Software Implementation Document

for

iText Based PDF Viewer

Version 1.0

Prepared by

Dhruvkumar Patel

Jace Robinson

Pranav Pranav

CS 7140 Advanced Software Engineering

July 20th 2016

Build Instructions:

The source code is given as a tar ball and an executable jar. To see the source code, simple extract the tar ball. To execute the code, double click the jar file or type java -jar JAR_FILE from command line, where jar_file is replaced with jar path name. This command should cause the initial GUI to appear.

Instructions on how to interact with the GUI are given below in the “Smoke Test” section with the screenshots.

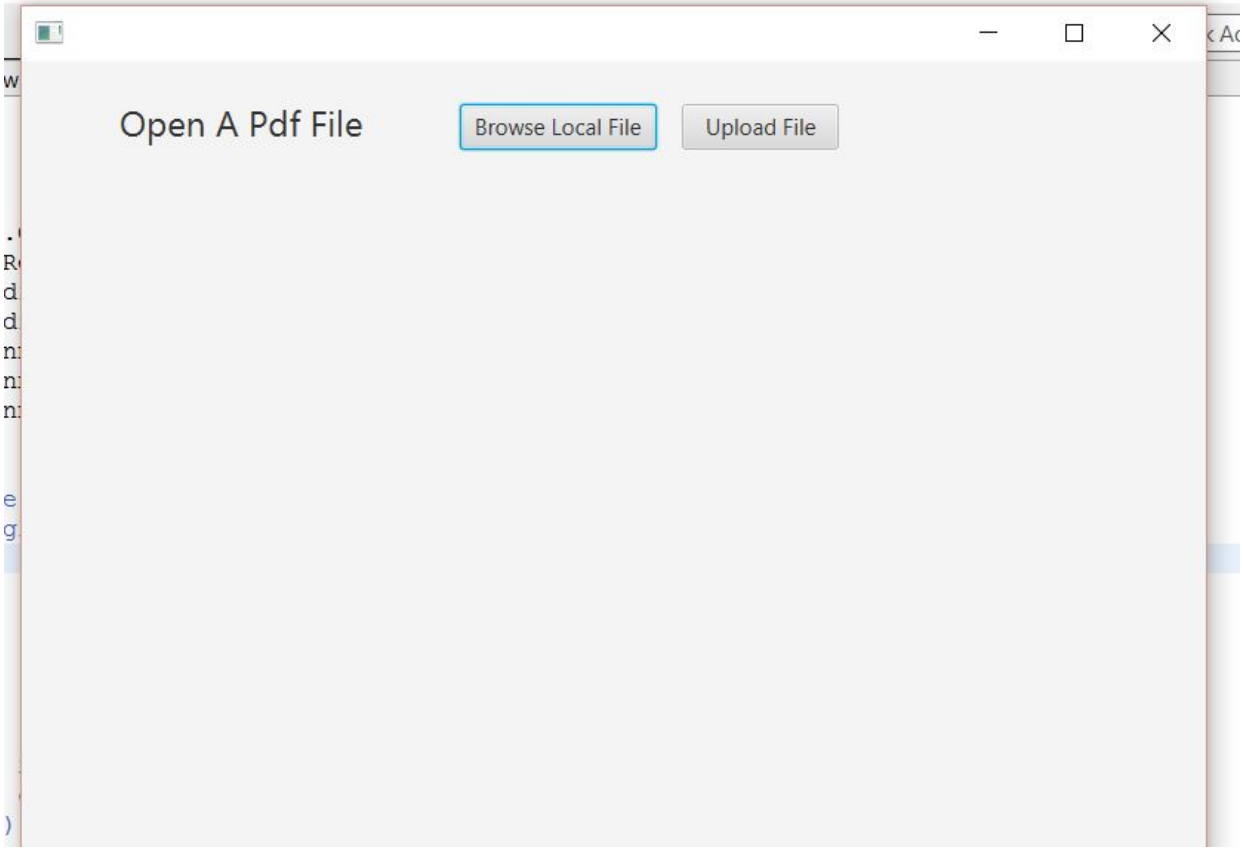
“Smoke Test” and Screen Shots

The smoke test was completed by uploading our requirements document as the test PDF (with some existing annotations added from PDF-XChange editor). Then we displayed contents of the PDF to a GUI using our viewer. Within this viewer the user has the ability to add and remove annotations. Lastly the user saves the file to a PDF and screenshots of the final result in PDF-Xchange editor is given.

