A Project Report on

CONVOPRO

Submitted to the Department of Computer Science & Engineering, GNITS in the partial fulfillment of the academic requirement for the award of B.Tech (CSE) under JNTU

By

T. Mouni Priyanka (13251A0558)

Under the guidance of

Mrs. Ch. Mandakini

Assistant Professor, CSE

Department of Computer Science & Engineering

G. Narayanamma Institute of Technology & Science (for Women)

Shaikpet, Hyderabad- 500 104

Affiliated to

Jawaharlal Nehru Technological University

Hyderabad – 500 062

2016

Department of Computer Science & Engineering
G. Narayanamma Institute of Technology & Science (for Women)

Shaikpet, Hyderabad – 500 104

Certificate

This is to certify that the Mini Project report on "**CONVOPRO"** is a bonafide work carried out by **T. Mouni Priyanka (13251A0558)** in the partial fulfillment for the award of B.Tech degree in Computer Science & Engineering, G. Narayanamma Institute of Technology & Science, Shaikpet, Hyderabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad under our guidance and supervision.

The results embodied in the Mini Project have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Mrs. Ch. Mandakini
Assistant professor,
Department of CSE.

Head of the Department
Dr. M. Seetha
Professor and HOD,
Department of CSE.

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr K. Ramesh Reddy, Principal,** GNITS, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. M. Seetha, Head and Professor**, Dept. of CSE, GNITS for all the timely support and valuable suggestions during the period of our mini project.

We are extremely thankful to **Mrs. P. Sunitha Devi, Asst.Prof, Mrs. D. Naga Swetha, Asst.Prof** Dept. of CSE, GNITS, the miniproject coordinators for their encouragement and support throughout the project.

We are extremely thankful and indebted to our internal guide**, Mrs. Ch. Mandakini, Asst.Prof**, Dept. of CSE, GNITS for her constant guidance, continuous advice, encouragement and moral support throughout the mini project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly, parents and friends for their cooperation in completing the mini project work.

T. Mouni Priyanka (13251A0558)

Contents

Abstract

Modern hand held devices such as smart phones and personal digital assistant (PDA) have become increasingly powerful in recent years. Dramatic breakthroughs in processing power along with the number of extra features included in these devices have opened the doors to a wide range of commercial possibilities. However, even with all these added, there are few applications that allow much passing of the environmental, educational, information and location based services. As mobile devices become more like Personal Computers (PC) they will come to replace objects we tend to carry around such as cheque books, credit cards, planners, MP3 players etc. In short we will be using them to accomplish our daily tasks.

ConvoPro is an android application that allows users to convert Document Format (PDF) to Word and convert Word to PDF. PDFs are widely used, but require a reader or browser plug-in, and furthermore they are not easy to edit. ConvoPro provides features like editing PDFs by converting word documents to PDFs. The files can be imported from other apps like gallery, file manager etc.

1. Introduction

Portable Document Format (PDF) is a computer application that can easily store, save and transmit the content editable format. This application is accessible and portable across platforms. The PDF document can be easily transferred from one system to another with different configuration. The compression algorithms make the file size of the PDF document smaller. This makes the PDF documents easily transferrable. The PDF file format also supports the multimedia elements. This file format is well known for its security features. A user can protect the document from unauthorised printing, editing, viewing and even copying. The beneficial features make the PDF file format vital and useful for the user for various purposes.

A Word Document refers an editable file format. In this type of file format, the user can prepare customised documents on any subject. This computer application equips the user with a slew of editable features to create attractive and professional documents. These features are beneficial for the users. The Word file format offers the user with an ability to insert shapes, charts, bookmarks, footnotes, hyperlinks and much more. One can even color the page and insert watermarks in the document. These are among the many advanced featured that make the user's task easy and convenient to prepare documents on any subject.

When the user wants to modify or edit the contents in the PDF file, it is not supported. In such case, the user has to create a completely new PDF document adding and modifying according to his requirements, which is a tedious task. In the recent years, this problem is defeated by converting the PDF document into an editable format that is Word file and making it possible to perform all the modifications. However, Word files are not secured for the transmission from one system to another. These security issues can be resolved by converting the Word file to PDF file.

1.1 Objectives

The aim of the project ConvoPro, an Android application is to provide an easy interface for a user for converting PDF documents to Word documents and Word documents to PDF documents. Not everyone has a computer around them at all times for using the conversion tools

that are available online but do have an access to smart phones that are now supporting the Word and PDF file formats, thus making an Android application have an edge over the conversion tools.

1.2 Methodology

ConvoPro is an Android application that converts a PDF document to Word document and vice versa that uses internet for its functioning. On opening the app, there is an app interface which provides two options, to continue for the conversion or to exit from the app. This interface is developed using Eclipse IDE - Android developer tools.

When the user chooses to continue by pressing the convert button, he is directed to a home page that displays various options to select from. This website is developed using HTML and CSS which is hosted onto the internet. The options provided are PDF to Word, Word to PDF, Help and About Us. The user can choose desired option from them.

On choosing PDF to Word or Word to PDF, the user is now directed to another page where he is displayed with the file upload option to select the required file and convert. These pages are developed using Microsoft Visual Studio. The user interface is developed using ASP.Net and back end is developed using C Sharp. The required packages are added as references to the websites and the Dynamic Linking Libraries (DLL) are imported for performing the conversion functions.

1.3 Organization of the project

- The application has three modules which are the app interface, the home page and the conversion module.
- When the first module is opened, the user can either go to the second module or can exit the application. Internet is required to continue.
- If the user chooses to exit the application, he is asked for the confirmation through a dialog box.

- When the user proceeds to second module, he can choose from the options displayed.
- If PDF to Word is chosen, he is directed to the next module where he is asked to upload a file by clicking the Upload button.
- On clicking the Upload, browse window is opened and the user can choose from the PDF files that are available. On clicking the Convert button after selecting the file, the file is converted to Word file with same name as PDF file but with .docx extension.
- Similarly, if Word to PDF file is chosen, he is directed to the next module where he is asked to upload a file by clicking the Upload button.
- On clicking the Upload, browse window is opened and the user can choose from the Word files that are available. On clicking the Convert button after selecting the file, the file is converted to Word file with same name as Word file but with .pdf extension.

