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

Table of Content

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.

Developers:
A developer will be interested in learning the requirements necessary to create the product. This group is suggested to read sections 2.2, 2.5, and all of section 3. Section 2.2 will give a high level summary of the functionality, section 2.5 will discuss important constraints, and section 3 lists the formal requirements.

Project Manager:
A project manager will want to understand a bigger picture view of the product in order to manage the work. He or she is encouraged to read all of section 2. This will give a big picture view of the product, users and constraints. To learn more details the manager can briefly review section 3.

Users:
Users of the product may read this document for a number of reason such as wanting to know some motivation for the product as discussed in section 2.1, or a list of functionality in section 2.2 and section 3.

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.

1.4 References

[1] "iText Developers", *Developers.itextPDF.com*, 2016. [Online]. Available: http://developers.itextPDF.com/. [Accessed: 15- Jun- 2016].

[2] "Tracker Software Products :: PDF-XChange Editor", *Tracker-software.com*, 2016. [Online]. Available: https://www.tracker-software.com/product/PDF-xchange-editor. [Accessed:15- Jun- 2016].

[3] "XODO PDF Reader & Annotator", *Xodo.com*, 2016. [Online]. Available: http://xodo.com/. [Accessed: 15- Jun- 2016].

[4] *PDF Reference*, 6th ed. Adobe Systems Incorporated, 2006.

[5] "Doxygen: Main Page", *Stack.nl*, 2016. [Online]. Available: http://www.stack.nl/~dimitri/doxygen/. [Accessed: 16- Jun- 2016].

1.5 Document Conventions

“PDF File” - For the purposes of this Requirements Document, “PDF file” refers to a file format in compliance with the standards defined in the PDF reference document version 1.7 [4].

1.6 Definitions, Acronyms, and Abbreviations

JAR	Java Archive
JDK	Java Development Kit
PDF	Portable Document Format
UI	User Interface

1.7 Figures

2. Overall Description

2.1 Product Perspective

As discussed, in section 1.3, the goal of this project is to develop a Java and iText based PDF viewer where the user should also be able to annotate the PDF, and save these changes with the file. And as discussed, in section 1.1, the project will be a subset of functionality and comparable quality of popular PDF viewers *PDF-xchange-editor* [2] and *XODO* [3].

As a class project, will be limited to a proof of concept. It will be written in Java 8 and JavaFX will be used for its front end. Exploiting the iText library, the ability of extracting texts and adding annotations will be required. See sections 2.2 and 3 for more details.

