ASSIGNMENT 1

JEDIT

JEdit is like a text editor written in Java which mainly runs on any operating systems with no specific IDEs of java versions 1.6 or higher. The features gather the functionality of Unix, Windows and Mac editors. It involves unlimited number of copy /paste and undo/redo options. Below can see the demonstration of Jedit.

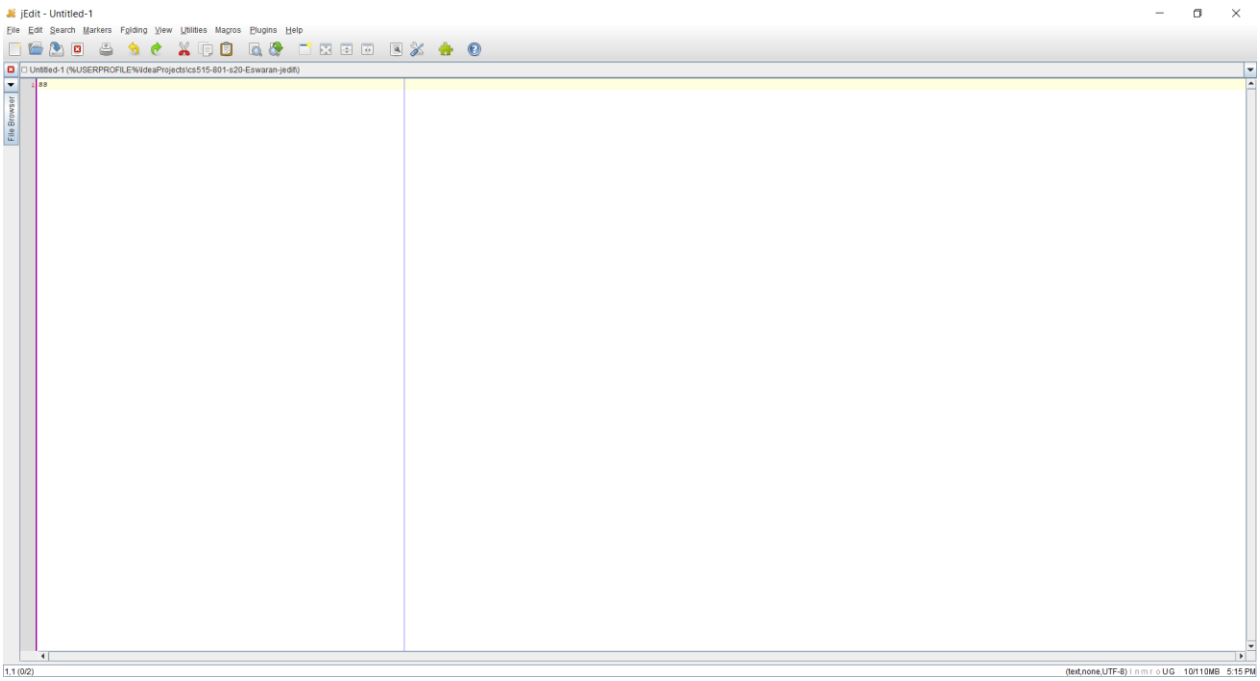

***Figure 1**: Overview of Jedit*

The main importance of Jedit is that it allows us to compile and run Java programs without leaving the editor. The major necessary thing to do is to install the console plug-in that gives access to the command line directly within the editor. From the plugin menu, go to plugin manager for installing the console options. Select the console plug-in and just click on install. The console will start downloading.

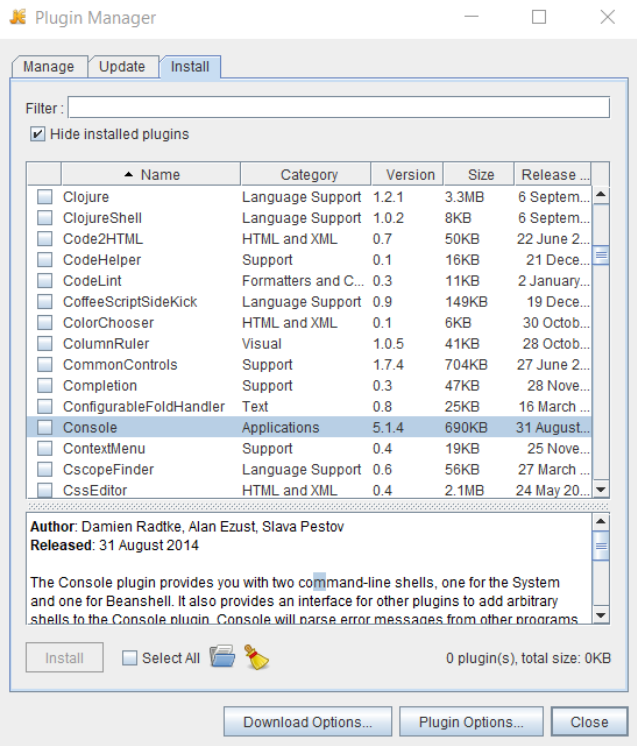

Figure 2: *Plugin Manager*

It should automatically download and install the plug-in for us in the manage tab and verify that it is installed which would be visible in the list. Now we can see the console option in the plugin menu, where console brings in a separate window.