2. Theoretical Analysis

2.1 Software Requirements

1. The Eclipse IDE(Java Developers version)
2. The Android SDK
3. Microsoft Visual Studio
4. .NET Framework

The front end is developed with Eclipse IDE (Java Developers version). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plug-ins and The Android SDK is used for internal development of the application which is the process by which new applications are created for the Android operating system. Applications are usually

developed in Java programming language using the Android software development kit (SDK), but other development environments are also available.

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft is used for developing the word file box code and the pdf box code for converting and for designing web site for the convertor. .NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. The conversion process is interfaced with websites that are developed using C sharp language in Microsoft Visual Studio which are hosted onto the internet.

2.2 Hardware requirements

1. Platform - Android device
2. Operating System -Windows 8 and above
3. Processor - Intel Core i5
4. RAM - 4.00GB

2.3 Android

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

2.3.1 Android architecture

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in Java programming language using the Android software development kit (SDK), but other development environments are also available. The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later. As of March 2015, the SDK is not available on Android itself, but the software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plug-in, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and Net Beans IDE also supports Android development via a plug-in. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

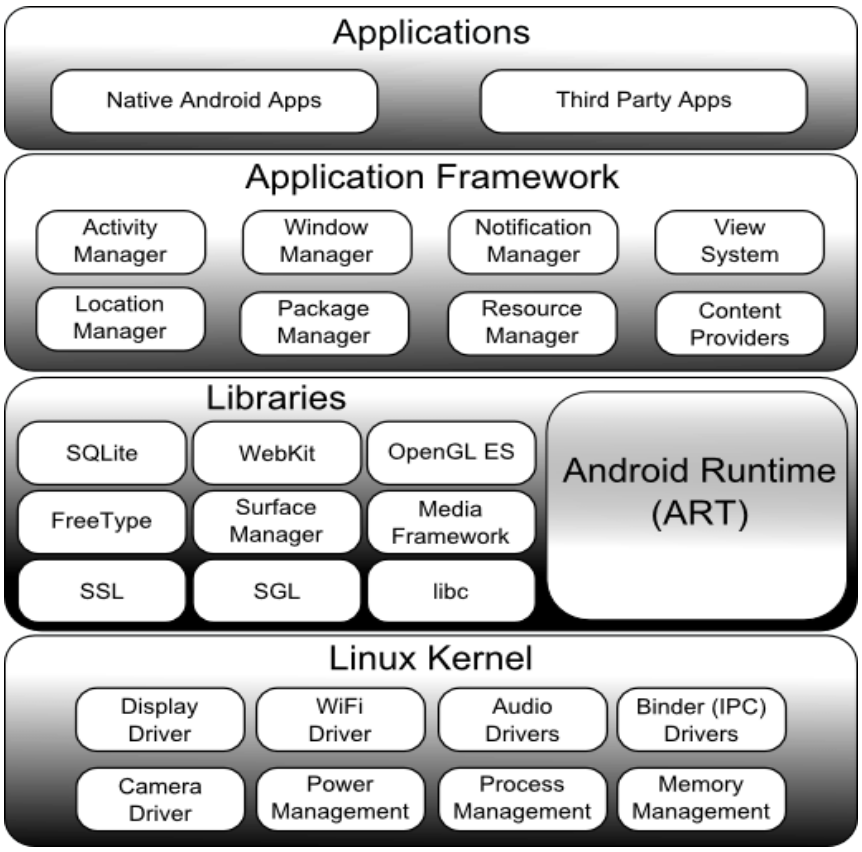

Fig 2.1 Android architecture

2.4 DocX File

DocX is written in an XML format, which consists of a ZIP archive file containing XML and binaries. Content can be analysed without modification by unzipping the file (e.g. in WinZip) and analysing the contents of the archive. The file _rels/.rels contains information about the structure of the document. It contains paths to the metadata information as well as the main XML document that contains the content of the document itself.

Metadata information is usually stored in the folder docProps. Two or more XML files are stored inside that folder, app.xml that stores metadata information extracted from the Word application itself and core.xml that stores metadata from the document itself, such as the author name, last time it was printed, etc. Another folder contains the actual content of the document in a Word document. An XML file called document.xml is the main document, containing most of the content of the document itself

2.5 PDFBox

Apache PDFBox is an open source pure-Java library that can be used to create, render, print, split, merge, alter, verify and extract text and meta-data of PDF files.

2.5.1 Features of PDF Box

- Extract Text

  Extract Unicode text from PDF files.

- Split & Merge

  Split a single PDF into many files or merge multiple PDF files.

- Fill Forms

  Extract data from PDF forms or fill a PDF form.

- Preflight

  Validate PDF files against the PDF/A-1b standard.

- Print

  Print a PDF file using the standard Java printing API.

- Save as Image

  Save PDFs as image files, such as PNG or JPEG.

- Create PDFs

  Create a PDF from scratch, with embedded fonts and images.

2.6 Dynamic Link Libraries (DLL)

2.6.1 Interop.Microsoft.Office.Core

COM Interop aims to provide access to the existing COM components without requiring that the original component be modified. It tries to make the .NET types equivalent to the COM types. In addition, COM Interop allows COM developers to access managed objects as easily as they access other COM objects.

