# ASSIGNMENT 1: GETTING FAMILIAR WITH JEDIT AND PDFSAM

Sanket Mehrotra, Nada Alalyani

EVOLUTIONARY RAMS CS515

# Table of Contents

PDFSam – v4.0.5 ..... 2
Introduction ..... 2
Code Structure ..... 2
Suggested Features ..... 3
Application/Functional Bug ..... 3
*Bug 1: Module: Split by bookmarks* ..... 3
*Bug 2: Page Extraction module* ..... 6
JEdit – v5.5.0 ..... 8
Introduction ..... 8
Code Structure ..... 8
Suggested Features ..... 9
Application/Functional Bug ..... 9
*Bug 1: File Not Found Exception* ..... 9
*Bug 2: jEdit Help window GUI* ..... 10
Other Miscellaneous bugs found while going over the code and using the applications ..... 12
jEdit – Bug ..... 12
PDFSam – Code Template Issue ..... 12

# PDFSam – v4.0.5

## Introduction

PDFSam is an open source software developed for manipulating pdf files. It offers several features allowing the user to merge, split, extract, mix and even rotate pages in a given file. It does not offer an editor, or even a pdf viewer as part of its functionality, those being offered in its paid version 'PDFsam Enhanced'. Several other premium features such as pdf signing, and attachments are offered in the Enhanced version. We built PDFSam from source on Eclipse as specified in instructions present in A1.pdf.

## Code Structure

1. **pdfsam-docs:** This folder contains
    a. Icons
    b. A font package is called 'sansation'
    c. A code file template which is applied across the project.
    d. A latest news/changes document in the form of an html file.
    e. Code format and templates
    f. Checkstyle plugin information

2. **pdfsam-fx and pdfsam-gui**: Contain code and classes for:
    a. GUI for the whole application (Dashboard, notification window, log window)
    b. Custom animations: Affects that are invoked during actions (e.g. Clicking the "what's new" icon in the top right of the window shifts the content left to make space for a column in which a Custom window appears.)
    c. Custom controls. (ones that are not generally found in swing or javafx) (e.g. Open file and confirmation dialogs)

3. **pdfsam-alternate-mix:** Code which handles the functionality and tests for the alternate mix module, with a few custom icons.

4. **pdfsam-i18n**: This module provides services and string translations to handle different locales of languages in the application.

5. **pdfsam-merge**: This module contains the following:
    a. Code and classes for the merge-pdf functionality.
    b. A custom icon for the left-hand side menu.
    c. Tests to be run for this functionality

6. **pdfsam-simple-split, pdfsam-split-by-bookmarks and pdfsam-split-by-size:** Grouping all these similar modules, we can see that they handle the functionalities of splitting the pdf according to certain features, such as:
    a. Size(MB,KB),
    b. Bookmarks(with different levels of bookmarks or certain fixed bookmarks, like 'Section' included)
    c. Regular splitting by pages (page numbers, odd/even pages and splitting after fixed intervals.)

7. **Pdfsam-rotate and Pdfsam-extract:** All of these modules represent the various functionalities present in PDFSam Basic. Like the merge and split folders mentioned earlier, these also contain:
    a. Code and classes for the rotate/extract-pdf functionality.
    b. A custom icon
    c. Tests to be run for each functionality

## Suggested Features

1. A feature to find the table of contents and extract chapters. Use Case: If a user is reading a book or Ph D dissertation, they are really long (some dissertations can be up to 200 pages in length). Maybe the user does not want to print the whole thing out like a textbook, but instead read it one chapter at a time. In this case, maybe this functionality will be helpful.

2. Feature to rip images or tables from a given file. Use Case: If a user is reading a research paper and wants to extract the tables or images from a pdf in the correct formatting(tables) or resolution(image). Then having this functionality would be helpful.

## Application/Functional Bug

### *Bug 1: Module: Split by bookmarks*

#### Summary

When using the split by bookmarks module, running it on a .pdf file (with bookmarks) and entering a regular expression searching for a heading with at least one space in front of it. It fails silently, with just the status and an application log being dropped. Leaving me to guess why this is happening.