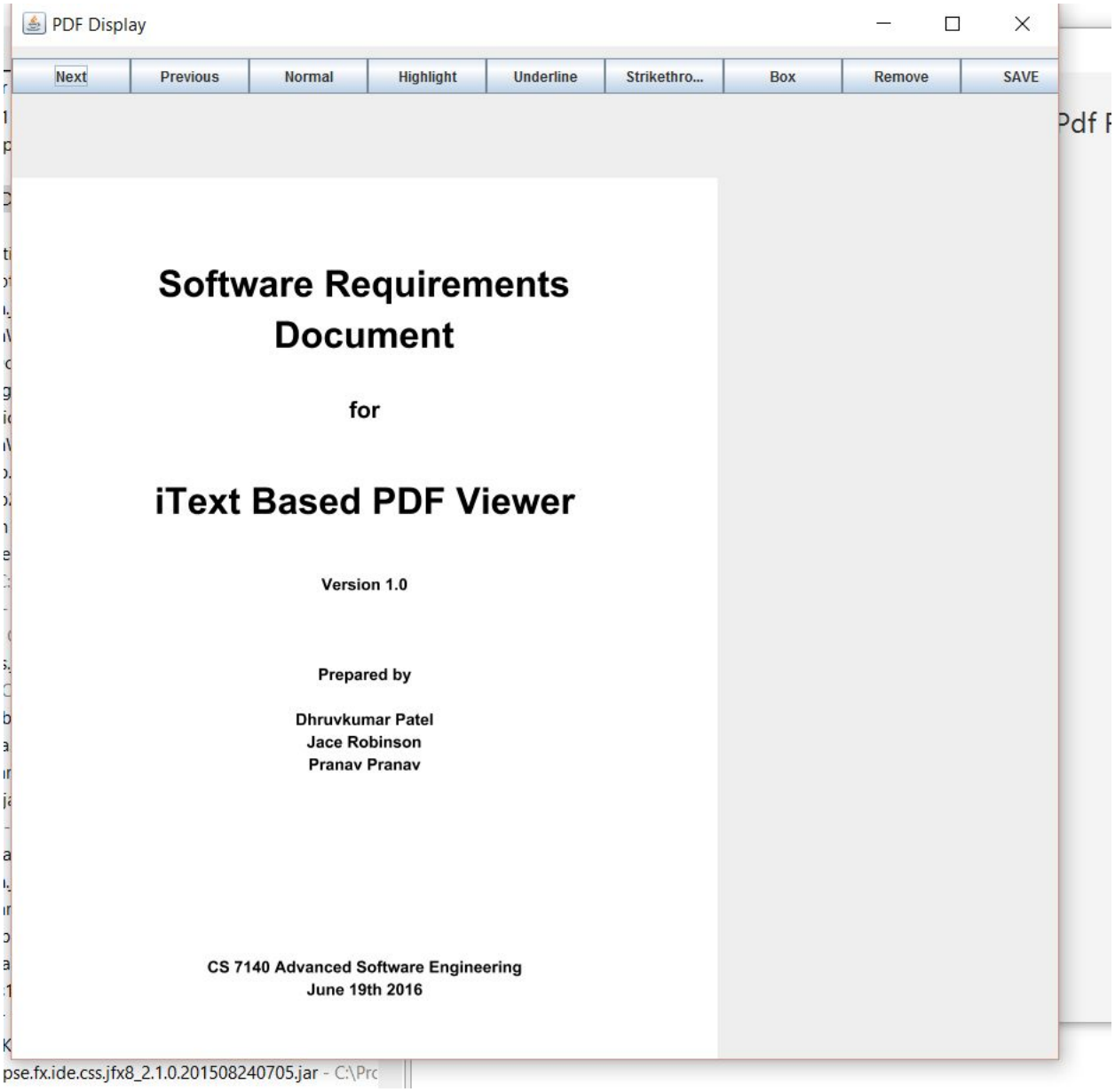
There is a few major known issues within the implementation. First the viewer only displays the textual content of a pdf, and does not display the annotations. As a result, when the user is adding and removing annotations, the content is not displayed live. Despite this issue, the annotations are edited in the final output file. Next the “underline” and “strikethrough” annotations are not created properly. The annotations are added but simply displayed as blue and red rectangles respectively.