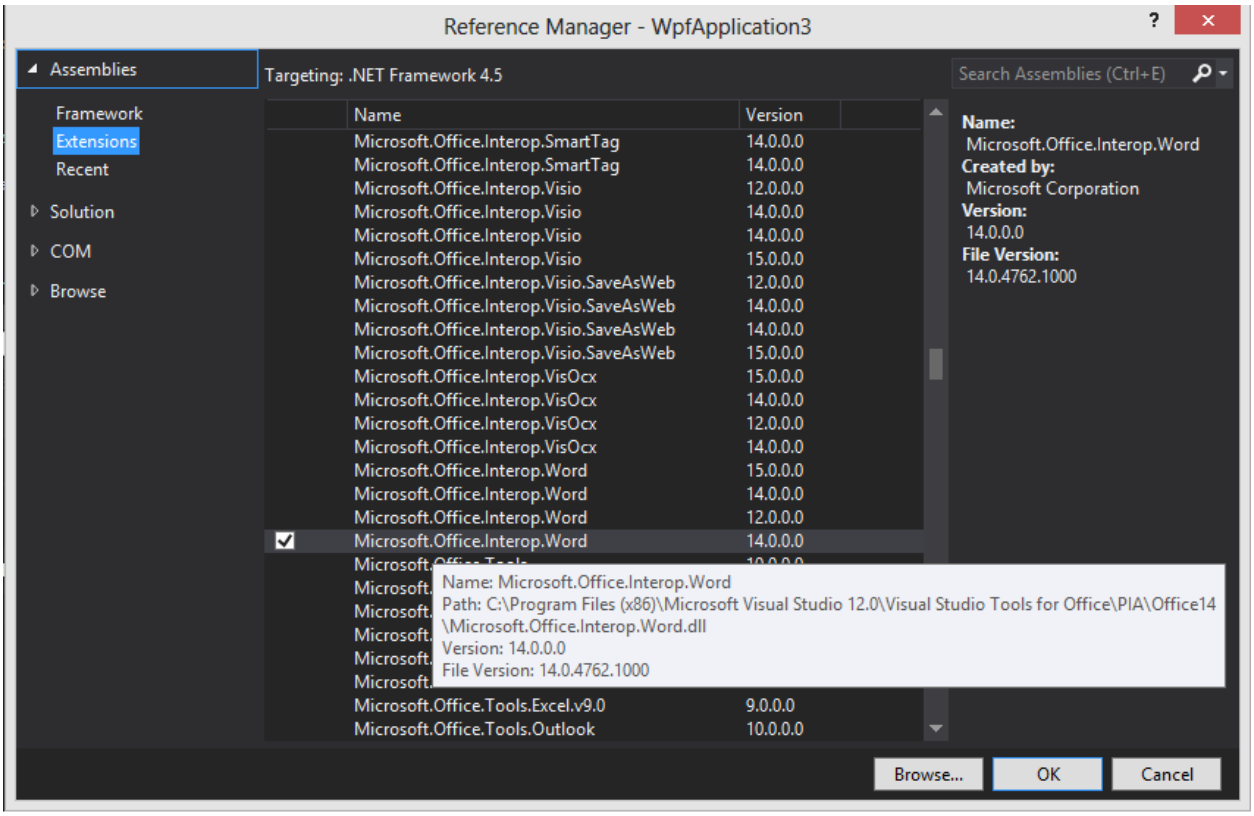

Fig 2.2 DLLs added

The .NET Framework creates a type library and special registry entries when a component is registered. It provides a specialized utility (RegAsm.exe, usually located in C:\Windows\Microsoft.NET\Framework) that exports the managed types into a type library and registers the managed component as a traditional COM component. When the type is instantiated through COM, the .NET CLR is the actual COM object that executes and it merely marshals any method calls or property access to the type implementation. A COM Interop for an existing COM component can be created through the use of the Tlbimp.exe tool supplied with the .NET Framework SDK. This tool will create a set of proxy classes implemented in CIL which is encapsulated in an assembly.

Office Primary Interop Assemblies

To use the features of a Microsoft Office application from an Office project, Primary Interop Assembly (PIA) is must used for the application. The PIA enables managed code to interact with a Microsoft Office application's COM-based object model.

When a new Office project is created, Visual Studio adds references to the PIAs that are required to build the project. In some scenarios, it might need to add references to additional PIAs (for example, if we want to use a feature of Microsoft Office Word in a project for Microsoft Office Excel).

The aspects of using the Microsoft Office PIAs in Office projects:

- Separate primary interop assemblies for building and running projects
- Using features of multiple Microsoft Office applications in a single project
- Full list of primary interop assemblies for Microsoft Office applications
- For more information about primary interop assemblies, see Primary Interop Assemblies.

I. Separate Primary Interop Assemblies for Building and Running Projects

Visual Studio uses different sets of the PIAs on the development computer. These different sets of assemblies are in the following locations:

- A folder in the program files directory.

These copies of the assemblies are used when we write code and build projects. Visual Studio installs these assemblies automatically.

- The Global Assembly cache.

These copies of the assemblies are used during some development tasks, such as when we run or debug projects. Visual Studio does not install and register these assemblies; must do this manually.

II. Primary Interop Assemblies in the Program Files Directory

When Visual Studio is installed, the PIAs are automatically installed to a location in the file system, outside of the global assembly cache. When a new project id created, Visual Studio automatically adds references to these copies of the PIAs to the project. Visual Studio uses these copies of the PIAs, instead of the assemblies in the global assembly cache, to resolve type references when the project is developed and built. These copies of the PIAs help Visual Studio avoid several development issues that can occur when different versions of the PIAs are registered in the global assembly cache.

Visual Studio installs these copies of PIAs to the following locations on the development computer:

- %ProgramFiles%\Microsoft Visual Studio 12.0\Visual Studio Tools for Office\PIA\Office14
  (or %ProgramFiles(x86)%\Microsoft Visual Studio 12.0\Visual Studio Tools for Office\PIA\Office14 on 64-bit operating systems)
- %ProgramFiles%\Microsoft Visual Studio 12.0\Visual Studio Tools for Office\PIA\Office15
  (or %ProgramFiles(x86)%\Microsoft Visual Studio 12.0\Visual Studio Tools for Office\PIA\Office15 on 64-bit operating systems)

III. Primary Interop Assemblies in the Global Assembly Cache

To perform certain development tasks, the PIAs must be installed and registered in the global assembly cache on the development computer. Typically, the PIAs are installed automatically when Office is installed on the development computer. The Office PIAs are not required on end-user computers to run Office solutions

IV. Using Features of Multiple Microsoft Office Applications in a Single Project

Every Office project template in Visual Studio is designed to work with a single Microsoft Office application. To use features in multiple Microsoft Office applications, or to use

features in an application or component that does not have a project in Visual Studio, it is must to add a reference to the required PIAs.

In most cases, developer should add references to the PIAs that are installed by Visual Studio under the %ProgramFiles%\Microsoft Visual Studio 12.0\Visual Studio Tools for Office\PIA\ directory. These versions of the assemblies appear on the Framework tab of the Reference Manger dialog box.

If the PIAs are already installed and registered in the global assembly cache, these versions of the assemblies appear on the COM tab of the Reference Manager Dialog box. Developer should avoid adding references to these versions of the assemblies, because there are some development issues that can occur when we use them. For example, if you have registered different versions of the PIAs in the global assembly cache, your project will automatically bind to the version of the assembly that was registered last—even if you specify a different version of the assembly on the COM tab of the Reference Manager Dialog box.