#### Version

V4.0.5

#### Steps to reproduce

1. Open the 'split by bookmarks' module.
2. Selected a pdf with any number of bookmarks
3. Enter the regex: ^.* +$

#### Expected Behavior

1. Message telling me that file does not include given regex

2. Message stating that it is an invalid regex.

### Observed Behavior

1. Failed status displayed above progress bar.
2. Application Logs (Log register) icon blinking.
3. Stack trace dumped in logs.
4. No other functionality

### *Stack Trace*:

```
ERROR  [21:38:32]: Task (org.sejda.impl.sambox.SplitByOutlineLevelTask@311fc502) execution failed.
org.sejda.model.exception.TaskExecutionException: Unable to split, no page number given.
at org.sejda.model.split.SplitPages.ensureIsValid(SplitPages.java:59)
at org.sejda.model.split.PageDestinationsSplitPages.ensureIsValid(PageDestinationsSplitPages.java:44)
at org.sejda.impl.sambox.component.split.AbstractPdfSplitter.split(AbstractPdfSplitter.java:71)
at org.sejda.impl.sambox.SplitByOutlineLevelTask.execute(SplitByOutlineLevelTask.java:79)
at org.sejda.impl.sambox.SplitByOutlineLevelTask.execute(SplitByOutlineLevelTask.java:46)
at org.sejda.core.service.DefaultTaskExecutionService.actualExecution(DefaultTaskExecutionService.java:148)
at org.sejda.core.service.DefaultTaskExecutionService.execute(DefaultTaskExecutionService.java:71)
at org.sejda.core.service.DefaultTaskExecutionService.execute(DefaultTaskExecutionService.java:58)
at org.pdfsam.task.TaskExecutionController.lambda$request$0(TaskExecutionController.java:87)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
at java.base/java.lang.Thread.run(Unknown Source)
```