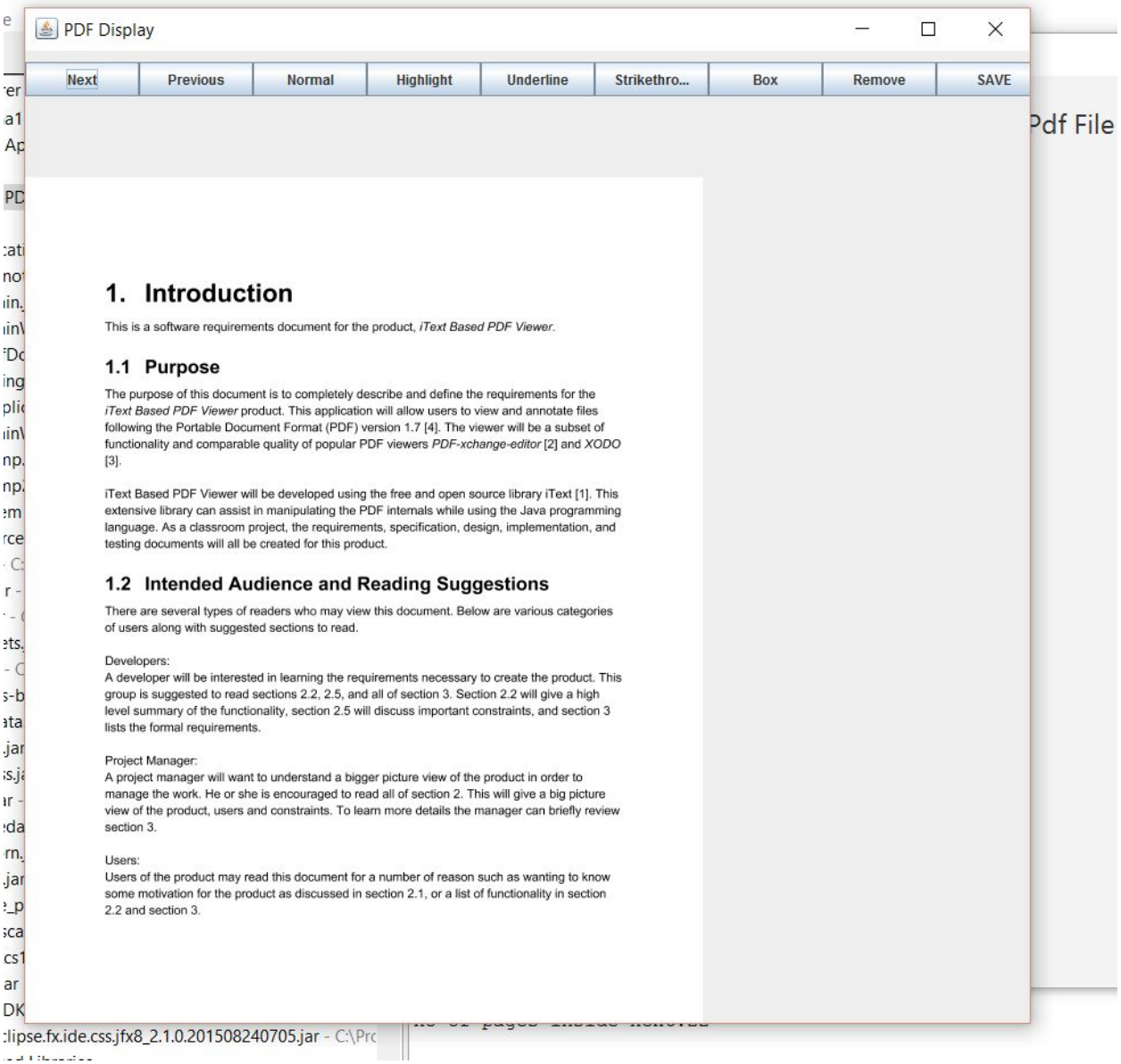
Within the smoke test screen shot descriptions are instructions on how to interact with the GUI.

Initial screen: User is to select a file using browse local file button, then select upload file.