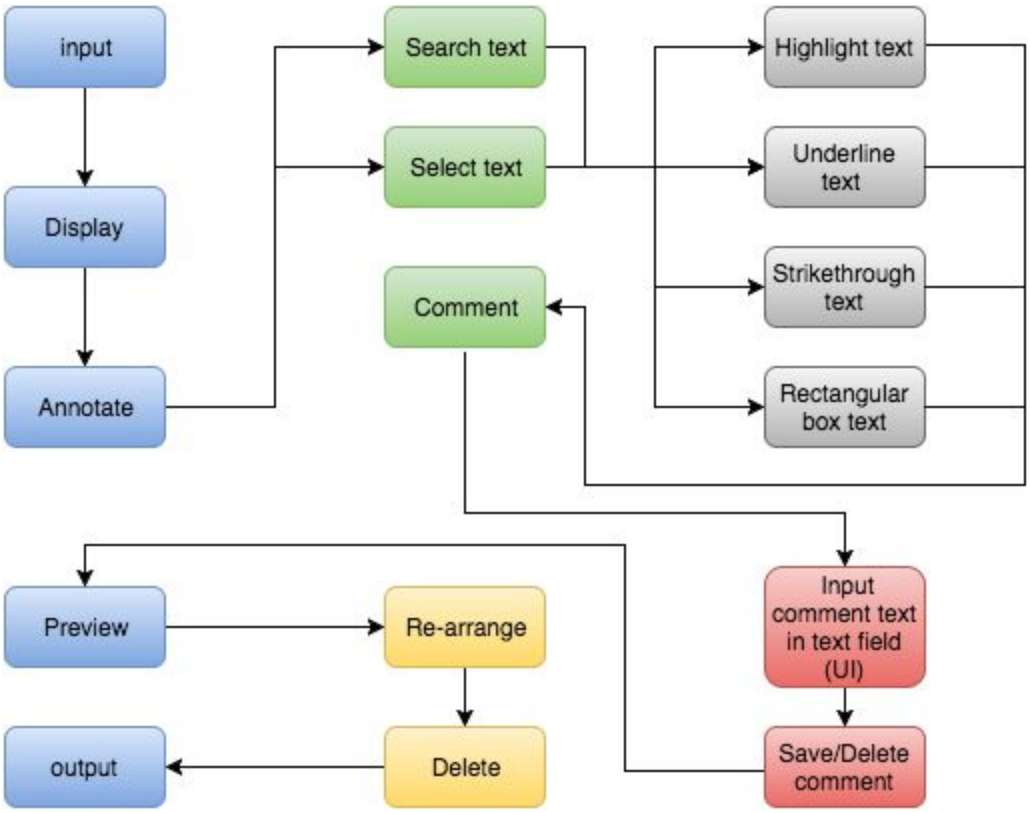

Figure 2.1.1: Work flow of the product.

As seen in figure 2.2.1, all blue color boxes are the basic functionalities of the product. Input, display, annotation, preview and output are discussed in more detail in following sections [2.2]. All green color boxes are sub functionality of the “Annotation”. All grey color boxes are different options for “Select text”. All red color boxes are sub functionality of the “Comment”. All yellow color boxes are sub functionality of the “Preview”.

2.2 Product Functions

Following are the functionalities given by the product

1. Input: User Interface (UI) which accepts input as uploaded file defined by PDF [4].
2. Display: UI to display the uploaded PDF in a way that content of the given PDF are not editable at the level of lines, words, or paragraphs, but to be viewed as same. Further the display will be used to annotate text as well.
3. Annotate: UI in the display will be used to annotate text in a number of ways and operations performed given below.
    a. Select text: There are four ways to select text.
        i. Text Highlighting: Mouse cursor will be provided to select and highlight the text

        ii. Under lining: Mouse cursor will be provided to select and underline the text
        iii. Strikethrough: Mouse cursor will be provided to select and strikethrough the text
        iv. Rectangular Box: Mouse cursor will be provided to select and box the text
    b. Comment: operations can be done as part of the annotation to the selected text.
        i. Open text field for keyboard input which can be save or delete. No edit strictly. UI should have fresh look and feel.
        ii. Save or delete the comment.
    c. Search text: operations will be created to allow user to search for specific text and display the location of matching text within the document.
4. Preview: An option provided to user to do below operations.
    a. Rearrange: the user will able to re-arrange all the pages of the given PDF pages.
    b. Delete: the user will able to delete the pages he/she wish to. Comments will be also deleted with the deleted pages if any.
5. Output: Save the project including the given preview page(s) and associated comment into a PDF [4] file as output.

2.3 User Classes and Characteristics

There are two main groups of users expected to use the product.

Professional PDF User:
A professional PDF user is interested in most to all of the product's functions. Users from this group are likely to work with PDF's very frequently, and make use of the ability to add and remove annotations, and search for specific text locations. This is the most important audience for the product.

Novice PDF User:
A novice PDF user is only interested in a subset of the products function. Specifically, this user group will only be interested in viewing PDFs, and will not add or remove annotations or search for text. This is a lesser important audience for the product.

2.4 Operating Environment

The application will be developed to work on up-to-date software at the time of the creation of this document. As the software will be built using Java and delivered as a .jar file, to run the software a user must have the Java Runtime Environment (JRE) version 1.8.0_91 or higher installed. The application will be usable with Windows 10 version 1511, and Ubuntu version 14.04.