## Screenshot

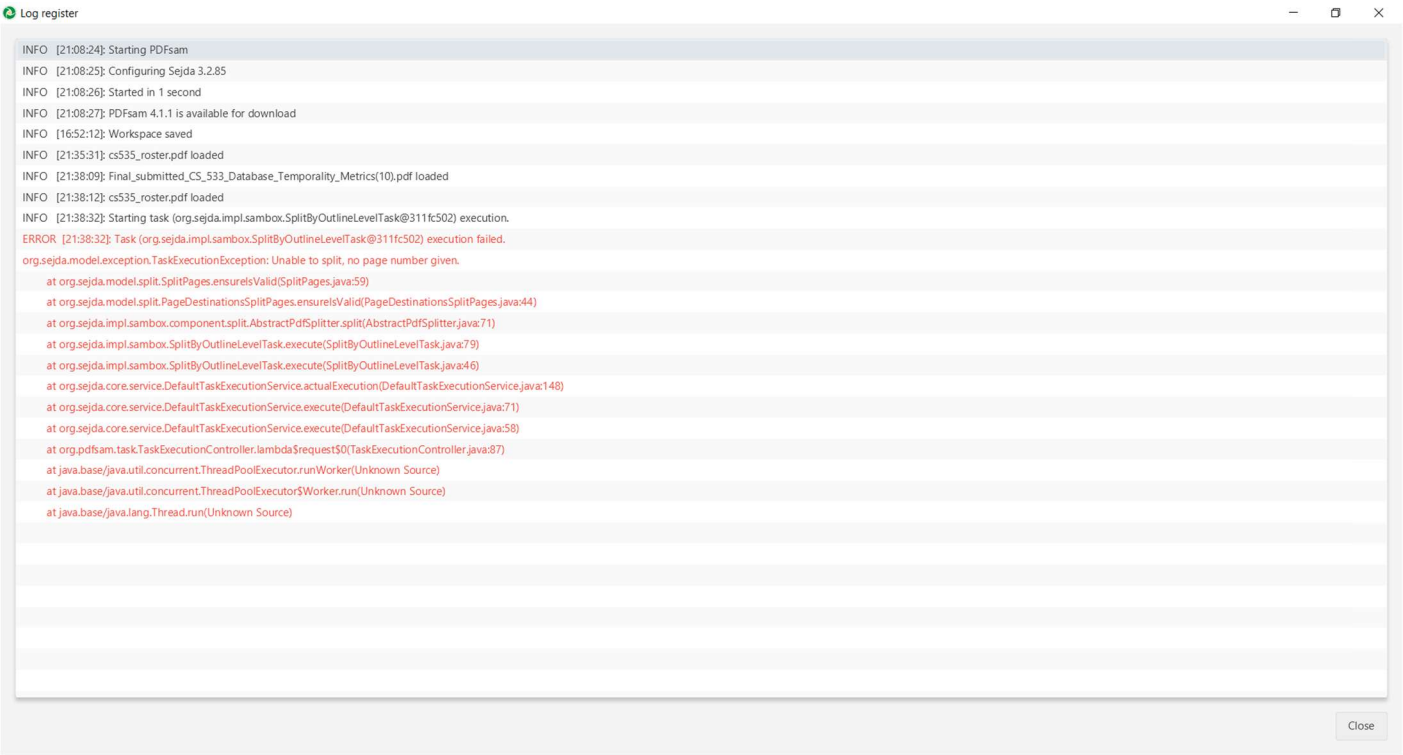


Figure 1: PDFSam Application screen when reproducing bug 1

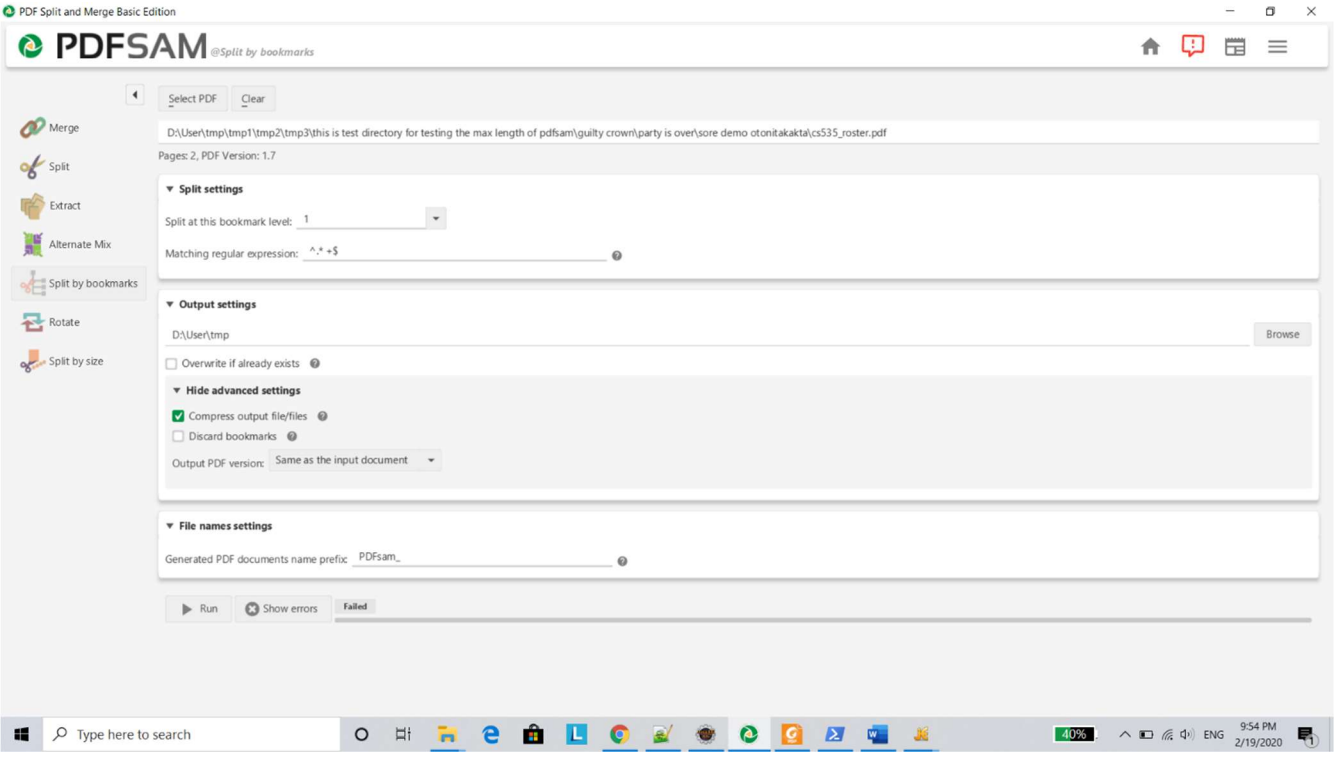


Figure 2: PDFSam: Screenshot of the application log stack trace of bug 1

*Bug 2: Page Extraction module*

### Summary

After adding a pdf file with only one page, I accidently typed in the page number to extract as 2 and the execution silently failed, dropping stack trace logs but I didn't exactly understand what was going on.