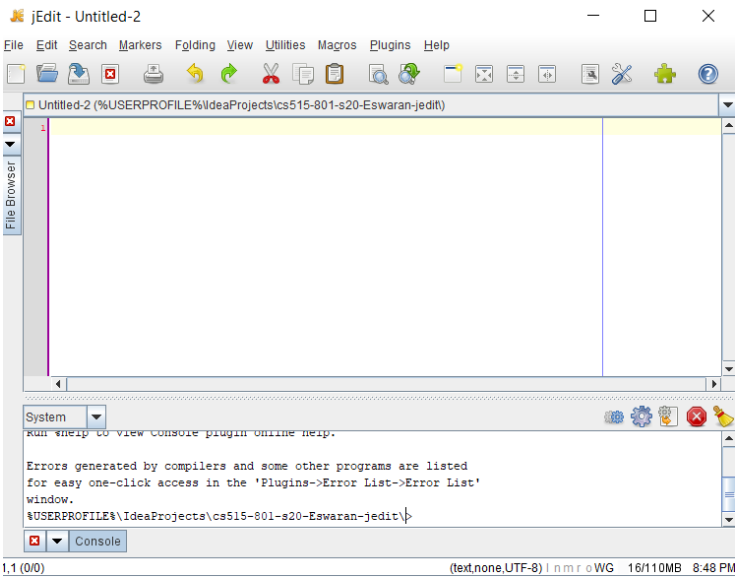

Figure 3: *Console as a plugin in Windows*

Next, it is necessary to view the global options so we can do docking in the program code. Docking is basically creating a view of the console in a single window. There we can select which ever plug-in to be dock. Here, we have selected console to be docked in bottom and file browser to be docked on the right position. Here console acts like a Windows command line where file explorer can be used as a file to be compiled. First, we are planning to run a simple hello world in java to show it process.

