUOTE
non-escaped = *TEXTDATA
COMMA = %x2C
CR = %x0D ;as per section 6.1 of RFC 2234 [2]
DQUOTE = %x22 ;as per section 6.1 of RFC 2234 [2]
LF = %x0A ;as per section 6.1 of RFC 2234 [2]
CRLF = CR LF ;as per section 6.1 of RFC 2234 [2]
TEXTDATA = %x20-21 / %x23-2B / %x2D-7E

## JSON specification:

JSON is a text format for the serialization of structured data. It is derived from the object literals of JavaScript.

- JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays).
- A string is a sequence of zero or more Unicode characters.
- An object is an unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, boolean, null, object, or array.
- An array is an ordered sequence of zero or more values.
- The terms "object" and "array" come from the conventions of JavaScript.
- The three literal names (true, false and null) MUST be lowercase. No other literal names are allowed.

## JSON file format:

<Json> = <Object> || <Array>
<Object> = '{' '}' || '{' <Members> '}' || '{' <Members>, <Members> '}'

<Members> = <name: value>
<Array> = '[' ']' || '[' <Element> ']' || '[' <Element>, <Element> ']'
<Element> = <value>
<value> = string || number || boolean || null || <Array> || <Object>

Reference:
https://en.wikipedia.org/wiki/Comma-separated_values
https://en.wikipedia.org/wiki/JSON
https://www.json.org/json-en.html

## Parameters:

CSV2JSON allows for disk and stdin input and output but for this assignment we decided to test only disk input and output. The csv2json program also contains options for dynamically typed, separator, tab separator and help. We decided to exclude help as it is out of scope for this assignment.

## Input File:

Input files will be OS acceptable or unacceptable csv or txt files depending on the test. The input file will be tested with different separators, i.e. comma (,) , semicolon (;), tabs (  ), space ( ).

## Output File:

Output  files will be OS acceptable or unacceptable json or txt files depending on the test. Output files will be created in the TestOutput/Files folder and compared with the corresponding Expected output files from the TestData folder.

## Testing specifications:

We used JavaScript and Jest framework for testing our CSV2JSON program with 10 tests labeled from 1-10 corresponding to their configurations. The expected output files from TestData are compared with the output files produced by the program in TestOutput/Files, with the input files from TestData folder as well. And after comparison a message for all the tests whether they passed or not is printed in the terminal.
To run:
Install CSV2JSON program - `npm install --save csv2json`
Install Jest - `npm install --save-dev jest`
Command to run our test - npm test

# Input Space Partitioning

| Characteristics | B1 | B2 |
|---|---|---|
| Input files is csv file | TRUE | FALSE |
| Output file is json file | TRUE | FALSE |
| Input file name provided | TRUE | FALSE |
| Output file name provided | TRUE | FALSE |
| Input file name accepted by OS | TRUE | FALSE |
| Output file name accepted by OS | TRUE | FALSE |
| Input file has header | TRUE | FALSE |
| Input file data entry number Zero | TRUE | FALSE |
| Input file contains value with leading whitespace | TRUE | FALSE |
| Input file contains value with whitespace at ending | TRUE | FALSE |
| Input file contains value with double quotes | TRUE | FALSE |
| Input file contains value with one double quote | TRUE | FALSE |
| Input file contains value with line break | TRUE | FALSE |
| dynamic typing value provided | TRUE | FALSE |
| dynamic typing provided value | TRUE | FALSE |
| separator value provided | TRUE | FALSE |
| separator value provided is the same as actual separator in input file | TRUE | FALSE |

# Constraints

Knowing input file type means the input file name is provided

(Input_File_CSV = TRUE || Input_File_CSV = FALSE) => Input_File_Provided = TRUE

Knowing input file name accepted by OS or not means the input file name is provided

(Input_File_Name_Accepted_By_OS = TRUE || Input_File_Name_Accepted_By_OS = FALSE) => Input_File_Provided = TRUE

Any of the characteristics for input file fields are TRUE or FALSE means the input file must be provided.