### Version

V4.0.5

### Expected Behavior

1. A message telling me that the page number that I have entered does not exist in the pdf.
2. OR a message telling me that the page number is invalid.

### Observed Behavior

1. Failed status badge above the progress bar.
2. Application logs icon becomes highlighted.
3. Stack trace contains the message "No page selected for extraction"

## Screenshots

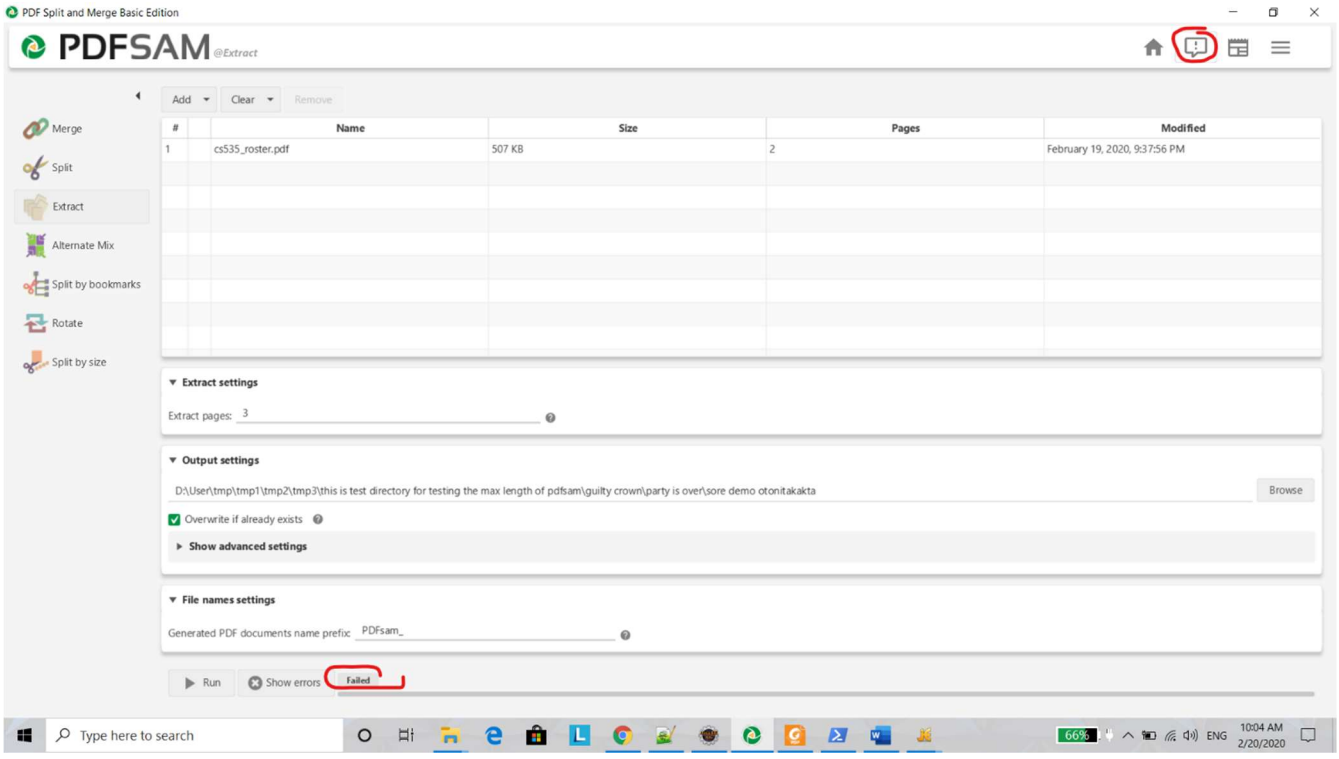


Figure 3: PDFSam Application screen when reproducing bug 2