2.5 Design and Implementation Constraints

There are several constraints affecting the development of this product. The constraints are listed below.

1) The product must be developed in six weeks. Due to this short time frame, the scope has been severely reduced.
2) The software development life cycle will follow a waterfall model. This model has the sequence of deliverables of requirements, specifications, design, implementation, and testing.

2.6 User Documentation

The product will be delivered with two manuals.

1) The first manual will describe the source code. This is intended for developers and not for users. The document will be created using *Doxygen* to describe the source code and display class diagrams [5].
2) A second document will be created to describe all of the features available to the user. This non-technical document will provide written tutorials on how to open a PDF, search for text, create annotations, and save PDF.

2.7 Assumptions and Dependencies

There are a few critical dependencies for this product that are listed below.

1) The viewer is developed to display PDF version 1.7 formatted files [4]. It is assumed that future versions of PDF will remain backwards compatible with this version.
2) The viewer is developed using free and open source library developed by iText. It is assumed the library remains free in the future.
3) The viewer is to be developed using JavaFX available in Java 8. It is assumed current and future JRE's past version 1.8.0_91 will be backwards compatible and execute the deliverable .jar file.
4) As a test to the saving and uploading of PDF documents, the PDF viewer XODO [3] will be used. A requirement defined in section 3 is a PDF file created in this product must be viewable in XODO. It is assumed that XODO will continue to be available during the development process. It is also assumed XODO will continue to follow PDF standards.

3. Specific Requirements

Within this section contains a numbered list of all requirements of the product. Numerous definitions of words used in the requirements are given immediately below.

"text" - For the purposes of this Requirements Document, "text" refers to "Text Objects" described by Section 5.3 of the PDF reference document [4] with the added restriction that ”text" is composed solely of sequences of Latin characters irrespective of font specified in the PDF for display of the characters. Latin characters are defined in Appendix D.1 of the PDF reference [4].

"annotation" - refers to a subset of annotations described in section 8.4 of the PDF reference document [4]. The subset chosen are **underline, highlight, strikethrough, rectangular box, and comment**. Table 8.28 of the PDF reference [4] describes rectangular box, and table 8.30 of the PDF reference [4] describe underline, highlight and strikethrough, and table 8.23 describes comment. A visual example of each annotation is given in figure 3.2.1 and figure 3.2.2 of this document.

“upload” - refers to the display of only text and annotations encoded within PDF documents selected from within a file system by a user. The text will display to the output monitor device of the computer.

“save” - refers to storing PDF documents with user selected file names at user selected file system locations.

“insert” - “insert” refers to adding content to the current state of the PDF document.

“delete” - refers to removing content from the current state of the PDF document.

“location” - refers to the position in the grid defined in section 4.2 of the PDF reference document [4].

“search” - refers to matching input text with all occurrences of text contained in the current page of document such that textInput = textDocument. The locations of the matching text within the document are determined.

“PDF page” - refers to a grouping of PDF content as defined in section 3.6 of PDF reference document [4].

Following are the numbered requirements for iText Based PDF Viewer.

3.1 External Interfaces

Following are the **User Interface** requirements for this PDF Viewer.

3101: Facility to upload an existing PDF file.
3102: Display both text and annotation content of a single page of PDF file into an output device such as monitor.
3103: Facility to change current content of display from one page of PDF file to a different page.
3104: Facility to search text.
3105: Facility to save PDF under current filename.
3106: Facility to insert an annotation.
3107: Facility to delete an annotation.
3108: Facility to delete page including the annotations on the page.
3109: Facility for highlighting annotation
3110: Facility for underlining annotation
3111: Facility for strikethrough annotation
3112: Facility for creating rectangular box annotation
3113: Facility for comment annotation

Following are the **Software Interface** required for the PDF Viewer.

3114: The product shall use iText 7 library to read PDF file.
3115: The product shall use iText 7 library to save PDF file.