2.6.2 Apache PDFBox - A Java PDF Library

The Apache PDFBox library is an open source Java tool for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Apache PDFBox also includes several command line utilities. Apache PDFBox is published under the Apache License v2.0.

2.6.2.1 Org.apache.pdfbox.pdmodel

The PDModel package represents a high level API for creating and manipulating PDF documents.

Class	Description
PD Destination Name Tree Node	This class holds all of the name trees that are available at the document level.
PD Document	This is the in-memory representation of the PDF document.
PD Document Catalog	This class represents the acroform of a PDF document.
PD Document Information	This is the document metadata.
PD Document Name Destination Dictionary	This encapsulates the "dictionary of names and corresponding destinations" for the /Dest entry in the document catalog.
PD Document NameDictionary	This class holds all of the name trees that are available at the document level.
PD Embedded FilesNameTreeNode	This class holds all of the name trees that are available at the document level.
PD JavaScript NameTreeNode	This class holds all of the name trees that are available at the document level.
PD Page	This represents a single page in a PDF document.
PD Pageable	Adapter class that implements the Pageable and Printable interfaces for printing a given PDF document.
PD PageNode	This represents a page node in a pdf document.
PD Resources	This represents a set of resources available at the page/pages/ stream level.

Table 1 Classes in PDModel and their description

2.6.2.2 Org.apache.pdfbox.util

This package contains utility classes that are used by the PDFBox project.

Class and Description
Image Parameters: This contains all of the image parameters for in inline image.
Matrix: This class will be used for matrix manipulation.
Overlay Position: Possible location of the overlaid pages: foreground or background.
PDF Operator: This class represents an Operator in the content stream.
PDFStreamEngine: This class will run through a PDF content stream and execute certain operations and provide a callback interface for clients that want to do things with the stream.
PDFTextStripper: This class will take a pdf document and strip out all of the text and ignore the formatting and such.
PositionWrapper: Wrapper of Text Position that adds flags to track status as linestart and paragraph start positions.
TextNormalize: This class allows a caller to normalize text in various ways.

Table 2 Classes in util and their description

3. System Design

3.1 Class diagram

A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes and operations.

Class Name
Attributes
Operations

Fig 3.1 Class Structure

- Classes represent an abstraction of entities with common characteristics.
- Associations represent the relationship between classes.

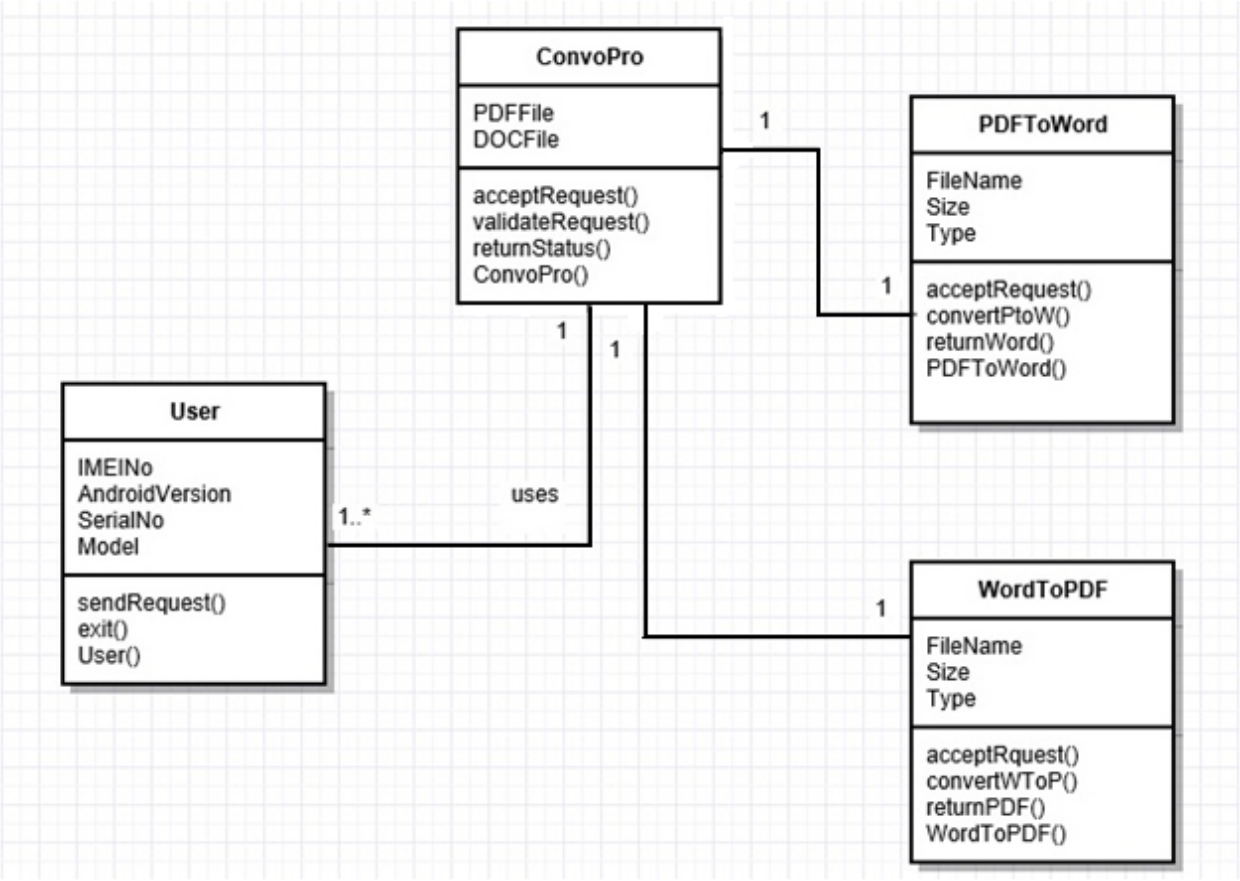

Fig 3.2 Class Diagram

User:

User interacts with the ConvoPro by sending the request.

Attributes: IMEINo

AndroidVersion

SerialNo

Model

- By using the sendRequest( ) function the user sends the request for converting the file.
- Using exit( ) function the user exits the application.

ConvoPro:

ConvoPro accept the request from the User, validates the request and upon validation it will send the request to the appropriate class, which is either PDFToWord or WordToPDF.