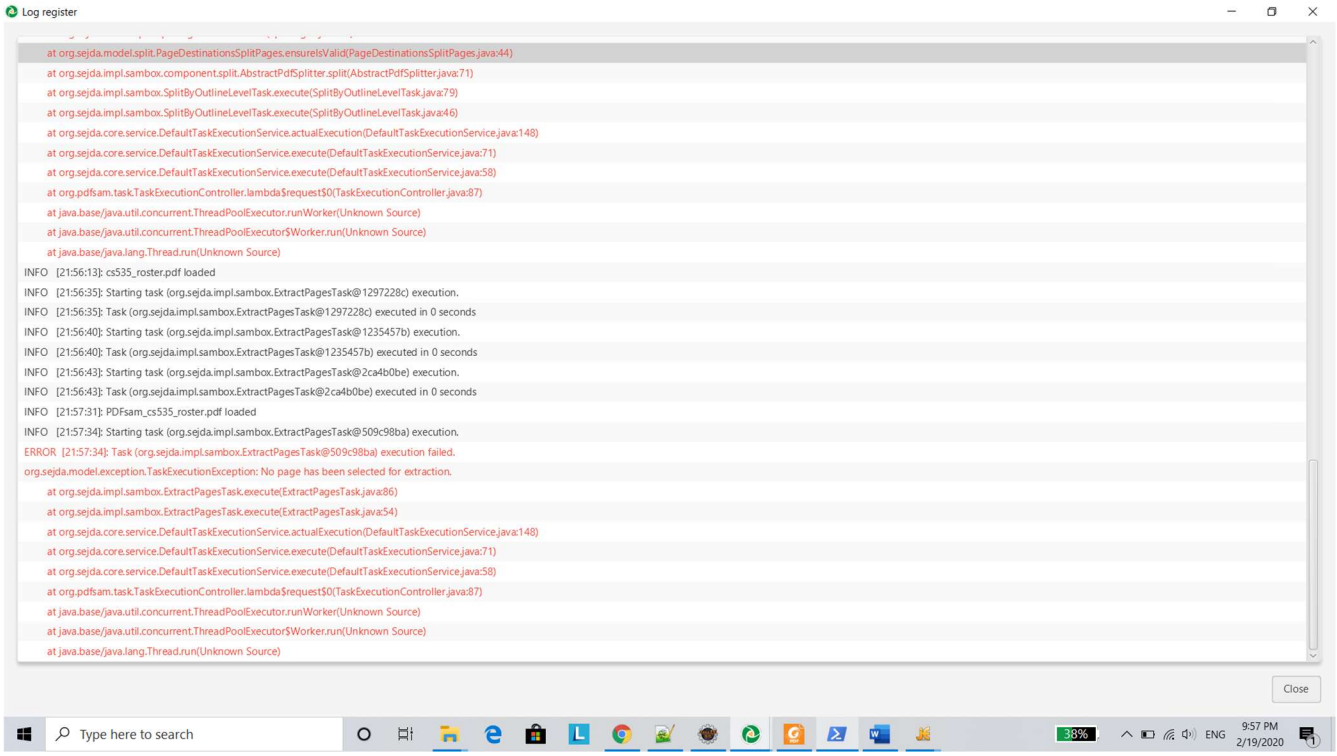


Figure 4: PDFSam Stack trace on application logs window

# JEdit – v5.5.0

## Introduction

jEdit is a programmer's text editor, which was originally developed by Slava Pestov available under the GNU GPL v 2.0. It allows itself to be customized using macros written in BeanShell and has a flexible plugin architecture allowing its features to be extended by other programs. We built jEdit from source on Eclipse using Apache Ant as specified in instructions present in A1.pdf.