3.2 Functional Requirements

Following are the major functionalities of iText Based PDF Viewer.

3201: The product shall upload an existing PDF file
3202: The product shall search particular text
3203: The product shall display location(s) of searched text
3204: The product shall do following on annotations:
- **3204.1**: Insert an annotation
- **3204.2**: Delete an annotation
- **3204.3**: Multiple annotations can appear on single PDF page
- **3204.4**: Different types of text selection annotations are available as in figure 3.2.1:
  - **3204.4.1**: Text Highlighting in color yellow
  - **3204.4.2**: Underlining in color blue
  - **3204.4.3**: Strike-through in color red
  - **3204.4.4**: Rectangular Box in color red

3204.5: Comment annotation as in figure 3.2.2

3204.5.1: Have insert text in the comment annotation

3204.5.2: The text of comment annotation can be deleted.

3205: The product shall allow rearranging order of pages of the PDF file

3206: The product shall allow deletion of page(s) of PDF file

3207: If uploaded file PDF file, then the saved file must must be PDF file

3208: Annotations inserted and saved in this product must be viewable in PDF viewer XODO

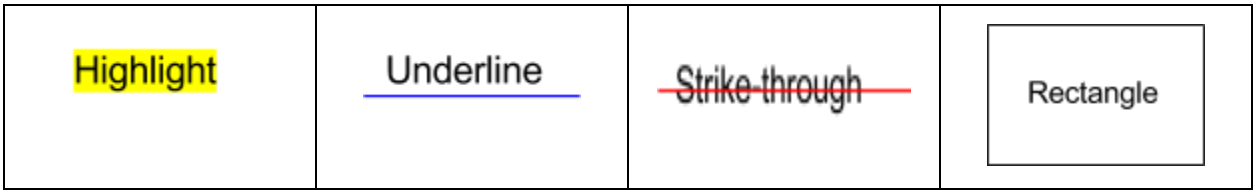

Figure 3.2.1

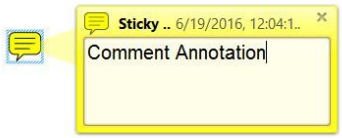

Figure 3.2.2

3.3 Performance

Following are the performance requirements for iText Based PDF Viewer.

3301: 95% of the time the product must display text content of compliant PDF file within 10 seconds of upload.

3302: 95% of the time the product must display text and annotation content of compliant PDF file within 15 seconds of upload.

3303: 95% of the time the product must display inserted annotation within 5 seconds to output monitor.

3.4 Software System Attributes

In order to guarantee a usable product for customers, the following attribute requirements are listed.

Reliability:

3401: 99% of the time the product shall not crash or hang after one hour of continuous usage.
3402: 99% of the time the product shall not crash when uploading a non-PDF compliant document

Portability:
3403: The product shall satisfy all requirements of this document in both Windows 10 and Ubuntu with versions discussed in section 2.4.

3.5 Additional Requirements

There are a few additional requirements imposed upon the developers.

3501: Code maintainability expectations
Well designed software should be maintainable. There are several techniques that will be followed in the creation of this product. First, requirements, specifications, and design will be created before implementation begins. Second the developers will follow the SOLID principles of object oriented programming. Third the programmers will be following Design by Contract. 80% of methods must have a pre and post condition, along with class invariants and loop invariants.

3502: Testing requirements
As the developers are creating maintainable code through following the principles described before, this code is also becoming much more testable. The design by contract principles will enforce assertions to ensure certain conditions are maintained throughout the development. All requirements listed in this document can be tested as boolean success or failure through acceptance testing. Unit testing on several classes will also be performed.

3503: Code reduction
In addition to the viewer, there will be the opportunity to seek improvement in iText code. The developers will examine a subset of the total functions available and look to reduce the code size by 5%. The development of pre and post conditions on many functions may reveal optimization opportunities.

4. Author Journals

Below are the journals of each of the authors maintained throughout the development process.