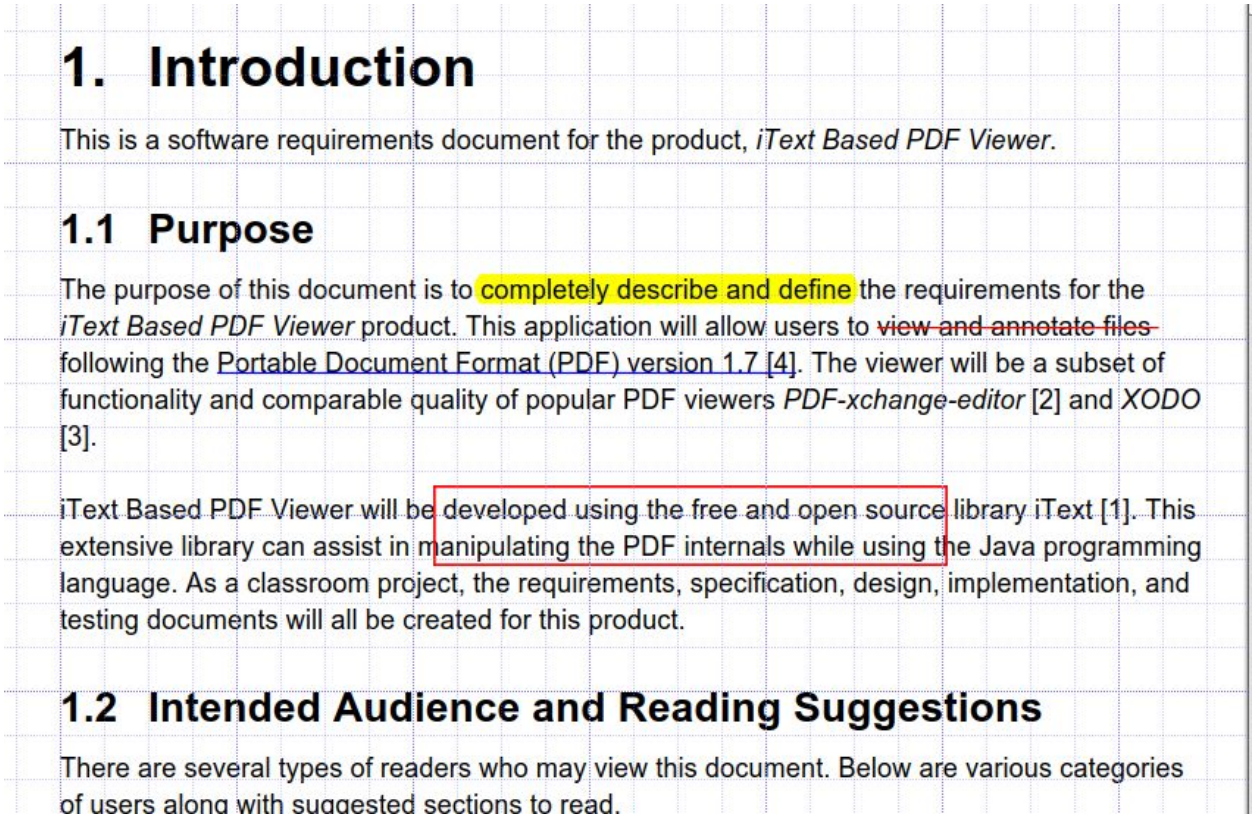


Viewer: This will bring up the viewer. We can see there are many buttons along the top. Next and previous navigate through the pdf. The mouse cursor will register clicks. When in normal mode (activated by default or clicking normal button), the user clicks will do nothing. When the highlight button is pressed, two user clicks are used to create an annotation rectangle. The first click gives the upper left corner (or lower left), and the second click gives the lower right (upper right). These annotations are not displayed to the screen, but they will be written to the output file. Click each type of annotation button will change which type of annotation is created. The delete button will cause the user clicks to delete an annotation which contains the cursor location. This functionality is able to remove existing annotations and newly added annotations. Lastly the save button prompts user to select save location.


PDF Display
Next
Previous
Normal
Highlight
Underline
Strikethro...
Box
Remove
SAVE
Software Requirements Document
for
iText Based PDF Viewer
Version 1.0
Prepared by
Dhruvkumar Patel
Jace Robinson
Pranav Pranav
CS 7140 Advanced Software Engineering
June 19th 2016



Before remove: Here we can see a screenshot of the previously added annotations by PDF-Exchange. Note that these next few screen shots are all taken from the PDF-Exchange editor and not from our viewer.



1. Introduction

This is a software requirements document for the product, *iText Based PDF Viewer*.