## Code Structure

1. **Browser package**: This package contains classes for browsing file from the system
2. **Buffer package**: This package contains several classes containing the main text-buffer in jEdit and implements its signature hard code folding (with the '{{{' and '}}}').
3. **Bsh Package**: This package concerns the handling of BeanShell scripted macros written to automate repetitive tasks in jEdit.
4. **Bufferset Package**: classes related to implementation of the BufferSets feature of jEdit.
5. **Datatransfer Package**: classes related to encoding and decoding MIME data to/from system clipboard, etc.
6. **GUI Package**: it contains all classes related to the graphical interfaces of the application
7. **Help Package**: contains classes for help functions such as search function
8. **Icons Package**: This package involves all icons for actions and document status.
9. **Indent Package**: contain classed which related to code indentation rules and functions.
10. **Input**: this package contains classes related to handling Key bindings and processing key board shortcuts.
11. **IO:** This package contains classes related to virtual file system and multi-threaded I/O
12. **Menu Package:** classes for generation of dynamic menus
13. **Msg Package:** handles the internal messaging of the application. e.g the event messages that are emitted when certain events occur (e.g. loading a plugin or editing a buffer).
14. **Options Package:** contains classes that provides Global Options dialog box panes
15. **Pluginmgr Package**: contains the UI, installation and managing of plugins into jEdit.
16. **Print Package:** contains the UI(Print Preview UI) and all classes relating to printing the text(pre-formatting).
17. **Proto Package:** contains classes that deal with URL protocol handler, which used to load resources from plugin JAR files.
18. **Search Package:** contains of classes that provides search and replace functions.
19. **Syntax Package**: The jEdit syntax highlighting engine. This package only depends on a few jEdit classes :- org.gjt.sp.util.SegmentCharSequence and org.gjt.sp.util.Log. It contains a parser, token handler, and syntax rules to implement highlighting in jEdit.
20. **Text area Package**: Classes related to jEdit's TextArea.
21. **Visitors Package**: Implements a visitor design pattern for classes for EditPanes, Views, and Text Areas

## Suggested Features

1. Presenting an outline for a given code file. As shown in Fig 4, This has been already implemented in Eclipse , It would be useful to scan a code file to detect its classes and methods.

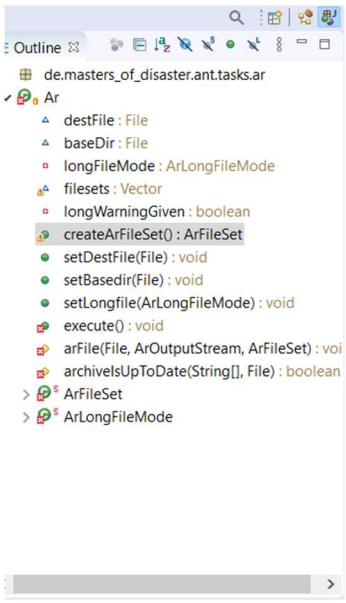


Figure 5: Eclipse Code outline view

2. Line Operations: Keyboard shortcuts for group commenting/uncommenting lines or changing case or removing empty lines as implemented in Eclipse and Notepad++ is a feature that we find very useful as programmers. jEdit has custom shortcut keys that are definable, but no explicit line operations that shortcut keys can be mapped to.

## Application/Functional Bug

### *Bug 1: File Not Found Exception*

#### Summary

The Help tab in Jedit has three options. One of these options is JEdit Help which providing documents for different purposes such as invoked document. When user clicks on these documents, the File not found Exception massage is shown. What expected is that the HTML files with the required details should be presented when these documents' links are invoked.

