SE 211: Specification and Design II

29 January 2020

SRS for Assignment #1

Mohamed Mohamed

Table of Contents

Introduction

1.1 Purpose:

The purpose of this documentation is specify the requirements and provide some elements of the CSV manager test application and its library. Overall, the program will go over the test application and library. Here are the roles for both articles of software that are discussed within this document.

1. Main Application: The human user's user interface in utilizing the CSV library's various functions.
2. The CSV library: The core of the main application's functionality. Called to perform tasks within the main application.

1.2 Intended Audience:

This document can be read by any user, developer, debugger, and project managers to gain a sense of familiarity behind the system's architecture and its specifications. The following details what how certain roles can be gain a sense of familiarity with the system.

User: Can understand the product features to ensure a sense of full usability and familiar with the test application and the purpose behind the CSV's library's implementation.

Developer role: Can understand the product on a more architectural level, they can understand the design philosophy behind the system to ensure that any modifications to the system do not appear to clash with other implementations already present within the test application and the CSV library.

1.4 Project Scope:

The test application provides the human user the ability to modify, read, and write to CSV files through a Python interpreter. Its general purpose is to utilize a CSV library to execute various functionalities that are determined through the human user's input within the test application.

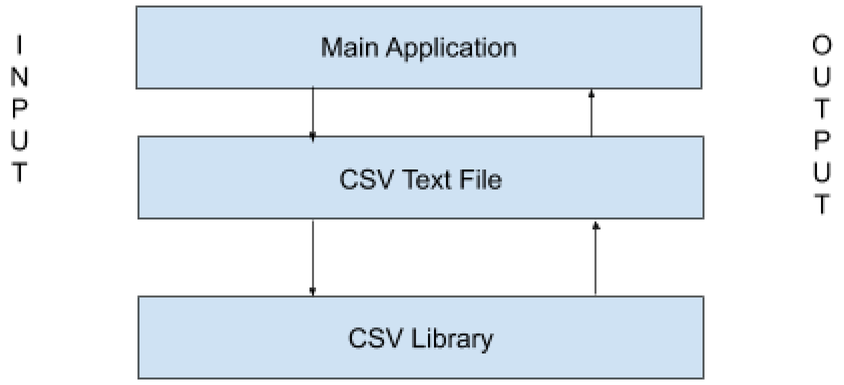
Overall Description:

2.1 Product Perspective:

The main application and by extension, the CSV library, are intended to be simple in utilizing. All that is required to run the program's functions is the CSV library, the main application, and also a CSV formatted file (specifically one with the "txt" file extension). The software is compatible whether the user does or does not have an internet connection. As such, the program's major components and their relationships can be broken down into a single diagram.

The three components within the software are the main application, the CSV Text File, and the CSV library.

- Main Application: Serves as the caller when a user inputs a command within the Python interpreter, requests a file to be analyzed. Serves as the human user's source for output once the program executes the task.
- The CSV text file: The object being modified. The main application both searches for the CSV text file to change or even output on the interface as a form of conformation that the task the human user selected was executed successfully.
- The CSV library: The library serves as the core source of functionality for the main application. It takes in the CSV file passed by the main application and perform a task. After the task has been executed, it takes in the CSV text file and push the output (depending on the task) out to the main application's console.



2.2 Product Features

This program contains the following major features:

- **Multi-OS support:** As long as the user's computer can run Python code and support text files, the main application and library can be used.
- **Changing separators:** The main application can take a file, and modify the default separator of values (the comma) into a pipe ("|") to help with visibility.
- **Read a file:** The main application can read a text file's contents into the console through user input.
- **Read and replace a value**: The main application can read a file, and replace any contents within the file and can change the value through user input.
- **Append into a file:** The main application can add more content into a preexisting file.

2.3 User Classes and Characteristics

The user requires little familiarity to use the main CSV application. As long as the human user has general experience with utilizing Python scripts, they can access all the functionality of the program.