1.1 Purpose

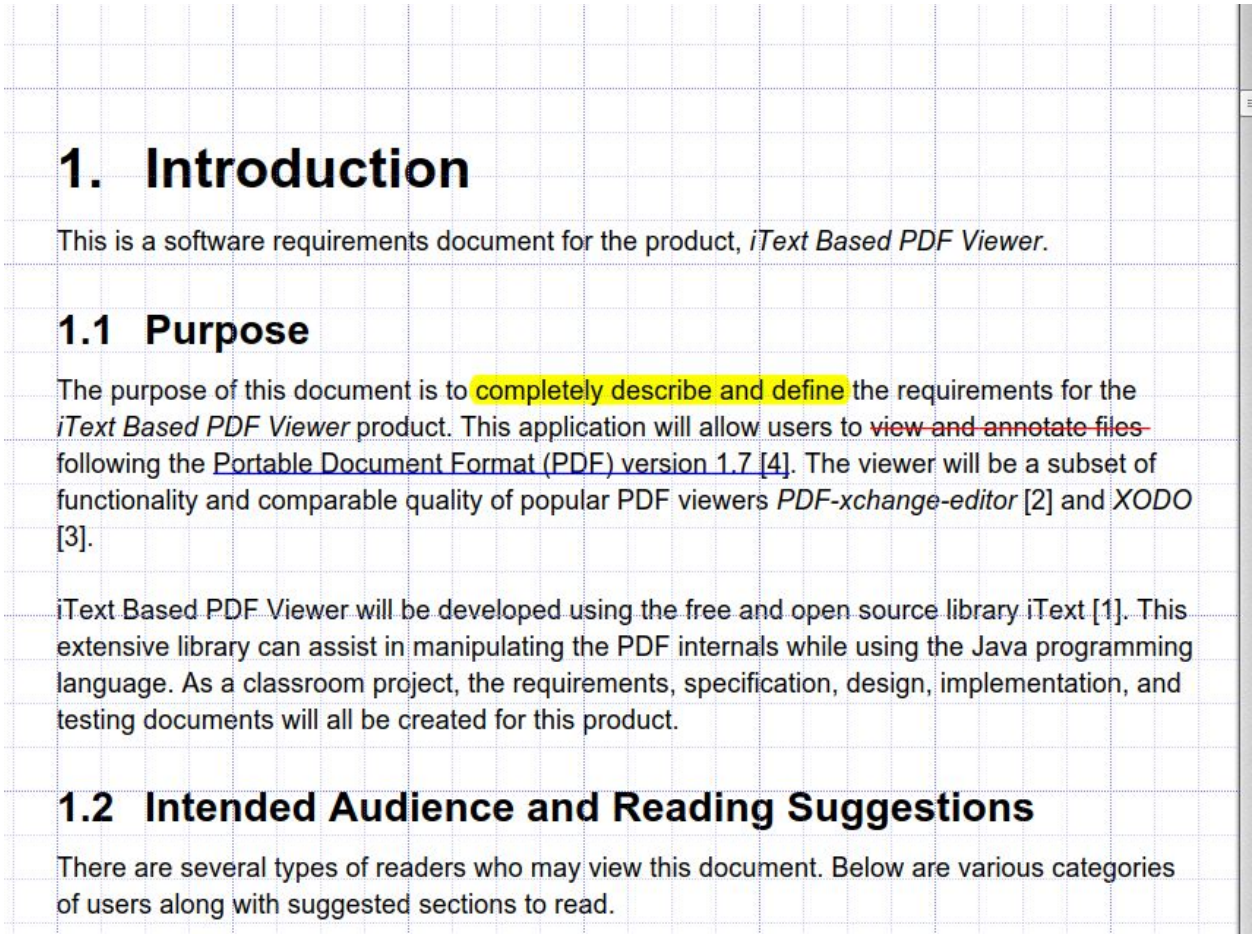
The purpose of this document is to completely describe and define the requirements for the *iText Based PDF Viewer* product. This application will allow users to ~~view and annotate files~~ following the Portable Document Format (PDF) version 1.7 [4]. The viewer will be a subset of functionality and comparable quality of popular PDF viewers *PDF-xchange-editor* [2] and *XODO* [3].

iText Based PDF Viewer will be developed using the free and open source library iText [1]. This extensive library can assist in manipulating the PDF internals while using the Java programming language. As a classroom project, the requirements, specification, design, implementation, and testing documents will all be created for this product.

1.2 Intended Audience and Reading Suggestions

There are several types of readers who may view this document. Below are various categories of users along with suggested sections to read.

After remove: We can see the red box in the bottom portion of the image has been removed.



1. Introduction

This is a software requirements document for the product, *iText Based PDF Viewer*.