Attributes: PDFFile*

DOCFile

After the request is processed, the status of the request which is either the file is converted or not is returned to the User through retunStatus( ) function.

PDFToWord:

PDFToWord class takes the request and the PDF file from ConvoPro and converts it into a Word document and returns it.

Attributes: FileName

Size

Type

WordToPDF:

WordToPDF class takes the request and the Word file from ConvoPro and converts it into a PDF document and returns it.

Attributes: FileName

Size

Type

3.2 Use case diagram

A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

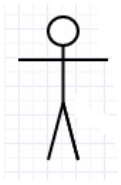

Fig 3.3 Actor

An actor represents a role that an outsider takes on when interacting with the business system. For instance, an actor can be a customer, a business partner, a supplier, or another business system. Every actor has a name

Association:

An association is the relationship between an actor and a business use case. It indicates that an actor can use a certain functionality of the business system – the business use case.

Use case: Use case describes the interaction between an actor and a system

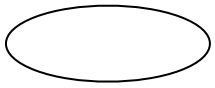

Fig 3.4 Use case

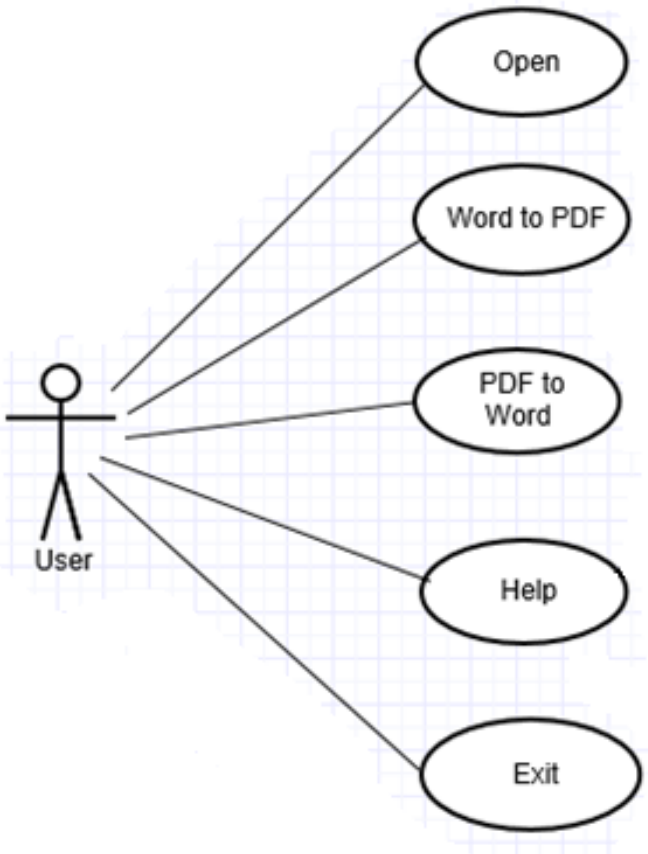

Fig 3.5 Use case

3.3 Activity diagrams

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Activity diagrams show the overall flow of the control. Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represent actions
- Diamonds represent decisions
- Bars represent the start (split) or end (join) of concurrent activities
- A black circle represents the start (initial state) of the work flow
- An encircled black circle represents the end (final state)

Arrows run from the start towards the end and represent the order in which activities happen.