Physical Actors:

- **Human user:** The user is the entity that interact through the main application's interface to perform various tasks on CSV text files.

System Actors:

- **CSV library:** The CSV library's functions will be called depending on the task the user selects. Ex: The user selects the read command, the CSV library's "read" function will be called in the main application.
- **CSV main application:** Serves as the human user's user interface.

2.4 Operating Environment

Any machine that can run native Python code and has compatibility with text files with the file extension name "txt" can run the main application, and thus, has compatibility with the CSV library.

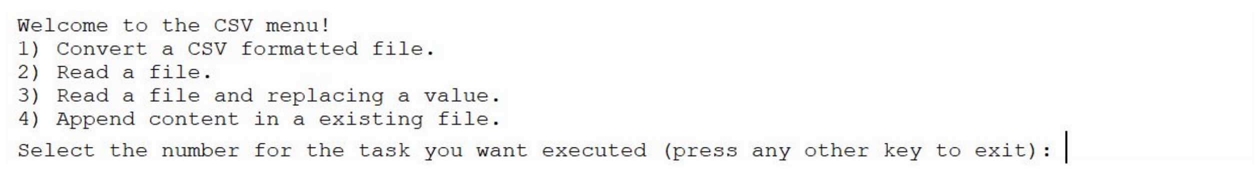
2.5 Design and Implementation Constraints

The pieces of software (the main application and CSV library) are written in Python. As a result, the machine trying to run the program must have a Python interpreter installed. Considering the software can be used without an internet connection, the machine's processor power must be sufficient to run the program. Another constraint is that a machine must be able to run the latest version of Python, in this case: Python 3.7 due to the kind of recursive function calling within main application not being supported in other versions of Python.

2.6 User Interfaces

The user interface of the main application holds the principles of simplicity and being direct. The user interface attempts separate various prompts to improve readability, due to the limited user interface elements that can be implemented through a Python script. Due to the fact that Python interpreters use a CLI (command line interpreter), the user interface does not require any scaling or special styling.

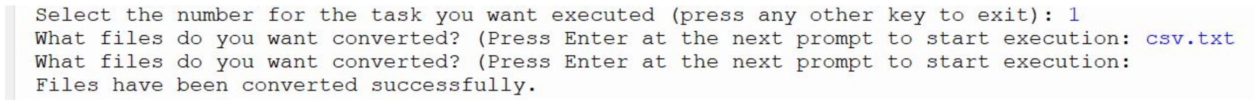
(UI 1 - Main Menu)



```
Welcome to the CSV menu!
1) Convert a CSV formatted file.
2) Read a file.
3) Read a file and replacing a value.
4) Append content in a existing file.
Select the number for the task you want executed (press any other key to exit):
```

The system's CLI as soon as the main application boots up (Main Menu).

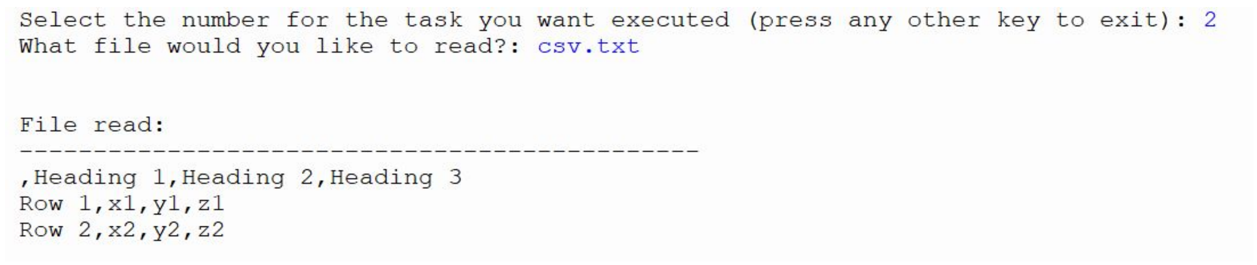
(UI 2 - Change Separators)



```
Select the number for the task you want executed (press any other key to exit): 1
What files do you want converted? (Press Enter at the next prompt to start execution: csv.txt
What files do you want converted? (Press Enter at the next prompt to start execution:
Files have been converted successfully.
```

CLI of a file having its separators being changed.