(Input_File_Entry_Number = 0 || Input_File_Entry_Number > 0) => Input_File_Provided = TRUE
(Input_File_Header = TRUE || Input_File_Header = FALSE) => Input_File_Provided = TRUE
(Input_File_Contains_Start_White_Spaces = TRUE ||
Input_File_Contains_Start_White_Spaces = FALSE) => Input_File_Provided = TRUE

(Input_File_Contain_End_White_Spaces = TRUE || Input_File_Contain_End_White_Spaces = FALSE) => Input_File_Provided = TRUE
(Input_File_Contain_Double_Quotes = TRUE || Input_File_Contain_Double_Quotes = FALSE) => Input_File_Provided = TRUE
(Input_File_Contain_One_Double_Quote = TRUE || Input_File_Contain_One_Double_Quote = FALSE) => Input_File_Provided = TRUE
(Input_File_Contain_Line_Break = TRUE || Input_File_Contain_Line_Break = FALSE) => Input_File_Provided = TRUE

Input file name not specified means input file is not provided, so all other characteristics for input file are not valid

Input_File_Provided = FALSE => (Input_File_CSV = FALSE && Input_File_Name_Valid = FALSE && Input_File_Name_Accepted_By_OS = FALSE && Input_File_Entry_Number = 0
Input_File_Header = FALSE && Input_File_Contains_Start_White_Spaces = FALSE &&
Input_File_Contain_End_White_Spaces = FALSE && Input_File_Contain_Double_Quotes = FALSE && Input_File_Contain_One_Double_Quote = FALSE &&
Input_File_Contain_Line_Break = FALSE)

Input file with 0 entry means input file does not have any other characteristics

Input_File_Entry_Number = 0 => (Separator_Provided = FALSE &&
Dynamic_Typing_Provided = FALSE &&
Input_File_Contains_Start_White_Spaces = FALSE &&
Input_File_Contain_End_White_Spaces = FALSE &&
Input_File_Contain_Double_Quotes = FALSE &&
Input_File_Contain_One_Double_Quote = FALSE &&
Input_File_Contain_Line_Break = FALSE)

Knowing output file type means the output file name is provided

(Output_File_CSV = TRUE || Output_File_CSV = FALSE) => Output_File_Provided = TRUE

Knowing output file name valid or not means the output file name is provided

(Output_File_Name_Valid = TRUE || Output_File_Name_Valid = FALSE) => Output_File_Provided = TRUE

Knowing output file name contains special characters or not means the output file name is provided

(Output_File_Name_Accepted_By_OS = TRUE || Output_File_Name_Accepted_By_OS = FALSE) => Output_File_Provided = TRUE

dynamic typing provided value is TRUE or FALSE mean dynamic typing is provided

(Dynamic_Typing = TRUE || Dynamic_Typing = FALSE) => Dynamic_Typing_Provided = TRUE

Separator value provided is the same as actual separator in input file or not means separator is provided

(Separator_Value_Same = TRUE || Separator_Value_Same = FALSE) => Separator_Provided = TRUE

## Combinatorial Test Generation using ACTS

See ACT_input.txt, ACT_output.txt and TCAS.xml

# Evaluation

From the ACTS output, we got

# Degree of interaction coverage: 2
# Number of parameters: 17
# Maximum number of values per parameter: 2
# Number of configurations: 10

We generated 10 test cases from 17 parameters. All test cases passed successfully. Without pairwise testing, we will need to generate 2^17 test cases, which would have been very time consuming to test. But there are tradeoffs between the coverage and the test efforts. We wrote less test cases, but the test coverage is not very good. For example, we have parameters like Input_File_Entry_Number, the value can be 0 or 5. However, all the tests were produced using Input_File_Entry_Number=5, Input_File_Entry_Number=0 have never been tested. Also Input_File_Provided and Output_File_Provided were always true for all 10 of our test cases, and did not test for a case where input or output file may not be provided.

# Reflection

Understanding the requirements is not hard, but the process of input space is not easy, we need to consider all the reasonable ways of partitioning. Creating the test cases is not easy too because we need to cooperate on different test cases. The process of producing required components to generate test cases could have been more efficient if we worked more closely with the program in our initial stages to familiarize ourselves with the nature of the program, the tasks it performs, and other nuances that are revealed through continuous use of the program. Besides, we should have started early on this project so that we had more time to revise it.

# TEST CASES

# ACTS Test Suite Generation: Thu Feb 17 10:32:40 PST 2022
#  "(don't care)" represents don't care value
# Degree of interaction coverage: 2
# Number of parameters: 17
# Maximum number of values per parameter: 2
# Number of configurations: 10
----------Test Cases----------

Configuration #1:

1 = Input_File_CSV=TRUE
2 = Output_File_JSON=TRUE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=FALSE
7 = Input_File_Header=FALSE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=FALSE
10 = Input_File_Contain_End_White_Spaces=FALSE
11 = Input_File_Contain_Double_Quotes=FALSE
12 = Input_File_Contain_One_Double_Quote=FALSE
13 = Input_File_Contain_Line_Break=FALSE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=FALSE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=FALSE

---

Configuration #2:

1 = Input_File_CSV=TRUE
2 = Output_File_JSON=FALSE

3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=TRUE
6 = Output_File_Name_Accepted_By_OS=TRUE
7 = Input_File_Header=TRUE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=TRUE
10 = Input_File_Contain_End_White_Spaces=TRUE
11 = Input_File_Contain_Double_Quotes=TRUE
12 = Input_File_Contain_One_Double_Quote=TRUE
13 = Input_File_Contain_Line_Break=TRUE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=TRUE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=TRUE

---

Configuration #3:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=TRUE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=TRUE
6 = Output_File_Name_Accepted_By_OS=FALSE
7 = Input_File_Header=TRUE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=FALSE
10 = Input_File_Contain_End_White_Spaces=TRUE
11 = Input_File_Contain_Double_Quotes=FALSE
12 = Input_File_Contain_One_Double_Quote=TRUE
13 = Input_File_Contain_Line_Break=FALSE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=TRUE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=FALSE

---

Configuration #4:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=FALSE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=TRUE

7 = Input_File_Header=FALSE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=TRUE
10 = Input_File_Contain_End_White_Spaces=FALSE
11 = Input_File_Contain_Double_Quotes=TRUE
12 = Input_File_Contain_One_Double_Quote=FALSE
13 = Input_File_Contain_Line_Break=TRUE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=FALSE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=TRUE

---

Configuration #5:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=TRUE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=TRUE
7 = Input_File_Header=TRUE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=FALSE
10 = Input_File_Contain_End_White_Spaces=TRUE
11 = Input_File_Contain_Double_Quotes=TRUE
12 = Input_File_Contain_One_Double_Quote=FALSE
13 = Input_File_Contain_Line_Break=FALSE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=TRUE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=TRUE

---

Configuration #6:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=FALSE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=TRUE
6 = Output_File_Name_Accepted_By_OS=FALSE
7 = Input_File_Header=FALSE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=TRUE
10 = Input_File_Contain_End_White_Spaces=FALSE

11 = Input_File_Contain_Double_Quotes=FALSE
12 = Input_File_Contain_One_Double_Quote=TRUE
13 = Input_File_Contain_Line_Break=TRUE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=FALSE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=FALSE

---

Configuration #7:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=TRUE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=FALSE
7 = Input_File_Header=FALSE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=TRUE
10 = Input_File_Contain_End_White_Spaces=TRUE
11 = Input_File_Contain_Double_Quotes=TRUE
12 = Input_File_Contain_One_Double_Quote=TRUE
13 = Input_File_Contain_Line_Break=FALSE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=TRUE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=TRUE

---

Configuration #8:

1 = Input_File_CSV=TRUE
2 = Output_File_JSON=FALSE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=TRUE
6 = Output_File_Name_Accepted_By_OS=TRUE
7 = Input_File_Header=TRUE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=FALSE
10 = Input_File_Contain_End_White_Spaces=FALSE
11 = Input_File_Contain_Double_Quotes=FALSE
12 = Input_File_Contain_One_Double_Quote=FALSE
13 = Input_File_Contain_Line_Break=TRUE
14 = Separator_Provided=TRUE

15 = Separator_Value_Same=FALSE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=FALSE

---

Configuration #9:

1 = Input_File_CSV=TRUE
2 = Output_File_JSON=TRUE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=FALSE
7 = Input_File_Header=TRUE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=TRUE
10 = Input_File_Contain_End_White_Spaces=FALSE
11 = Input_File_Contain_Double_Quotes=FALSE
12 = Input_File_Contain_One_Double_Quote=FALSE
13 = Input_File_Contain_Line_Break=TRUE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=TRUE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=TRUE

---

Configuration #10:

1 = Input_File_CSV=FALSE
2 = Output_File_JSON=FALSE
3 = Input_File_Provided=TRUE
4 = Output_File_Provided=TRUE
5 = Input_File_Name_Accepted_By_OS=FALSE
6 = Output_File_Name_Accepted_By_OS=TRUE
7 = Input_File_Header=FALSE
8 = Input_File_Entry_Number=5
9 = Input_File_Contains_Start_White_Spaces=FALSE
10 = Input_File_Contain_End_White_Spaces=TRUE
11 = Input_File_Contain_Double_Quotes=TRUE
12 = Input_File_Contain_One_Double_Quote=TRUE
13 = Input_File_Contain_Line_Break=FALSE
14 = Separator_Provided=TRUE
15 = Separator_Value_Same=FALSE
16 = Dynamic_Typing_Provided=TRUE
17 = Dynamic_Typing=FALSE

---