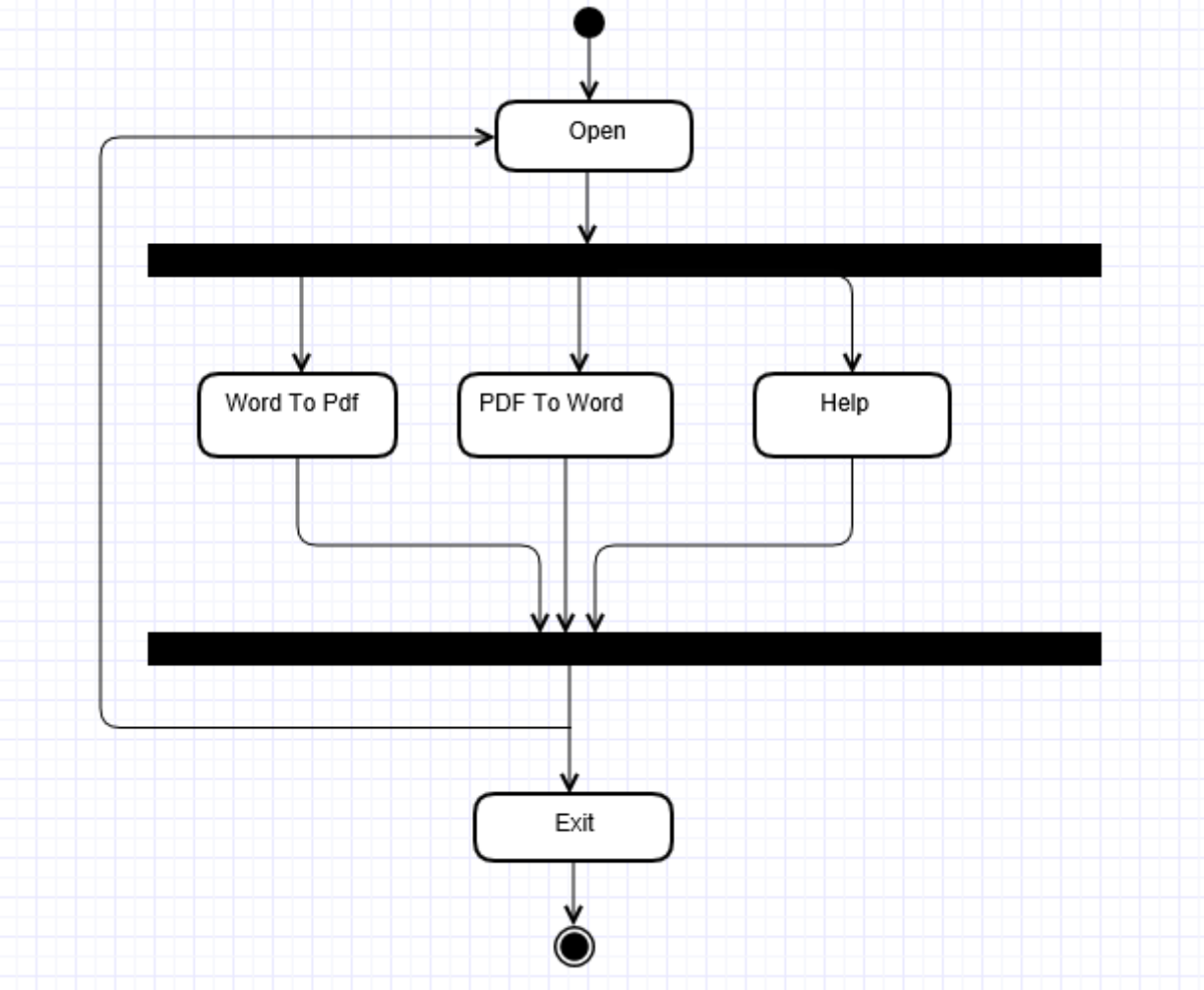

Fig 3.6 Activity Diagram

3.4 Sequence Diagrams

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct as a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Objects as well as classes can be targets on a sequence diagram, which means that messages can be sent to them. A target is displayed as a rectangle with some text in it. Below the target, its lifetime extends for as long as the target exits. The lifeline is displayed as a vertical dashed line.

When a target sends a message to another target, it is shown as an arrow between their lifelines. The arrow originates at the sender and ends at the receiver. Near the arrow, the name and the parameters of the message for an messages are shown.

The white rectangles on the lifeline are called activations and indicate that an object is responding to a message. It starts when the message is received and ends when he object is done handling the message**.**