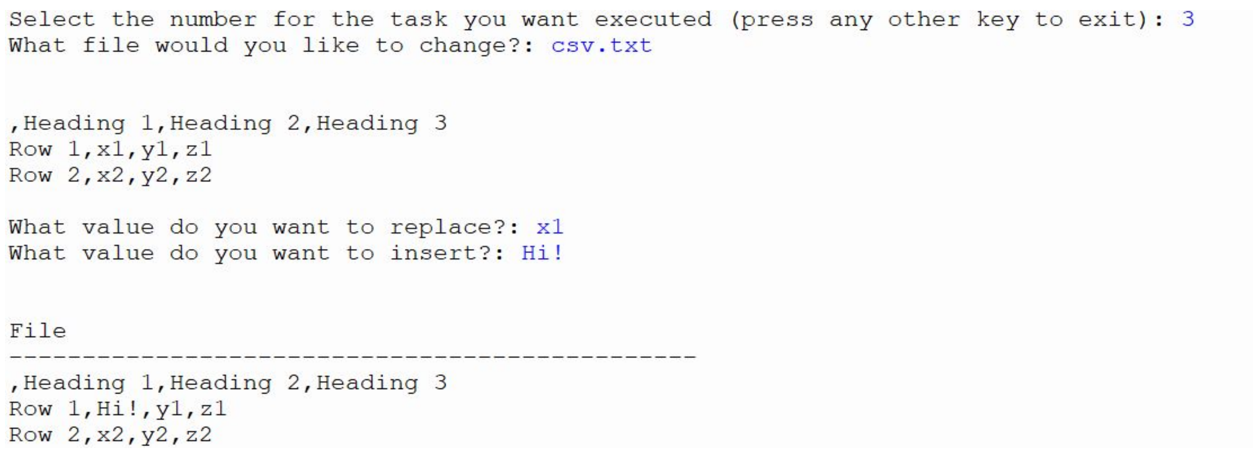
(UI 3 - Read File)



```
Select the number for the task you want executed (press any other key to exit): 2
What file would you like to read?: csv.txt


File read:
----------------------------------------------
,Heading 1,Heading 2,Heading 3
Row 1,x1,y1,z1
Row 2,x2,y2,z2
```

CLI of a file being read.