4.1 Dhruvkumar Patel

Start date : 5-15-16

Professor described Project and we all ready on iText Based Pdf Viewer and decided to implement in Java Programming Language.

End Date: 5-15-16
Start Date :5-25-16

I installed iText7 Pdf library in my eclipse and start maven project to simple create a pdf file and tried to insert image, Text, annotation into pdf using iText7 Tutorials.

End Date : 5-25-16

Start Day 5-31-16

Today I joined github Project Repository which is created by my group member and experimenting with iText
using Eclipse IDE.

I have successfully synced project with Eclipse and imported iText libraries.

Successfully implemented initial example what i tried earlier.

Also i used PdfReader functionality from iText 7 Tutorials.

End Day 5-31-16

Start Day 6-5-16

Today I am trying to define initial Requirements and Specification of iText Based Pdf Viewer.

I read Pdf Documents for few topics then i wrote following requirements and specification on Piazza.

Requirements)

Requirements :- There are Functional, Non Functional and User Interface Requirements for our PDF-Project.

Functional requirements: 1) user can Read PDF file using this Tool.
2) Display Existing PDF by pages.
3) Search particular word from PDF file and display the cursor to that particular Word.
4) Display all characters, images, etc.
5) user can able to Add,Delete and Edit Annotations.

Non-Functional Requirements : 1) Performance
2) Accessibility
3) Flexibility

Specification :- 1) Design PDF tool which provides Reading, Split a PDF by pages,Edit Annotations, Searching Features.
2) Accuracy is important while extracting a Text.
3) use iText7 library in java to develop all the functionalities.
4) Java Programming language is necessary
5) use JavaFx or Java Swing for GUI part.

End Day 6-5-16

Start Day 6-8-16

Today I am meeting with Jace and Pranav.

First task is defining high level specifications for the project.

Input: PDF compliant document

Output: PDF compliant documentation

Input-Output: PDF + annotation, input = output

End Day 6-8-16

Start Day 6-14-16

Today I am meeting with Jace and Pranav.

So now we decided to work on Only Requirements Rather than Specification.

we found a template suggested by IEEE in section 5 here
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574
and all three decide to write requirements using this Format of the document.

we discuss our all three's requirements what we wrote on Piazza and tried to make one final Draft of Requirements by points.

we used Google Docs to wrote this document so anyone can write and access document any time.

we divide the sections of the document and commit to complete by next meeting.

End Day 6-14-16

start date 6-15-16

Today i wrote my section of portion which contains section 3 in document covered specific Requirements Likes External interface, Functional Requirement, Performance, Software System Attributes and few additional Requirements.

end date 6-15-16

start date 6-16-16

Today we met with the whole document content. So we discussed on each and every section and finalized Report. Also we added Annotations Figure which i generated by program using iText 7 library Tutorials. Also we add Workflow Diagram Figure.

Avoid any errors we just decide to Read through the whole document on saturday and decided to submit on saturday night.

end date 6-16-16

4.2 Jace Robinson

5-15-16
Initial Research into PDF format…

Start Day 5-31-16
Today I will be setting up a project on github, using intellij IDE, and experimenting with iText.

I have successfully synced project with intelli and imported iText libraries.

Successfully ran initial example that writes to PDF file.

I skimmed the remaining chapters of the itext tutorial, it seems worthwhile to read through when I have time.

End Day 5-31-16

Start Day 6-5-16
Today I am creating initial requirements and specifications for PDF project.

These documents should cover both iText and our product.

Requirements)

Opening
Open an existing PDF following PDF Spec version 1.7.
Open an existing PDF that slightly deviates from PDF Spec version 1.7*** (currently not sure how to define slightly deviating)
Create a new blank PDF.

Saving
Save a modified PDF file under current filename.
Save a PDF file to separate filename specified by user.
Save subset of pages selected by user to separate PDF file with filename specified by user.