3.4.1 Sequence diagram for converting Word to PDF

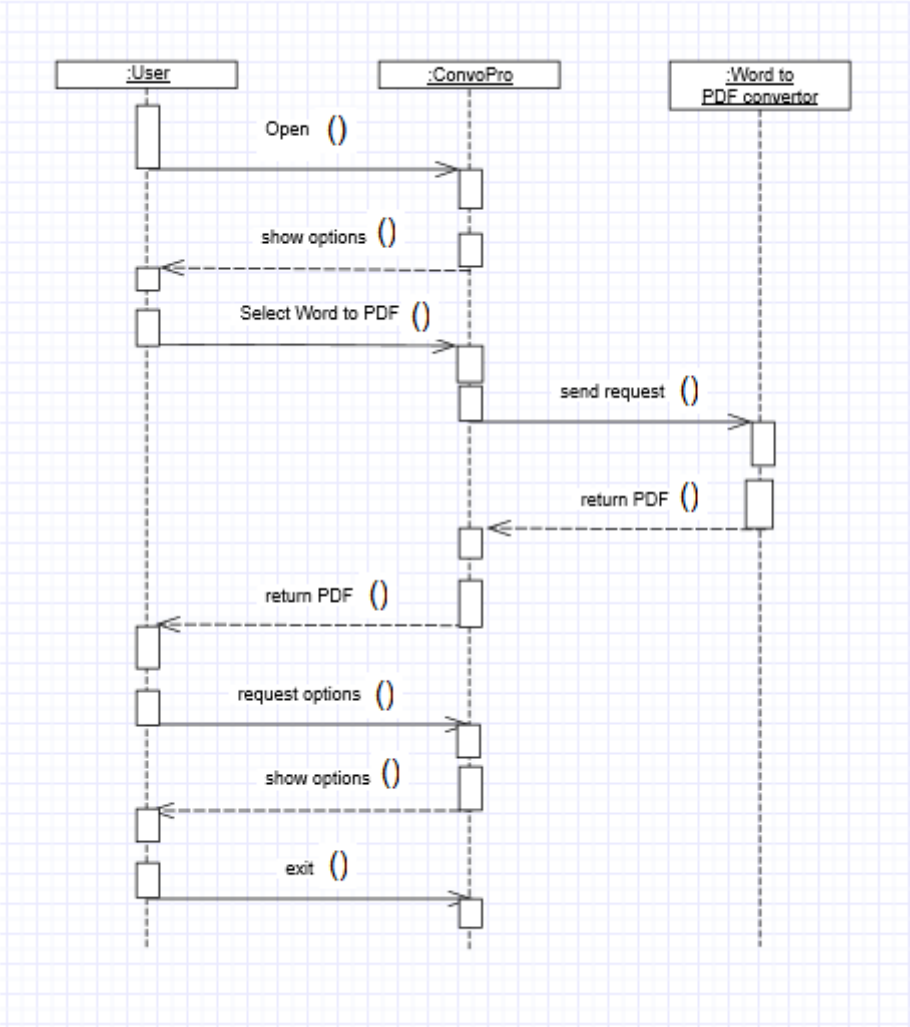

Fig 3.7 Sequence diagram for converting Word to PDF

3.4.2 Sequence diagram for converting PDF to Word

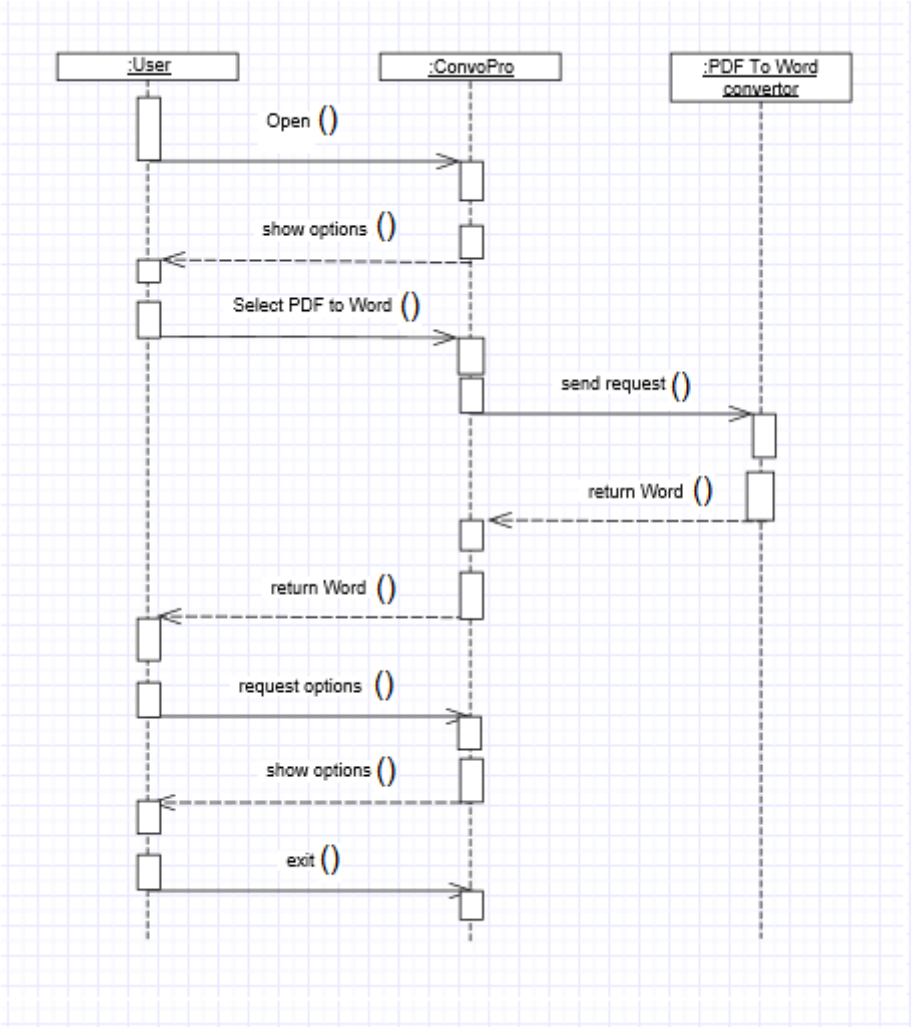

Fig 3.8 Sequence diagram for converting PDF to Word

4. Output Screens

Fig 4.1 App Interface

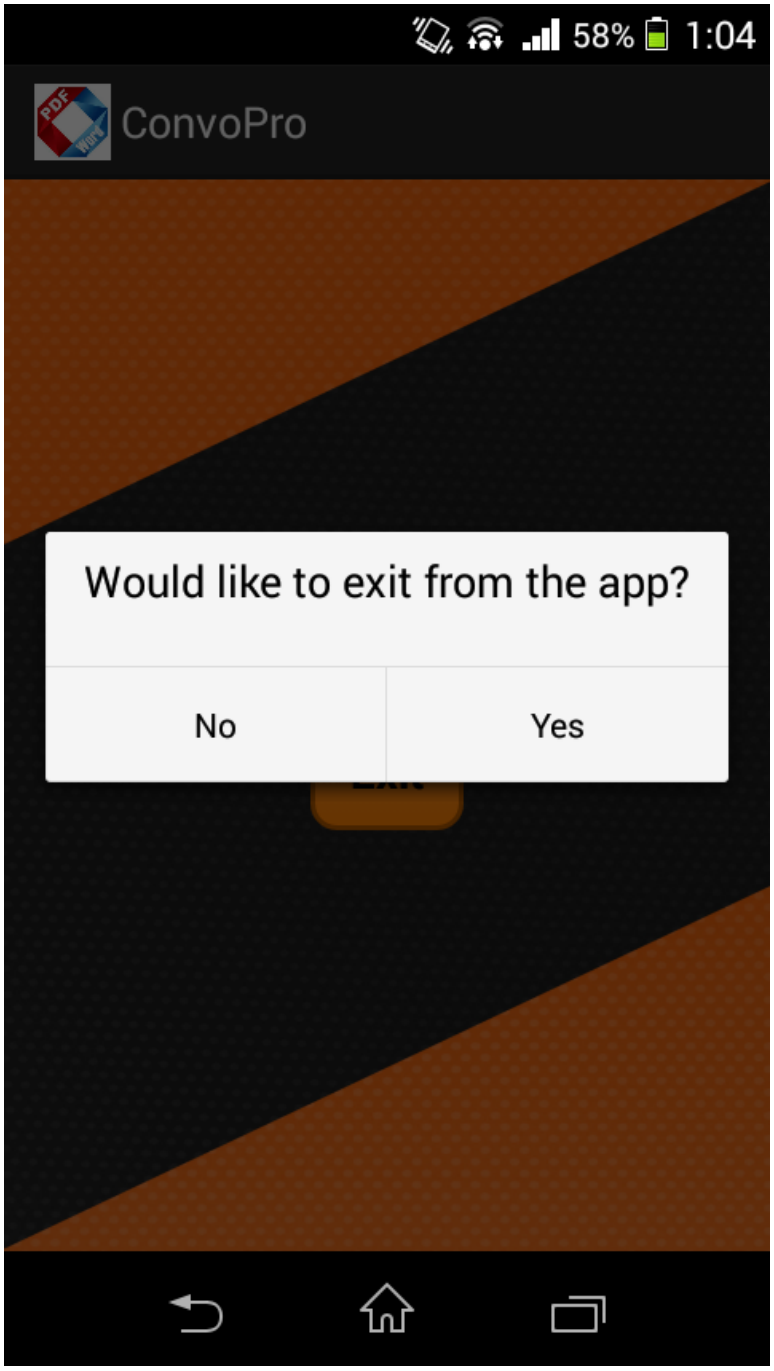

Fig 4.2 User request to exit