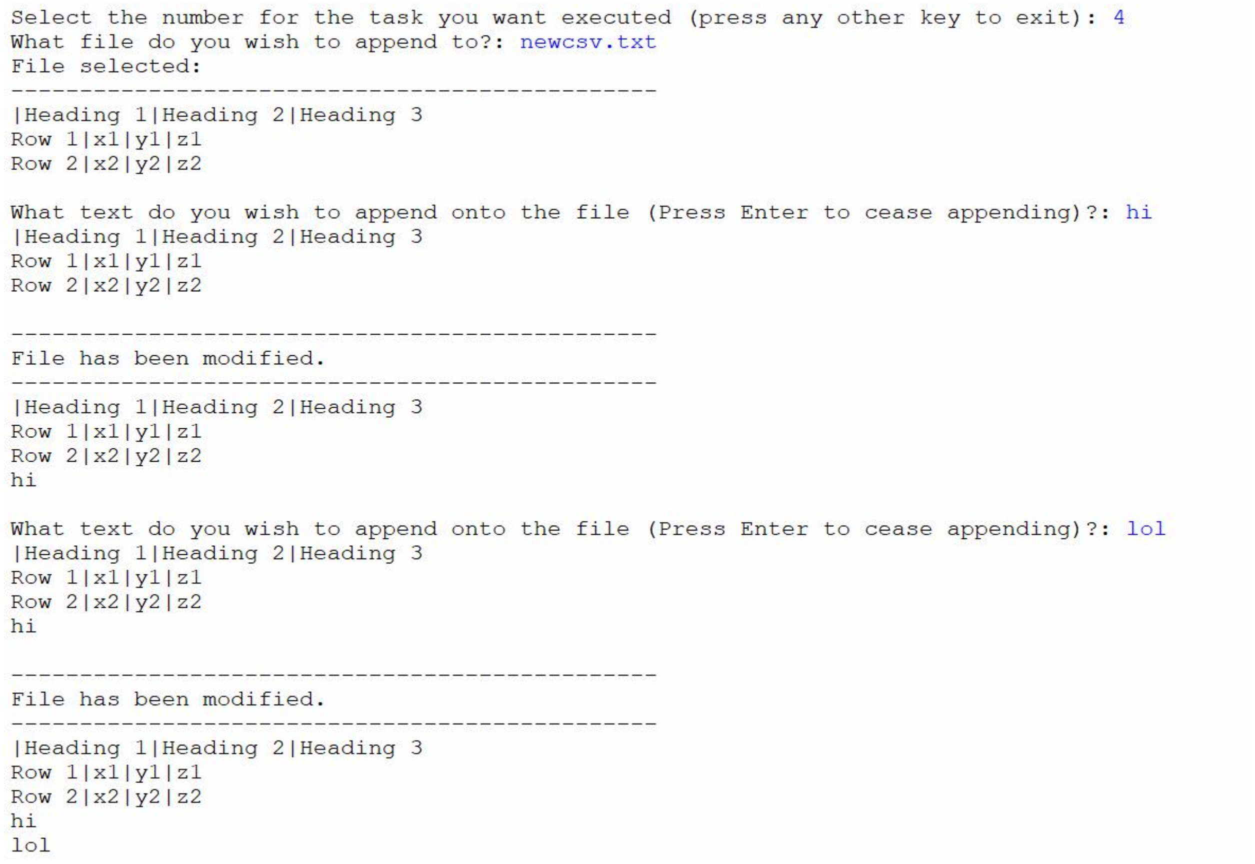
(UI 4 - Replace value in file)



```
Select the number for the task you want executed (press any other key to exit): 3
What file would you like to change?: csv.txt


,Heading 1,Heading 2,Heading 3
Row 1,x1,y1,z1
Row 2,x2,y2,z2

What value do you want to replace?: x1
What value do you want to insert?: Hi!


File
----------------------------------------------
,Heading 1,Heading 2,Heading 3
Row 1,Hi!,y1,z1
Row 2,x2,y2,z2
```

The CLI when the user wants to replace a value with a new one.