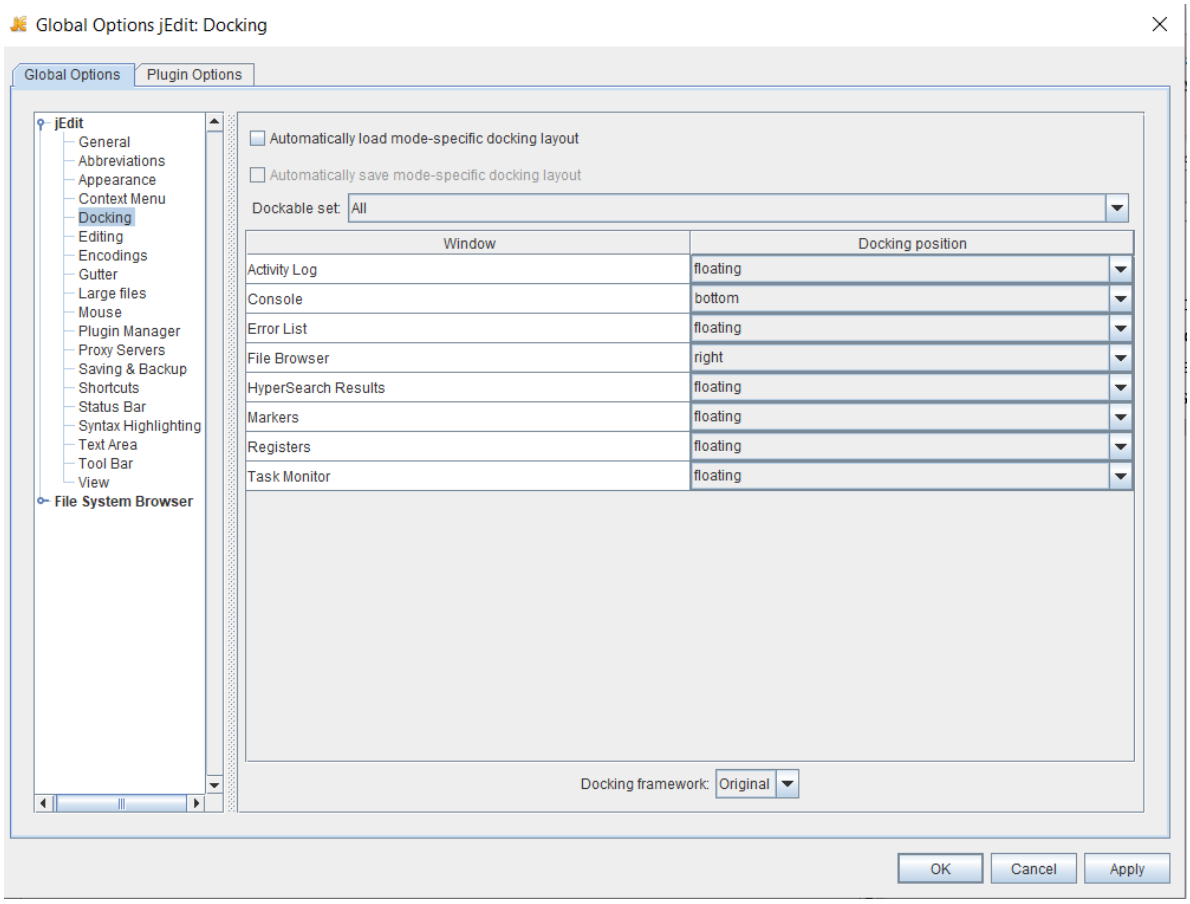

Figure 4: *Docking of Console plugin with File Browser*

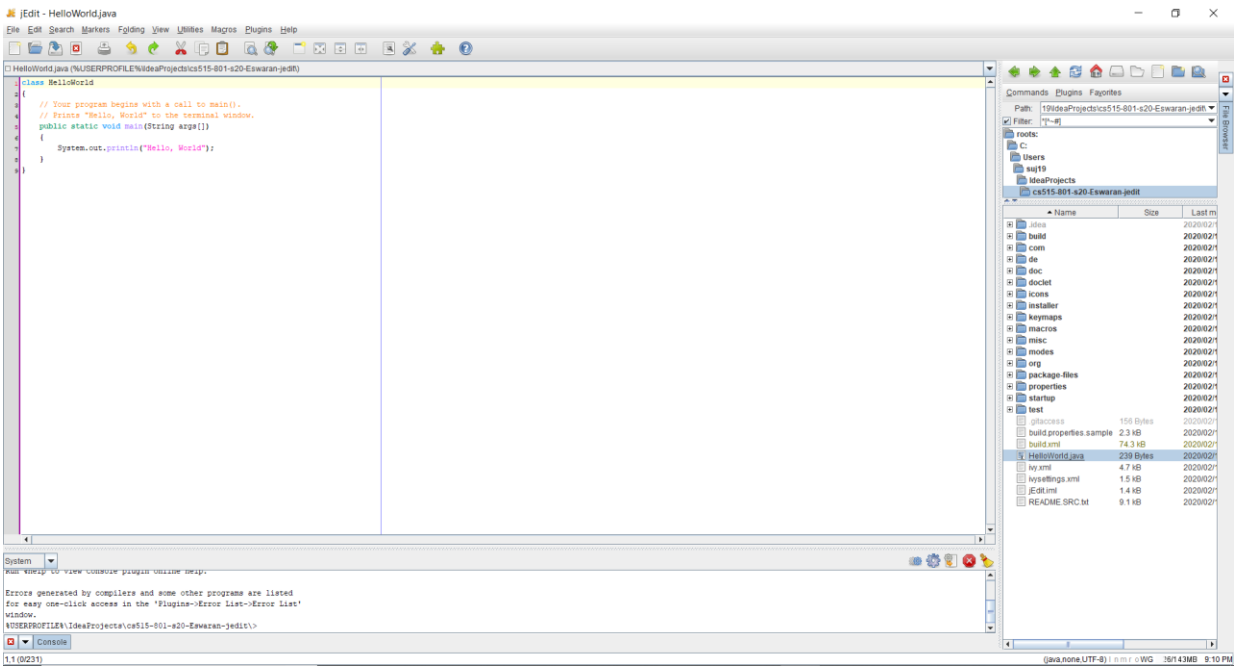

Figure 5: *Demonstrating HelloWorld.java program in Jedit*

As far as now, there is not any error to be viewed. We have viewed hello world as the output. Thus, jedit can be considered as a compiler tool for java.

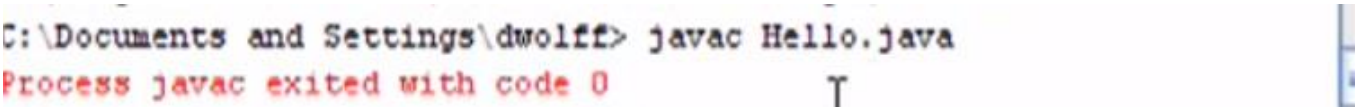

Figure 6: *Compiling of HelloWorld.java program in Jedit*

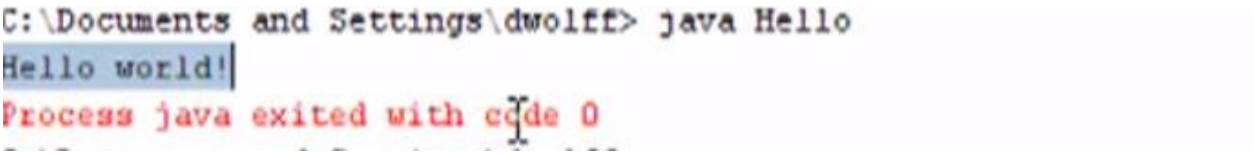

Figure 7: *Execution of HelloWorld.java program in Jedit*

It can support languages like pascal, ruby, scala, MySQL, XML, HTML, Python, C, C++ and C#. The file types associated with this editor are lex, bibtex, pdf, tex, ptl, lisp and latex. The main advantage of bracket matching which leaves quoted literals and comments. According to the source, it has about 200 plugins available in jedit.

Below, there is the UML diagram for this project which involves the packages in it. This is drawn with the help of SimpleUML plugin in Intellij.