1.1 Purpose

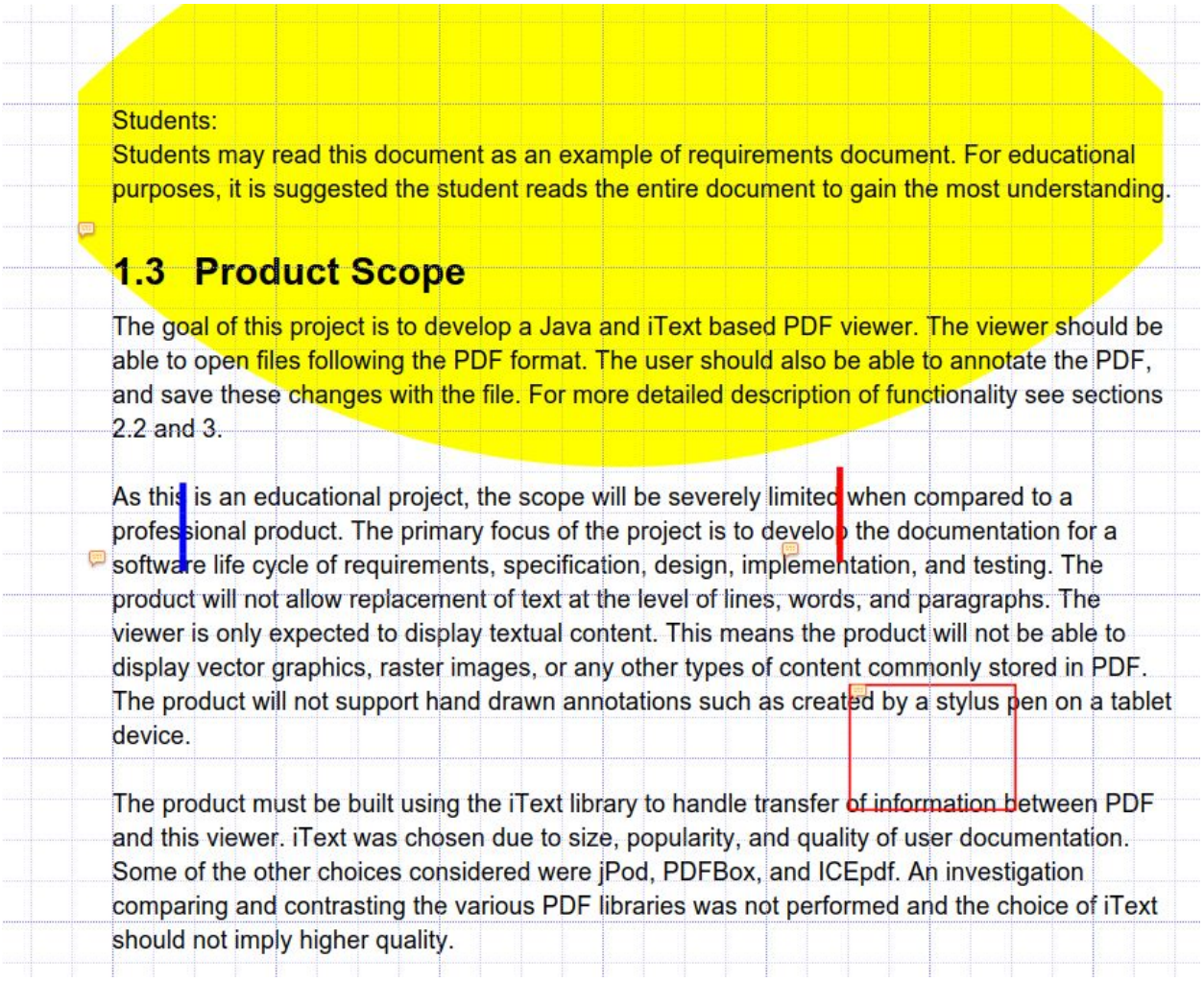
The purpose of this document is to completely describe and define the requirements for the *iText Based PDF Viewer* product. This application will allow users to ~~view and annotate files~~ following the Portable Document Format (PDF) version 1.7 [4]. The viewer will be a subset of functionality and comparable quality of popular PDF viewers *PDF-xchange-editor* [2] and *XODO* [3].

iText Based PDF Viewer will be developed using the free and open source library iText [1]. This extensive library can assist in manipulating the PDF internals while using the Java programming language. As a classroom project, the requirements, specification, design, implementation, and testing documents will all be created for this product.

1.2 Intended Audience and Reading Suggestions

There are several types of readers who may view this document. Below are various categories of users along with suggested sections to read.

Lastly we can see the result of the annotations added using our viewer. As you can see, the underline and strikethrough annotations are given at the location of the user clicks but not displayed properly.



Students:
Students may read this document as an example of requirements document. For educational purposes, it is suggested the student reads the entire document to gain the most understanding.

1.3 Product Scope

The goal of this project is to develop a Java and iText based PDF viewer. The viewer should be able to open files following the PDF format. The user should also be able to annotate the PDF, and save these changes with the file. For more detailed description of functionality see sections 2.2 and 3.

As this is an educational project, the scope will be severely limited when compared to a professional product. The primary focus of the project is to develop the documentation for a software life cycle of requirements, specification, design, implementation, and testing. The product will not allow replacement of text at the level of lines, words, and paragraphs. The viewer is only expected to display textual content. This means the product will not be able to display vector graphics, raster images, or any other types of content commonly stored in PDF. The product will not support hand drawn annotations such as created by a stylus pen on a tablet device.

The product must be built using the iText library to handle transfer of information between PDF and this viewer. iText was chosen due to size, popularity, and quality of user documentation. Some of the other choices considered were jPod, PDFBox, and ICEpdf. An investigation comparing and contrasting the various PDF libraries was not performed and the choice of iText should not imply higher quality.

Journals

See testing document for team journals.

PDF_Viewer

Generated by Doxygen 1.8.11

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 application.Annotations Class Reference

Static Public Member Functions

- static boolean **addHighlightAnnotation** (PdfDocument myDocument, float x, float y, float width, float height, int pageNum)
- static boolean **addUnderlineAnnotation** (PdfDocument myDocument, float x, float y, float width, float height, int pageNum)
- static boolean **addStrikeThroughAnnotation** (PdfDocument myDocument, float x, float y, float width, float height, int pageNum)
- static boolean **addBoxAnnotation** (PdfDocument myDocument, float x, float y, float width, float height, int pageNum)
- static void **printAnnots** (PdfDocument myDocument, int pageNum)
- static void **deleteAnnot** (PdfDocument myDocument, int pageNum, float x, float y)

3.1.1 Detailed Description

This class will add and remove ANNOTATIONS using iText. The (x,y) locations of the annotations are to be given by the caller.