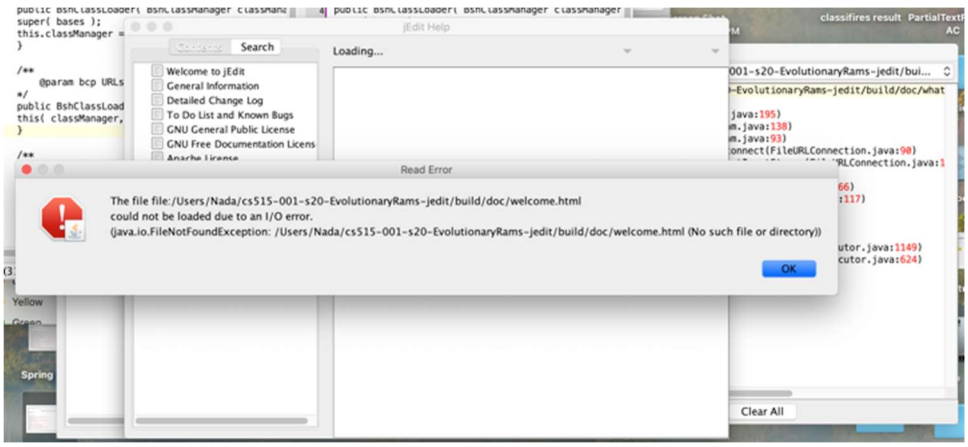


Figure 6: jEdit error dialog box when reproducing bug 1

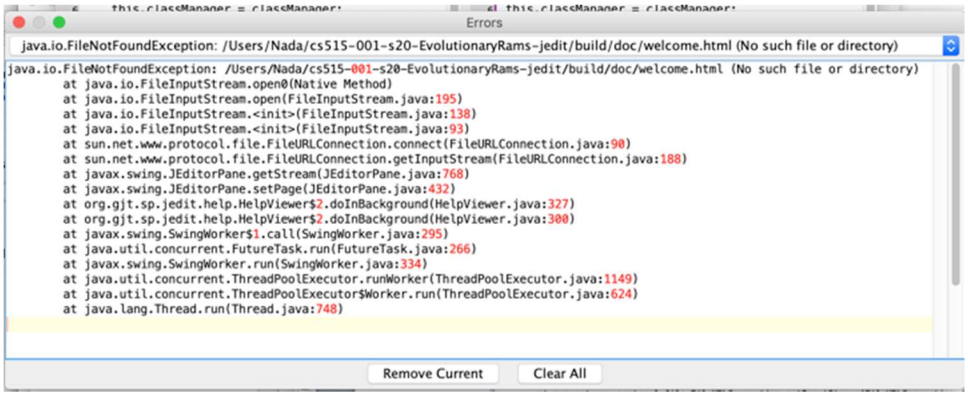


Figure 7: jEdit Stack Trace dumped for bug 1

## *Bug 2: jEdit Help window GUI*

### Summary

Exploring the Help section of jEdit, We came across a bug in the GUI which would cause a little unexpected confusion to a user. A unmarked control was left on the UI which duplicates the feature of another element in the UI.

### Steps to reproduce

1. Open jEdit help (Toolbar: ‘Help’ -> jEdit Help OR press F1)
2. Select any page other than the welcome page (e.g. ‘General Information’)

### Expected Behavior

1. Open jEdit help (Toolbar: ‘Help’ -> jEdit Help OR press F1)
2. Select any page other than the welcome page (e.g. ‘General Information’)
3. The navigation (arrow) buttons in the top right will darken, allowing the one on the left to be used to go back and the one on the right to go forward.

### Observed Behavior

4. There is a button on the UI in between the back and the forward buttons that does not have a icon displaying it to the user but hovering your mouse over it highlights it

anyway. This hidden middle button has the same ‘Forward’ functionality as the right hand side button.