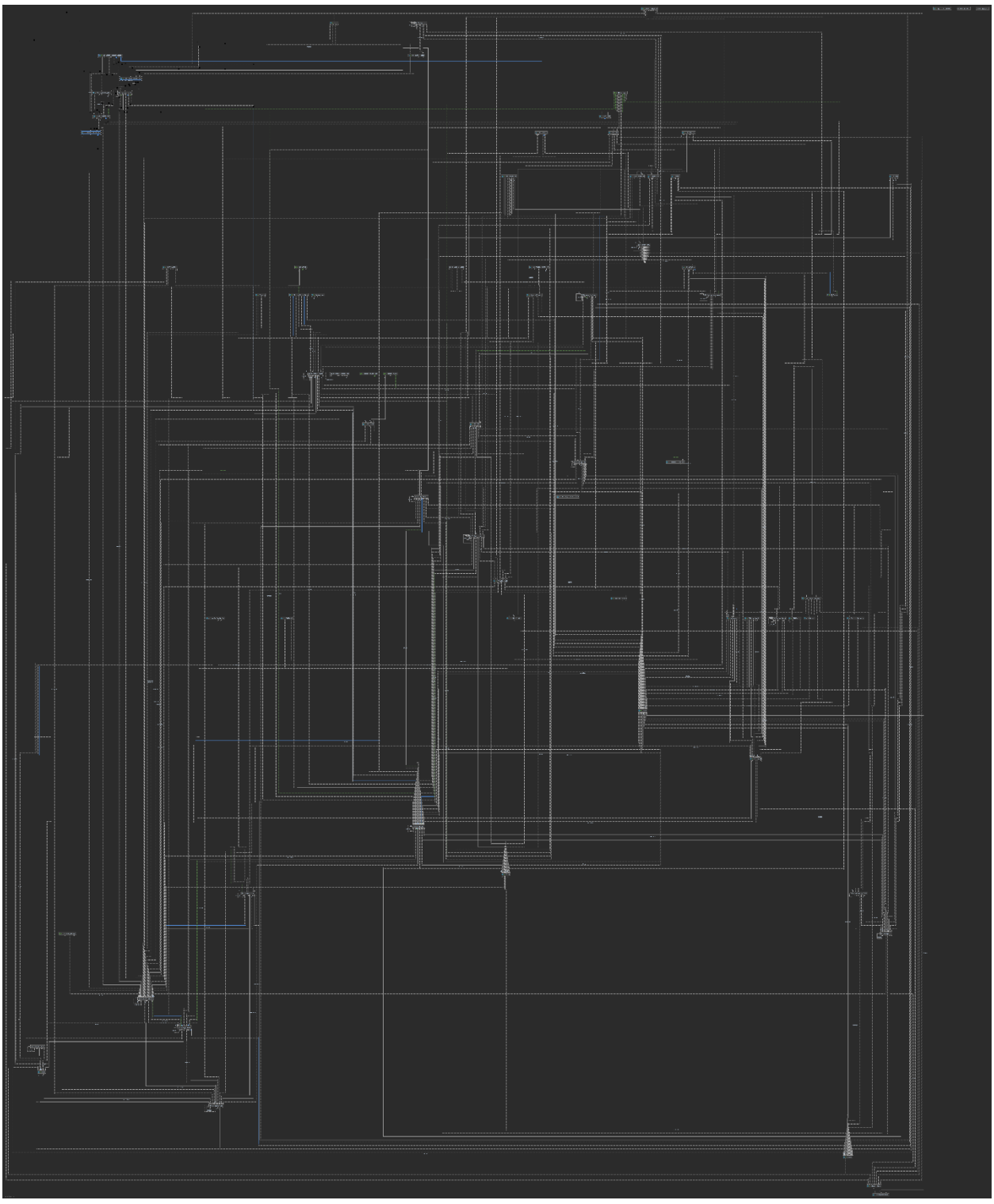

Figure 8: *UML Diagram of Jedit with the help of SimpleUML plugin in Intellij*

List	Description
.idea	Consists of XML of Copyrights with the libraries and scopes in it.
Build	It is the main part of the project where it contains all the functionalities of the projects like AutoSave, EditPlugin and so on.
com	List of java programs of various document types with java properties in it.
De	An ant task of functionality of calculate size and archives.
Doc	A simple list of text files contains readme, copyright and guidelines.
Doclet	Helps in generating TOCXML in java
Icons	Pictures of Jedit icons with file symbol in it.
Installer	Various installers of the projects in it like TarInput, TarOutput and CRC.
Keymaps	List of shortcut keypads needed for jedit
macros	Contains all the bash files for jedit
misc	It has the code for debugging the faults in the project.
modes	Consists of several modes of operations in xml file type.
org	It mainly focuses on the task operations between one after the other.
Package-files	Packages for several operating systems.
startup	Basic readme file for the project.
test	Several test cases for running this project.

Table 1: *Description of various folders in the package of Jedit*

The bugs associated while compiling it on Jedit text editor are:

- While opening a new text file, save it under default name untitled-1, later close it and return to jedit it begins with the location where it last saved it with same empty page title untitled-1. But when trying to open the first saved untitled-1, it fails to open.
  Expected Result: It should work normally like a text editor opening the first saved file instead stops opening.
  Steps to reproduce:
  1. Make sure you have saved your file as default i.e. “Untitle-1” in any location.
  2. Exit and Re-enter Jedit.
  3. Go to file and open Untitled-1 from the location it was saved.

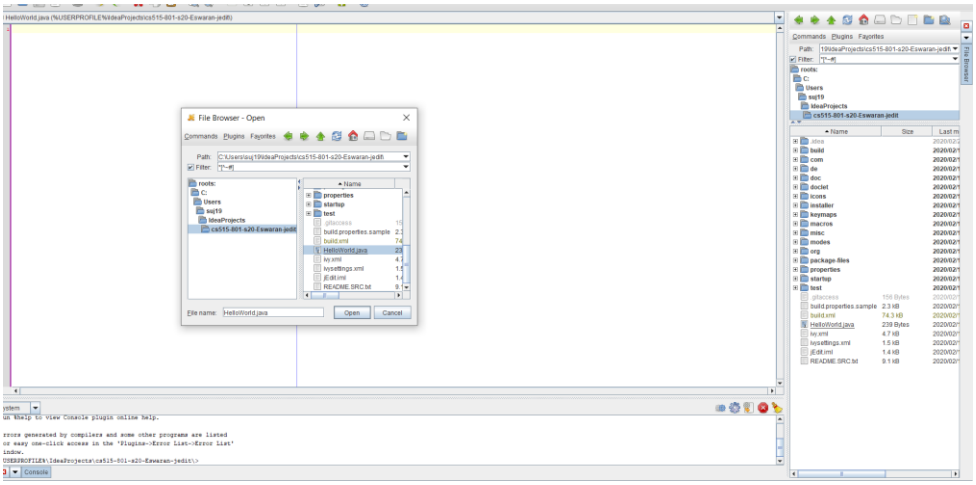

Figure 9: *Bug while opening a file*

- While working on a large program or a lengthy process, the message "OutOfMemoryError" seems to be visible since the memory allocation of Java for jedit is less.

  Expected Output: The expected result should have produced the output for it, instead it showed lot of memory used because of infinite loops.

  Steps to reproduce:

  1. Build a program where the condition always evaluates to be true. A sample line of the code is given below:

```
public void Try( int i )
{
    while ( i != 0 )
    {
        i-- ;
    }
}
```

  2. Save your program code and run it.
  3. The result would keep on going, thus results in "OutOfMemoryError"

```
Iteration 12 Free Mem: 60183176
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
```

Figure 10: *Bug while running a infinite loop program*

No.	Bug Report	Expected Result	Observed Result	Slack Traces if exists
1	Fails to open the saved default file	It should work normally like a text editor opening the first saved file instead stops opening.	It fails to open instead asking to open another file.	
2	"OutOfMemoryError" for large files	It should function properly instead resulting an unexpected error.	The message "OutOfMemoryError" seems to be visible	OutOfMemoryError

Table 2: *Bug with expected results and observed results in Jedit*

The features that can be implemented in Jedit:

- Increase the memory allocation of the jedit which can help in running a large program which can help in improving the software.

Figure 10: *Overview of PDFSAM*

- Implement an automated test generation which can help in understanding the test coverage.

PDFSAM

PDFSAM is a open project which was built to do features like split, merge, extract pages, rotate and mix all the documents. PDFSAM Basic is written in JavaFx where the version 4 requires JRE 8 or more.