Author

Jace, Dhruv, Pranav

3.1.2 Member Function Documentation

3.1.2.1 static boolean application.Annotations.addBoxAnnotation (PdfDocument *myDocument,* float *x,* float *y,* float *width,* float *height,* int *pageNum*) [static]

The four coordinates of a rectangle are upper left, lower left, upper right, lower right, which correspond to coordinate pairs (x,y+height), (x,y), (x+width, y+height), (x+width, y)

Parameters

myDocument	
x	lower left x coordinate of a rectangle
y	lower left y coordinate of a rectangle
width	of rectangle
height	of rectangle
pageNum	

Returns

3.1.2.2 static boolean application.Annotations.addHighlightAnnotation (PdfDocument *myDocument,* float *x,* float *y,* float *width,* float *height,* int *pageNum*) [static]

The four coordinates of a rectangle are upper left, lower left, upper right, lower right, which correspond to coordinate pairs (x,y+height), (x,y), (x+width, y+height), (x+width, y)

Parameters

myDocument	
x	lower left x coordinate of a rectangle
y	lower left y coordinate of a rectangle
width	of rectangle
height	of rectangle
pageNum	

Returns

3.1.2.3 static boolean application.Annotations.addStrikeThroughAnnotation (PdfDocument *myDocument,* float *x,* float *y,* float *width,* float *height,* int *pageNum*) [static]

The four coordinates of a rectangle are upper left, lower left, upper right, lower right, which correspond to coordinate pairs (x,y+height), (x,y), (x+width, y+height), (x+width, y)

Parameters

myDocument	
x	lower left x coordinate of a rectangle
y	lower left y coordinate of a rectangle
width	of rectangle
height	of rectangle
pageNum	

Returns

3.1.2.4 static boolean application.Annotations.addUnderlineAnnotation (PdfDocument *myDocument,* float *x,* float *y,* float *width,* float *height,* int *pageNum*) [static]

The four coordinates of a rectangle are upper left, lower left, upper right, lower right, which correspond to coordinate pairs (x,y+height), (x,y), (x+width, y+height), (x+width, y)

Parameters

myDocument	
x	lower left x coordinate of a rectangle
y	lower left y coordinate of a rectangle
width	of rectangle
height	of rectangle
pageNum	

Returns

3.1.2.5 static void application.Annotations.deleteAnnot (PdfDocument *myDocument,* int *pageNum,* float *x,* float *y*) [static]

Delete an annotation which contains (x,y) if it exists. Otherwise do nothing. If there are overlapping annotations, the first annotation in the list is deleted.

Parameters

myDocument	!= null
pageNum	> 0 and <= numPages
x	>= 0 and x <= pageSize.x
y	>= 0 and y <= pageSize.y

3.1.2.6 static void application.Annotations.printAnnots (PdfDocument *myDocument,* int *pageNum*) [static]

Display annotations at pageNum to the screen.

Parameters

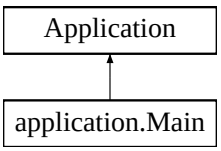
myDocument	!= null
pageNum	> 0 and pageNum <= numPages

The documentation for this class was generated from the following file:

- src/application/Annotations.java

3.2 application.Main Class Reference

Inheritance diagram for application.Main:



Public Member Functions

- void **start** (Stage primaryStage)

Static Public Member Functions

- static void **main** (String[ ] args)

3.2.1 Detailed Description

This PDFViewer will display the text portion of a pdf to the screen. The user has the ability to add and remove annotations, but they are not visible in the current viewer. The annotations can be saved and viewer in a separate viewer such as xodo.

Author

Jace, Dhruv, and Pranav

The documentation for this class was generated from the following file:

- src/application/Main.java

3.3 application.MainWindowController Class Reference

Public Member Functions

- void **browseButtonClick** (ActionEvent e)
- void **uploadButtonClick** (ActionEvent e) throws FileNotFoundException, IOException

3.3.1 Detailed Description

This class is the controller for the main windows. There will be two windows. The first window prompts the user to upload a pdf file. The second window displays the pdf to the screen and gives the user several options to annotate.

Author

Jace, Dhruv, and Pranav

3.3.2 Member Function Documentation

3.3.2.1 void application.MainWindowController.browseButtonClick (ActionEvent *e*)

Button to allow the user to select a PDF file. If the pdf file is valid, prepare it to be viewed when the uploadButton is selected.

Parameters

e	

3.3.2.2 void application.MainWindowController.uploadButtonClick (ActionEvent *e*) throws FileNotFoundException, IOException

The upload button must be selected AFTER the browse button has selected a PDF. If a pdf file, the file will then be displayed in the viewer. Otherwise the user will be prompted to upload a valid file. The method opens two temporary files, one for displaying, one for editing annotations.