Edit and Searching
Find location(s) of text matching a key text provided by user.
Highlight the text matching the key text and call this the MATCHED_TEXT.
Allow selection of text with cursor, this is SELECTED_TEXT.
An annotation consists of a location, MATCHED_TEXT or SELECTED_TEXT in the PDF, a title, and a description.
Allow user to add annotations.
Location of annotation is determined by user cursor location or user input.
Allow user to edit title of annotation.
Allow user to edit description of annotation.
The MATCHED_TEXT or SELECTED_TEXT can be highlighted, underlined, or line-throughed.
Redacting consists of covering a portion of the PDF with a black rectangle such that content in the same location as the rectangle is no longer contained in PDF.
Allow user to redact content in PDF.

Displaying PDF
Allow user to zoom on PDF.
Allow user to rotate a PDF 90 degrees.
Display a continuous scrolling PDF.
Provide ability to navigate by scrolling.
Provide ability to navigate entire page at a time.
User shall be able to view document properties.
Document properties include file size in bytes, location of file, file name, creation date, and latest modification date.

User Interface
Display content of a PDF page to screen.
User has ability to navigate PDF pages using scroll bar.
Text field displaying current page out of total pages.
Button to navigate back a single page.
Button to navigate forward a single page.
Button to zoom in document.
Button to zoom out document.
Slider to adjust zoom level.
Button to enable/disable annotation mode for cursor.
When annotation mode is enabled, the location a user clicks on the PDF will create an empty annotation.
Button to save PDF under current filename.
Button to 'save as' under a different filename.
Button to open PDF.
Sidebar displaying preview of PDF pages.
Sidebar displaying sequential list of all annotations.
Text field to input key text for searching.

Performance
Program must display user interface within 10 seconds on initial execution.
User shall be able to use application in Windows, Linux and OSX with Java SE Runtime Environment version 1.8.0_91 installed.

Robustness
Software does not crash when opening a file of non PDF format.

Documentation
External documentation attempts to explain all user interface GUIs.
Source code contains entry and exit conditions for all functions.

Other (ideas that may or may not be included in requirements based on time constaints)
Allow user to print PDF to connected printer device.
Allow user to email PDF to a specified email address.
Allow user to merge two or more existing PDFs.
Allow user to save PDF directly to a google drive account.
Allow user to encrypt saved PDF.
Allow user to interact with form fields.
Allow user to provide digital signature.

P.S. here is an example of requirements specification I used as reference https://reqview.com/ext/ReqView-SRS_Software_Requirements_Specification_Example.html.

End Day 6-5-16

Start Day 6-8-16

Today I am meeting with Dhruv and Pranav.

First task is defining high level specifications for the project.

Input: PDF compliant document

Output: PDF compliant documentation

Input-Output: PDF + annotation, input = output

End Day 6-8-16

Start Day 6-14-16

Today I am meeting with Dhruv and Pranav.

Our last meeting was not productive as we were looking at specifications instead of requirements, now that we know what to work on, we should be more productive.

My goal by the end of today is to decided on a requirements document template, divide some work among the different sections of document, and discuss some of the other items on the rubric.

I found a template suggested by IEEE in section 5 here
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574

Erik Buck had the highest quality requirements on piazza so we will use his as a reference to start ours. I was also happy with mine although there were too many.

For help with correctly describing technical details such as text, I will be looking in the official PDF reference document.

We have created a google document containing our requirements document. We are using a format very similar to the IEEE template discussed above.

We have divided the document among user, I will take the introduction and some

End Day 6-14-16

6-15-16
Today I am writing the introduction for the requirements document. This includes purpose, scope, audience, and a few other small sections.