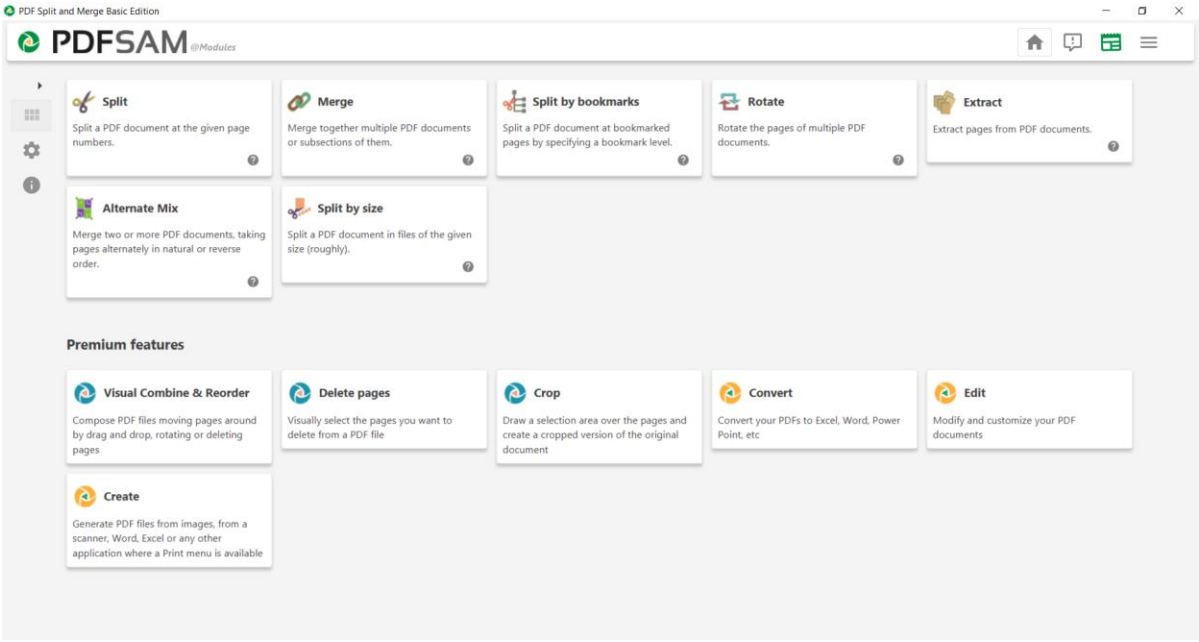

Figure 11: *Overview of PDFSAM*

Various functionalities of PDFSAM are:

1. Extract:
   Selection of multiple PDF are files for extracting pages to a single file.

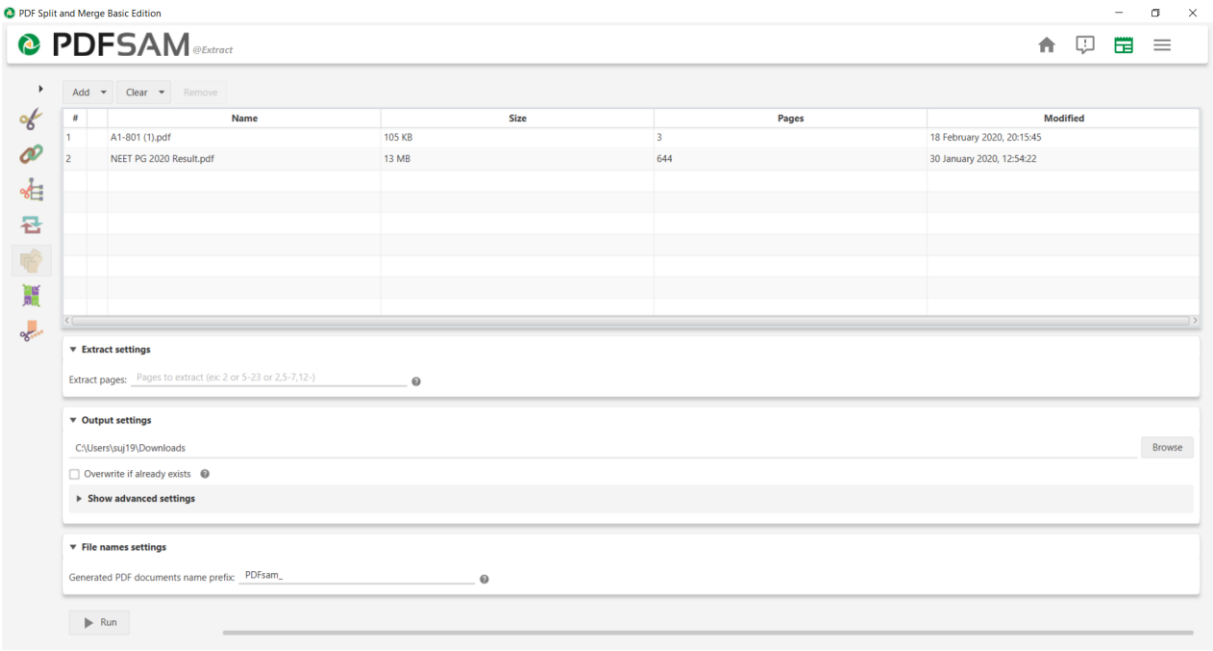

Figure 12: *Extract Operation in PDFSAM*

2. Split:

   Splitting can be done either every page, even odd pages, by bookmark levels, or even by given set of page numbers.