Parameters

e	

Exceptions

FileNotFoundException	
IOException	

The documentation for this class was generated from the following file:

- src/application/MainWindowController.java

3.4 application.PdfDocumentcreation Class Reference

Static Public Member Functions

- static String **createFromOld** (String source) throws FileNotFoundException, IOException

- static boolean **addAnnotation** ()
- static String **openPDF** (String source) throws IOException
- static boolean **savePDF** ()
- static PdfDocument **getDocument** ()
- static void **reOpenDocument** () throws IOException

Static Public Attributes

- static final String **tempdest** = "src/application/temp.pdf"
- static final String **tempdest2** = "src/application/temp2.pdf"

3.4.1 Detailed Description

This class will handle all document creation functions. There are two "temp" file locations, one for the static pdf to be displayed, and a second for the annotated pdf.

Author

Jace, Dhruv, Pranav

3.4.2 Member Function Documentation

3.4.2.1 static boolean application.PdfDocumentcreation.addAnnotation ( ) [static]

Testing function for adding, printing, and deleting annotation.

Returns

3.4.2.2 static String application.PdfDocumentcreation.createFromOld (String *source*) throws FileNotFoundException, IOException [static]

Creates pdf file at tempdest. This is the file to be read by the viewer. The file is closed at the end of this function.

Parameters

source	!= null

Returns

Exceptions

FileNotFoundException	
IOException	

3.4.2.3 static PdfDocument application.PdfDocumentcreation.getDocument ( ) [static]

return reference to document

Returns

3.4.2.4 static String application.PdfDocumentcreation.openPDF (String *source*) throws IOException [static]

Open a PDF at source file location. The document remains open after the call of this function.

Parameters

source	

Returns

Exceptions

IOException	

3.4.2.5 static void application.PdfDocumentcreation.reOpenDocument ( ) throws IOException [static]

reopen same document if closed to save.

Exceptions

IOException	

3.4.2.6 static boolean application.PdfDocumentcreation.savePDF ( ) [static]

This function will save and close the current document.

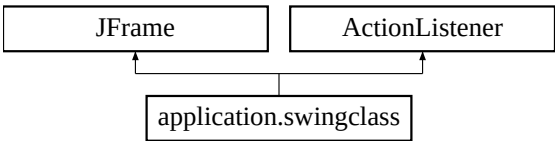
Returns

The documentation for this class was generated from the following file:

- src/application/PdfDocumentcreation.java

3.5 application.swingclass Class Reference

Inheritance diagram for application.swingclass:



Public Member Functions

- **swingclass** (String filename) throws IOException
- void **setup** (String filename) throws IOException
- void **actionPerformed** (ActionEvent e)

3.5.1 Detailed Description

This is the viewer for the PDF. The viewer is from PDF-Renderer found here `https://java.net/projects/pdf-renderer`. The renderer does NOT seem to display annotations, but only text. The user is able to navigate pages, and add/remove annotations based on mouse clicks.

Author

Jace, Dhruv, Pranav

3.5.2 Constructor & Destructor Documentation

3.5.2.1 application.swingclass.swingclass (String *filename*) throws IOException

Initilize viewer with filename pdf.

Parameters

filename	

Exceptions

IOException	

3.5.3 Member Function Documentation

3.5.3.1 void application.swingclass.actionPerformed (ActionEvent *e*)

The various button clicks. The next and previous buttons navigate the pdf. The various buttons change the "mode" of the cursor to add/remove/do nothing for annotations.

3.5.3.2 void application.swingclass.setup (String *filename*) throws IOException

Add all of the buttons and pdf viewer component to JPanel.

Parameters

filename	

Exceptions

IOException	

The mouse event tracks the users clicks on the panel. Every two clicks are used to create a rectangle, which determines where the annotations are created or deleted.

The documentation for this class was generated from the following file:

- src/application/swingclass.java

3.6 Tests.TestRunner Class Reference

Static Public Member Functions

- static void **main** (String[ ] args)

The documentation for this class was generated from the following file:

- src/Tests/TestRunner.java

3.7 Tests.UnitTests Class Reference

Public Member Functions

- boolean **checkInputFile** (String filename)
- boolean **checkFileSave** (String fileToSave)
- void **testInputFile** ()
- void **testFileSave** ()
- void **testAddUnderlineAnnotation** () throws IOException
- void **testAddHighlightAnnotation** () throws IOException
- void **testAddStrikeThroughAnnotation** () throws IOException
- void **testAddBoxAnnotation** () throws IOException
- void **testDeleteAnnot** () throws IOException
- boolean **testDeleteAnnot** (PdfDocument myDocument, float x, float y, float width, float height, int pageNum)

The documentation for this class was generated from the following file:

- src/Tests/UnitTests.java

Index