## Screenshots

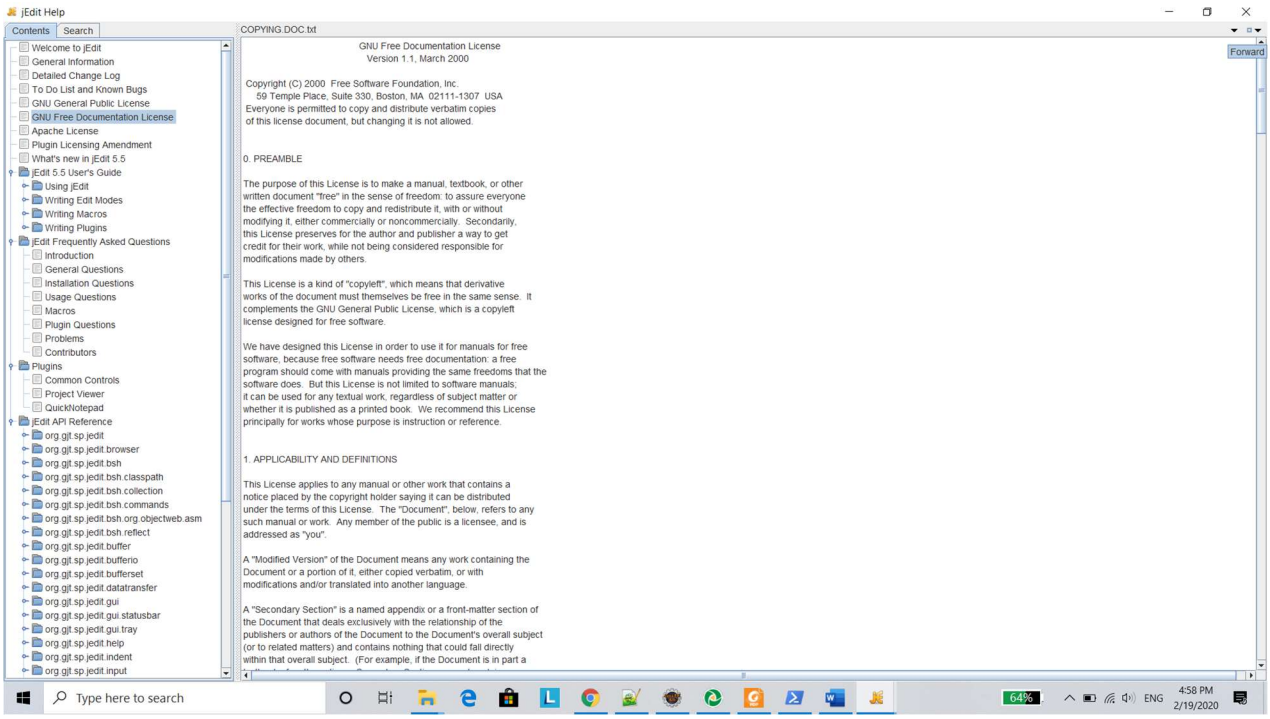


Figure 8: jEdit: The Help window

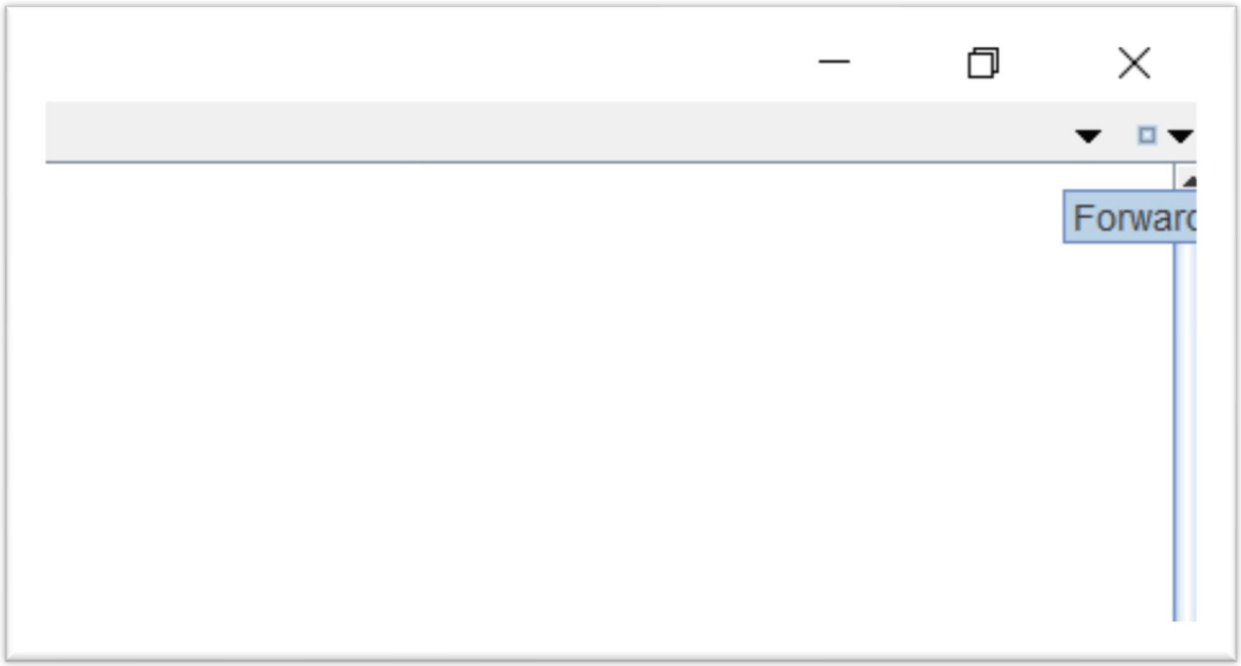


Figure 9: The hidden button as highlighted by a mouse.

# Other Miscellaneous bugs found while going over the code and using the applications

## jEdit – Bug

While using the application, I installed a recommended plugin called Project Viewer.

Opened Toolbar Menu item Plugins -> ProjectViewer -> Projects -> New project here.

This opened a window which was sized incorrectly. Not all the fields were visible.

## PDFSam – Code Template Issue

Even code files in other functionalities(Extract, or Alternate Mix) start with the code comment

```
This file is part of the PDF Split And Merge source code…
```

This seems like a template applied globally.