Figure 13: *Split Operation in PDFSAM*

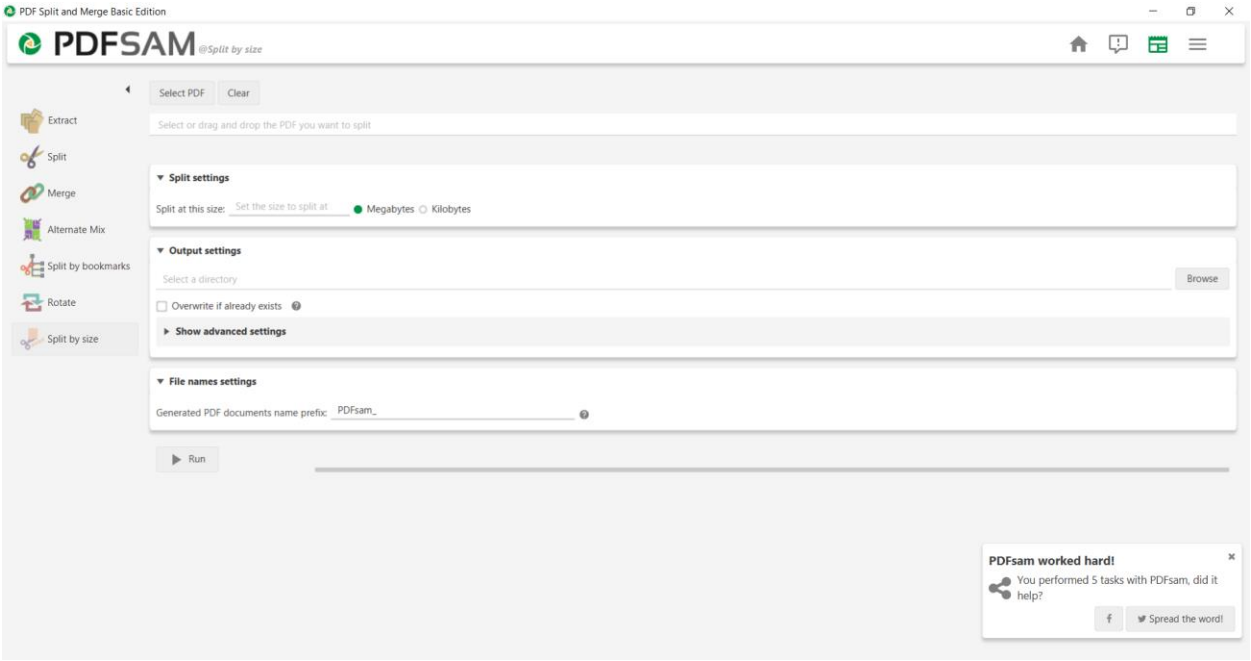

Figure 14: *Split by Size Operation in PDFSAM*

3. Merge

It selects the entire documents or subset of it where it has an option of generating table of contents with normalized page size and add blank pages to it.

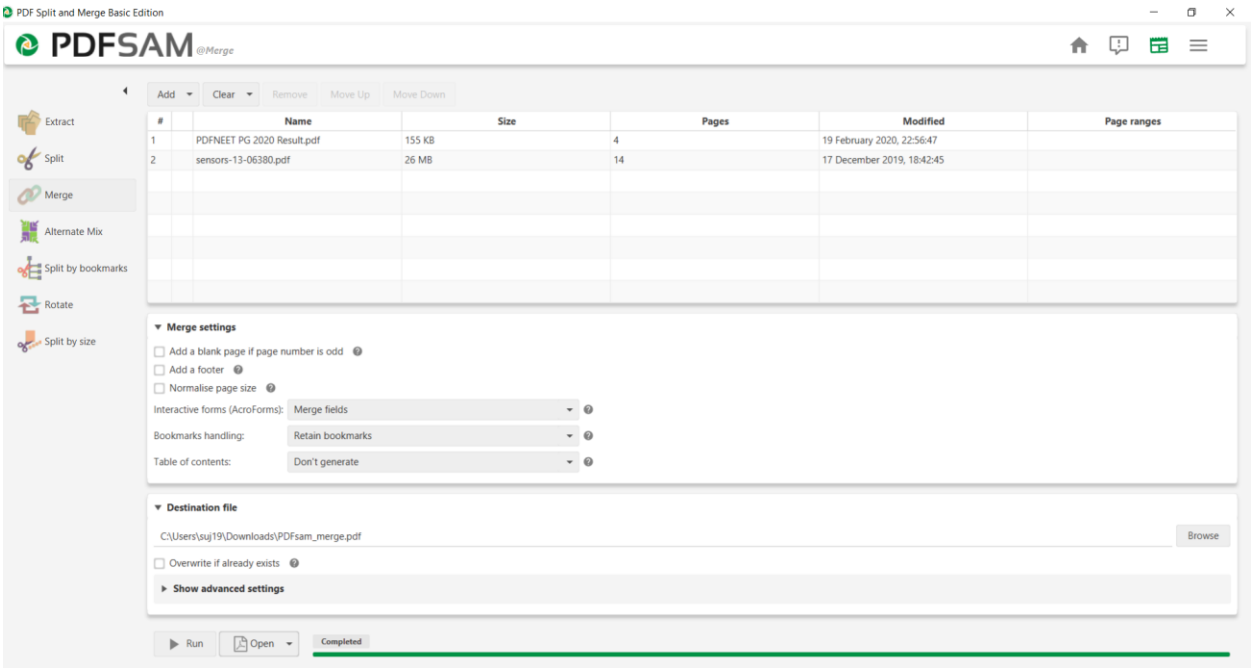

Figure 15: *Merge Operation in PDFSAM*

4. Rotate

Multiple files can be rotated either every page or a selected list of pages.