6-16-16
Today I am writing some more sections of the document, including all of section 1, sections 2.3-2.7, and section 3.5. So far we have content in all sections initially complete except ‘unique functionality’ as proposed in the rubric. Later today Drhu, Pranav and I will be meeting. I would like to spend some time ensuring the requirements in section 3 are complete and follow the necessary qualities described by Bertrand Meyer. The qualities expressed as the ‘seven sins of specifier’ are…

1) Noise
2) Silence
3) Over-specification
4) Contradiction
5) Ambiguity
6) Forward
7) Wishful thinking

To avoid any issues, we are defining all keywords at the beginning of the requirements section. The document is coming along nicely. **Each individual completed first draft of their section of the work within the desired time frame.** We intend to submit completed document on Saturday night. The current document today is mostly complete and just needs proof read.

4.3 Pranav Pranav

5-15-16
Group formed. Set channel for communication. Discussed for group meeting whenever required.

Start Day 5-31-16

Read itext on wikipedia and browse through PDF reference from adobe.

End Day 5-31-16

Start Day 6-5-16

Joined project group on github.
Today I am creating initial requirements and specifications for PDF project.

End Day 6-5-16

Start Day 6-8-16
Worked on the requirement and specification.

Requirement

Input: PDF document (abiding official definition of PDF by Adobe)

Output: new PDF document (input plus annotation [Clarification: how annotation is required to display in output?]), all_annotation_together.txt (assuming text file/can be PDF)

GUI:

Load Screen,

Display Screen/s (Clarification: depends on all content on the scrollable pane or pageable pane?),

Edit Screen/s (for annotation: - separate text field to capture user input, NO editing of existing documents; with saving feature and proper success or failure messages).

Specification:

FileLoad:

assertion1(fn_FileLoad(args), true);

assertion1(fn_CheckFileIntoDir(args), true);

LoadPDF:

assertion1(fn_LoadPDF(args), true);

assertion1(fn_CheckPDFobject(args), true);

ExtractPDFContent:

assertion1(fn_ExtractPDFContent(args), true);

assertion1(fn_ReadContentPrintToLogOrConsole(args), true); //not sure

PreserveContextFromPDF:

assertion1(fn_PreserveContextFromPDF(args), true);

assertion1(fn_CheckByWritingPDFWithNoEdit(Input_PDF), true); //not sure

CaptureEdit:

assertion1(fn_CaptureEdit(args), true);

assertion1(fn_CheckEditWithPrintToLogOrConsole(args), true);

SaveFile:

assertion1(fn_SaveFileWithEditsAndProperMessage(args), true);

assertion1(fn_CheckFileFromOutputDir(args), true);

We are having our first group meeting today.

Todays discussion is based on course lecture today.

Tried to define high level specification.

End Day 6-8-16

Start Day 6-9-16

As discussed,

1. INPUT

PRE: assertion_pre_input(fn_FileLoad_ExpectFile(File file), true);

POST: assertion_post_input(fn_CheckFile_PDFFile(File file), true); // abide the official definition of PDF (7 bits ASCII file in a COS format.)

2. OUTPUT

PRE: assertion_pre_output(fn_GetObjects_InputPDFObjsAndAnnotationObj(List objs), true); // input PDF is broken down into list of PDF document objects and Annotation objects (type of PDF document)

POST: assertion_post_output(fn_ReturnFile_PDFFile(List Objs), true); // Combining all Objs in proper manner such that the output file is abiding the official definition of PDF (7 bits ASCII file in a COS format.)

3. RELATION

Definition of Input file = Definition of Output file

End Day 6-9-16

Start Day 6-14-16

Second Group meeting.

For our first deliverable we need to produce requirement document not specification document. We pointed out all project requirement that we can figure out from piazza and discussed among us.

Draw figures and debated on requirements. Output was crisp definition of our requirement.

We started filling in the section in the google document created by Jace based on the IEEE format.

We have divided the document among user, I will take the Overall description.

End Day 6-14-16

Start Day 6-15-16

My responsibility is to define section 2. Overall description.

End Day 6-15-16

Start Day 6-16-16

We finished the document. Jace helped me to finish few section of my part in the document.

We are reviewing the document in our group meeting.

Workflow discussed and done with figure.

End Day 6-16-16