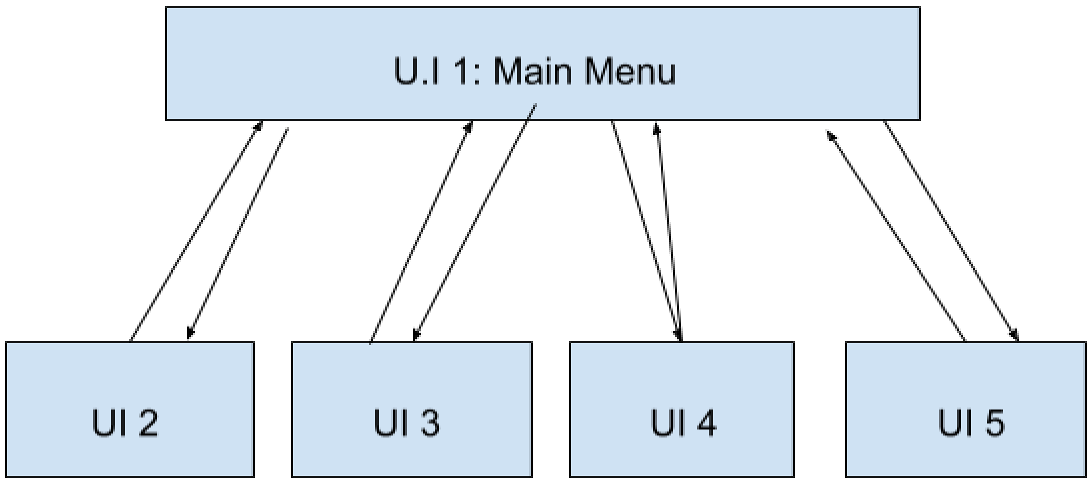
(UI 5 - Appending Values into File)



```
Select the number for the task you want executed (press any other key to exit): 4
What file do you wish to append to?: newcsv.txt
File selected:
-----------------------------------------------
|Heading 1|Heading 2|Heading 3
Row 1|x1|y1|z1
Row 2|x2|y2|z2

What text do you wish to append onto the file (Press Enter to cease appending)?: hi
|Heading 1|Heading 2|Heading 3
Row 1|x1|y1|z1
Row 2|x2|y2|z2

-----------------------------------------------
File has been modified.
-----------------------------------------------
|Heading 1|Heading 2|Heading 3
Row 1|x1|y1|z1
Row 2|x2|y2|z2
hi

What text do you wish to append onto the file (Press Enter to cease appending)?: lol
|Heading 1|Heading 2|Heading 3
Row 1|x1|y1|z1
Row 2|x2|y2|z2
hi

-----------------------------------------------
File has been modified.
-----------------------------------------------
|Heading 1|Heading 2|Heading 3
Row 1|x1|y1|z1
Row 2|x2|y2|z2
hi
lol
```

CLI once the user selects the append option and starts to add input.

(UI Flow Chart)



2.7 Assumptions and Dependencies

The main CSV application was designed with the concept that will be run on a local machine with the basic necessities (Python interpreter and CSV text files). The main application and library assumes that all of the user's CSV files are present in the same folder as the library and itself.

Specific Requirements:

3.1 Functional Requirements:

ID: SEPDIV

TITLE: Changing Separators

PRIORITY: HIGH

DESCRIPTION: Once the user starts the main application, they will send to a main menu. If they select the prompt to changing separators, they will be prompted to give the name of the file modified. Depending if the file exists, the program will state to the user that the task has been successfully or the file does not exist.

ID: REAFIL

TITLE: Read File

PRIORITY: HIGH

DESCRIPTION: Once at the main menu, the user can select the read file option. The user will be prompted to put a filename into the input space. If the file exists, the application will print the file's contents into the CLI. If not, the application will print out an error message.

ID: REPFIL

TITLE: Replace Value in File

PRIORITY: HIGH

DESCRIPTION: At the main menu, the user can select the replace value option. The user will be instructed to input a filename to have it read (see REAFIL). After this, the user will be prompted to give a value to replace and the value you want to insert in the original value's place. The application will print out the modified file's content in the CLI.

ID: APPFIL

TITLE: Append Content In File

PRIORITY: HIGH

DESCRIPTION: At the main menu, the user can select the Append Content option. The application will prompt the user in entering a file to change. Once selected and if the file exists, the user will be prompted to enter any content that they want to add. This will occur in a loop until the user terminate the loop. The program will print out the contents each time the user adds a change into the file.

3.2 Non-Functional Requirements:

ID: QUIT

TITLE: Quitting Out Of Application

PRIORITY: HIGH

DESCRIPTION: At the main menu, if the user input any invalid input, the user terminate the application and exit out of it.

3.3 Performance Requirements:

Performance: The main application's performance depends on the machine's sufficiency and compatibility with processing Python code and modifying text files. Despite that nearly every machine within the market has this compatibility from factory settings, it is still important to note. If the machine does not have compatibility, the application will refuse to execute.

3.4 Safety Requirements:

Consistency: To ensure that the user does not hit accidentally type any errors in the filename, the input is visible to the human user and, if so, the application will give out an error message detailing what occurred.

3.5 Security Requirements:

File must have open permissions to be accessed in the main application.