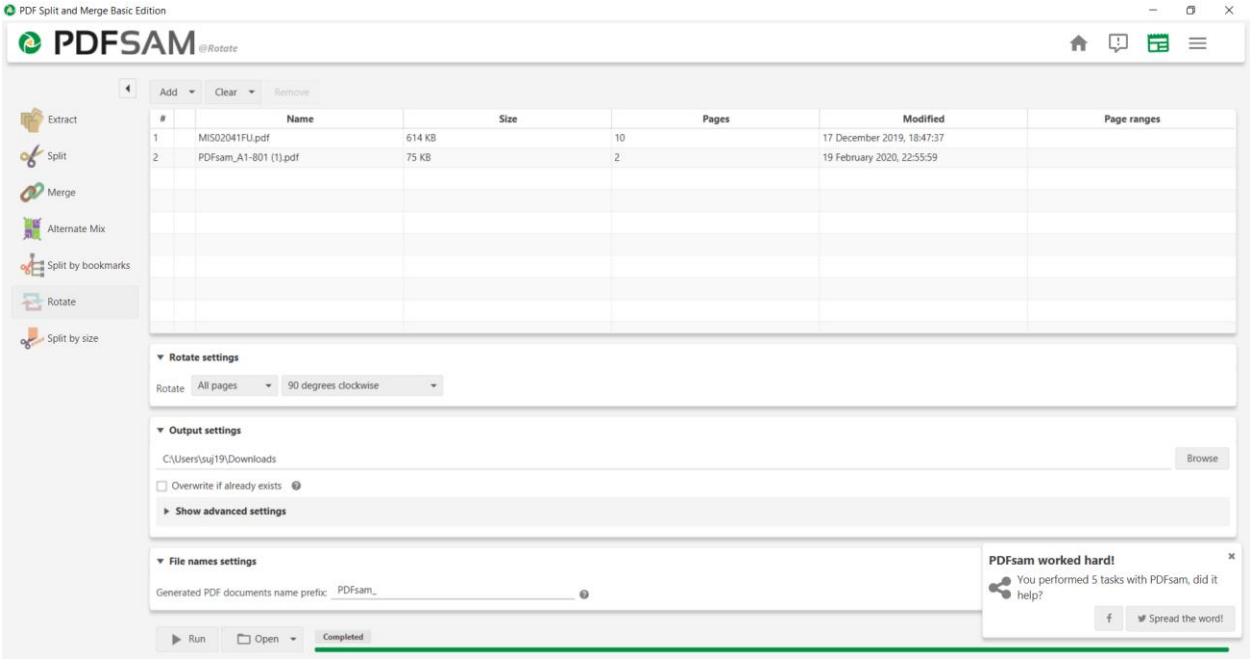

Figure 16: *Rotate Operation in PDFSAM*

5. Alternate Mix:

Mixing of PDF files can be done where a list of pages is merged thus taking alternate ones from them.

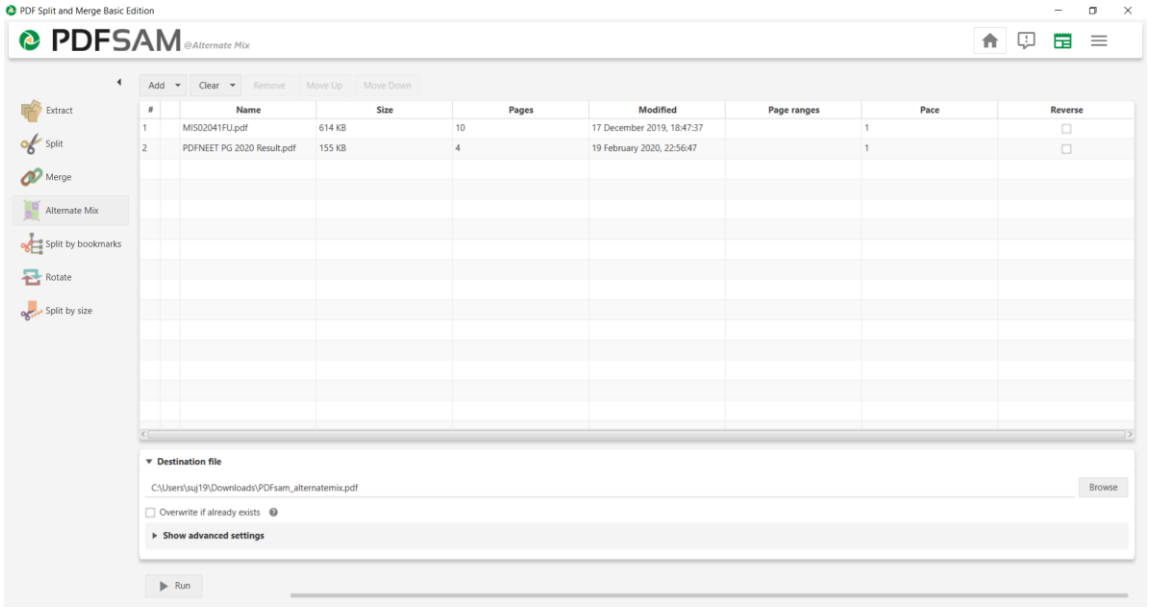

Figure 17: *Alternate Mix Operation in PDFSAM*

List	Description
.github	Consists an yml of the author
.idea	XML files for the project to display
Pdfsam-alternate-mix	Program codes for alternate mix of multiple pfds
Pdfsam-basic	Program codes for utilizing the basic PDFSAM
Pdfsam-core	Program codes for utilizing the core PDFSAM
Pdfsam-docs	Program codes for utilizing various docs in PDFSAM
Pdfsam-extract	Program codes for utilizing the extractions multiple PDFs
Pdfsam-fx	Consists of codes for functionalities like file choosing browsing and so on.
Pdfsam-gui	User Interfaces needed for the project App.
Pdfsam-i18n	Code for implementing i18n which was not explored.
Pdfsam-merge	Consists of programs for merging the PDFs.
Pdfsam-rotate	Consists of programs for rotating the PDFs.
Pdfsam-service	Consists of programs for loading various PDFs.
Pdfsam-simple-split	Consists of programs for splitting the PDFs.
Pdfsam-split-by-bookmarks	Consists of programs for merging the PDFs by bookmarks.
Pdfsam-split-by-size	Consists of programs for splitting the PDFs by size.

Table 4: *Bug with expected results and observed results in PDFSAM*

The bug associated while compiling it on PDFSAM Basic are:

- The size of the inserted set of PDF files are not matching, eventually resulting in different sizes while merging.
  Expected Output: It should produce the equivalent page while merging instead gave different sizes
  Steps to Reproduce:
  1. Select two PDF of different sizes
  2. Go to PDFSAM merge operation
  3. Run the operation and open the file.

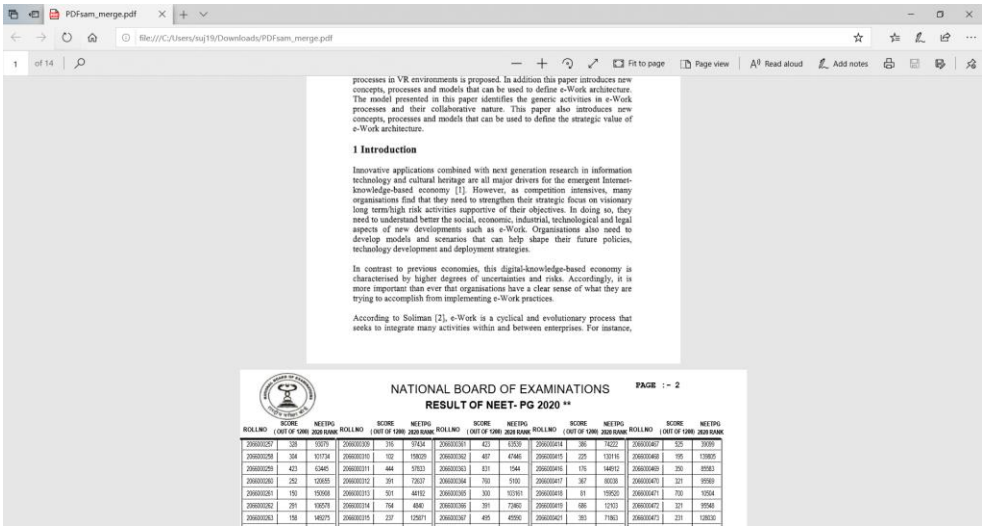

Figure 17: *Bug associated with Merge Operation in PDFSAM*

- After merging two PDF of same size, could not rotate 180 degrees even though the completion says it is done.
  Expected Result: The pages in the PDF should have been inverted but it results the same.
  Steps to Reproduce:
  1. Merge two documents of same size.
  2. Go to rotate operations.
  3. Select the merged PDF to be rotated.
  4. Run the operation and open the file.

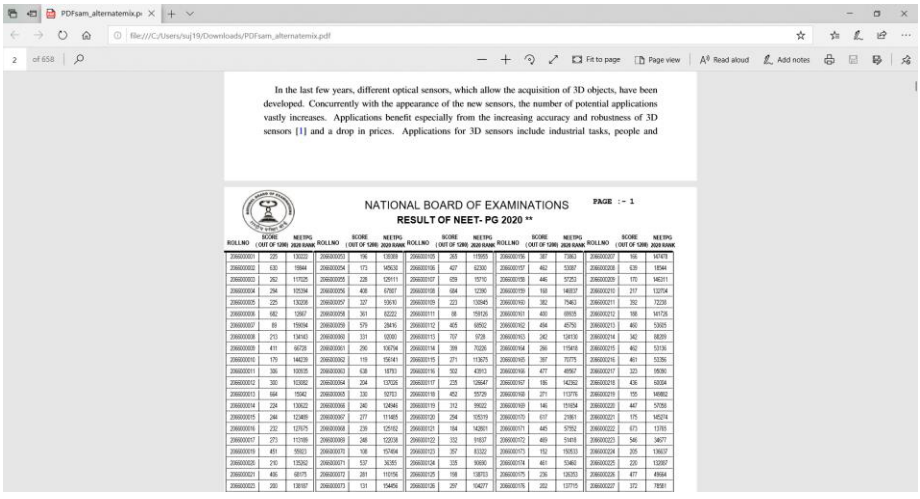

Figure 18: *Bug associated with Rotate Operation of a merged PDF in PDFSAM*

No.	Bug Report	Expected Result	Observed Result	Slack Traces if exists
1	Size of merged PDF are not equivalent	It should produce the equivalent page while merging instead gave different sizes	It showed different sizes even though the size is given the same.	
2	Merged documents cannot be rotated	The pages in the PDF should have been inverted but it results the same.	It should have inverted but resulted the same.	

Table 4: *Bug with expected results and observed results in PDFSAM*

New features for improving PDFSAM can be:

- Implementing conversion of various file types to pdf which can help in formatting the documents according to the needs. For example, creating a option of converting CSV to PDF which can be used to merge the table records in another PDF.

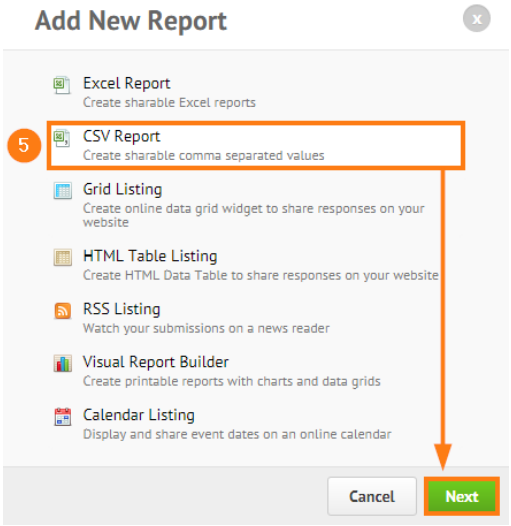

Figure 19: *Usage of CSV files on to PDFSAM for conversion to PDF*

- Adding up of text in PDF can be helpful in filling forms and e-signatures since they are difficult to work with text in PDF files, as they are perceived to be pictures. Thus editing can also be added as a feature in the near future.

REFERENCES

1. Wenzel, M. (2012, July). Isabelle/jEdit–a Prover IDE within the PIDE framework. In *International Conference on Intelligent Computer Mathematics* (pp. 468-471). Springer, Berlin, Heidelberg.\
2. Lu, X., Zhuge, J., Wang, R., Cao, Y., & Chen, Y. (2013, January). De-obfuscation and detection of malicious PDF files with high accuracy. In *2013 46th Hawaii International Conference on System Sciences* (pp. 4890-4899). IEEE.
3. http://www.jedit.org/
4. https://pdfsam.org/

NOTE

1. Project cloned from: http://svn.code.sf.net/p/jedit/svn/jEdit/tags/jedit-5-5-0/ https://github.com/torakiki/pdfsam
2. Jedit cloned in repository: https://github.com/surajeswar22/cs515-801-s20-Eswaran-jedit
3. PDFSAM cloned in repository: https://github.com/surajeswar22/cs515-801-s20-Eswaran-pdf
4. For Jedit, Ant built was done with Java version 8 and for PDFSAM, maven built was done with java version 11.
5. IDE used here is Intellij.