Fig 4.3 Home Screen

Fig 4.4 Options available

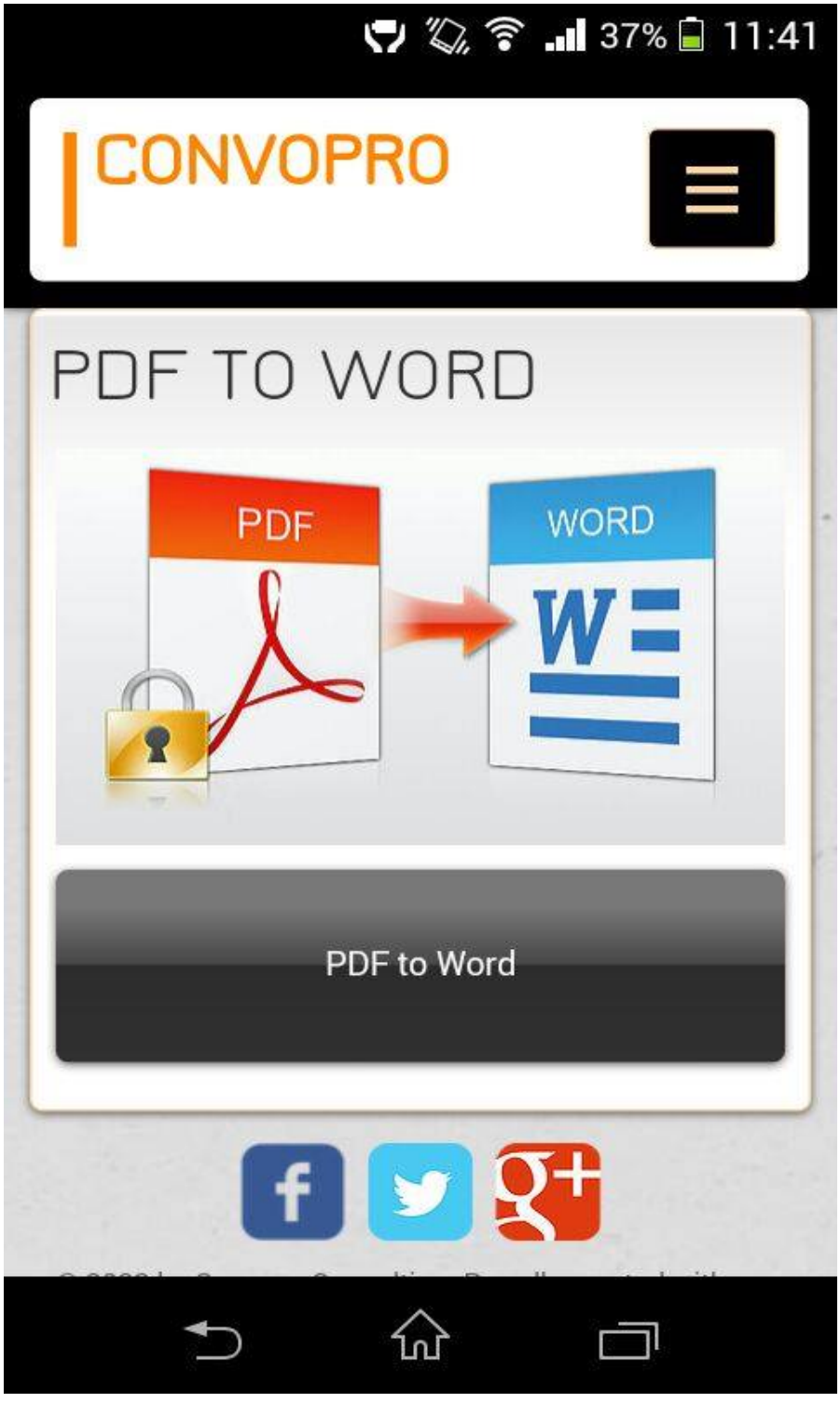

Fig 4.5 PDF to Word module

Fig 4.6 Interface for selecting PDF file

Fig 4.7 File selected for conversion

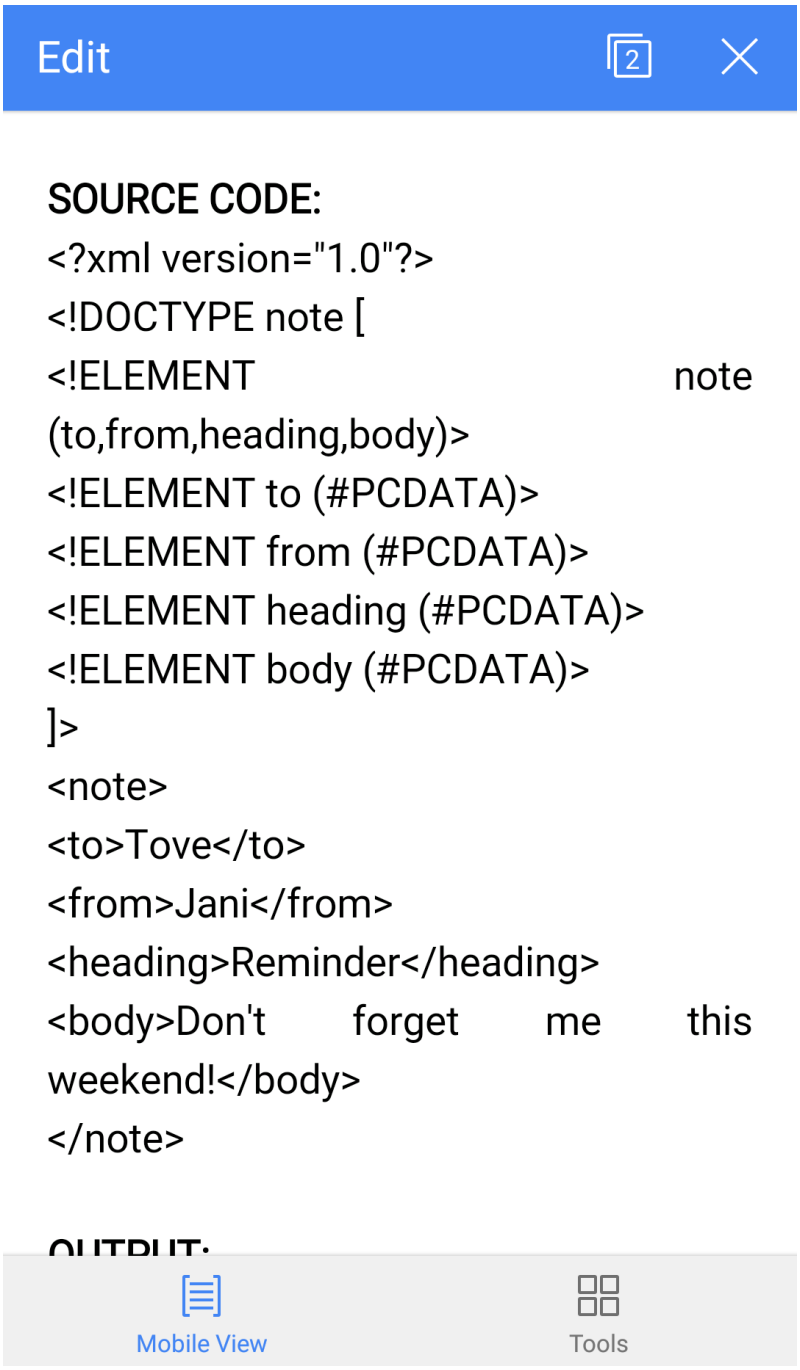

Fig 4.8 Sample.pdf file converted to Sample.docx

Fig4.9 Word to PDF module

Fig 4.10 Interface for selecting PDF file

Fig 4.11 Sample.doc file selected for conversion

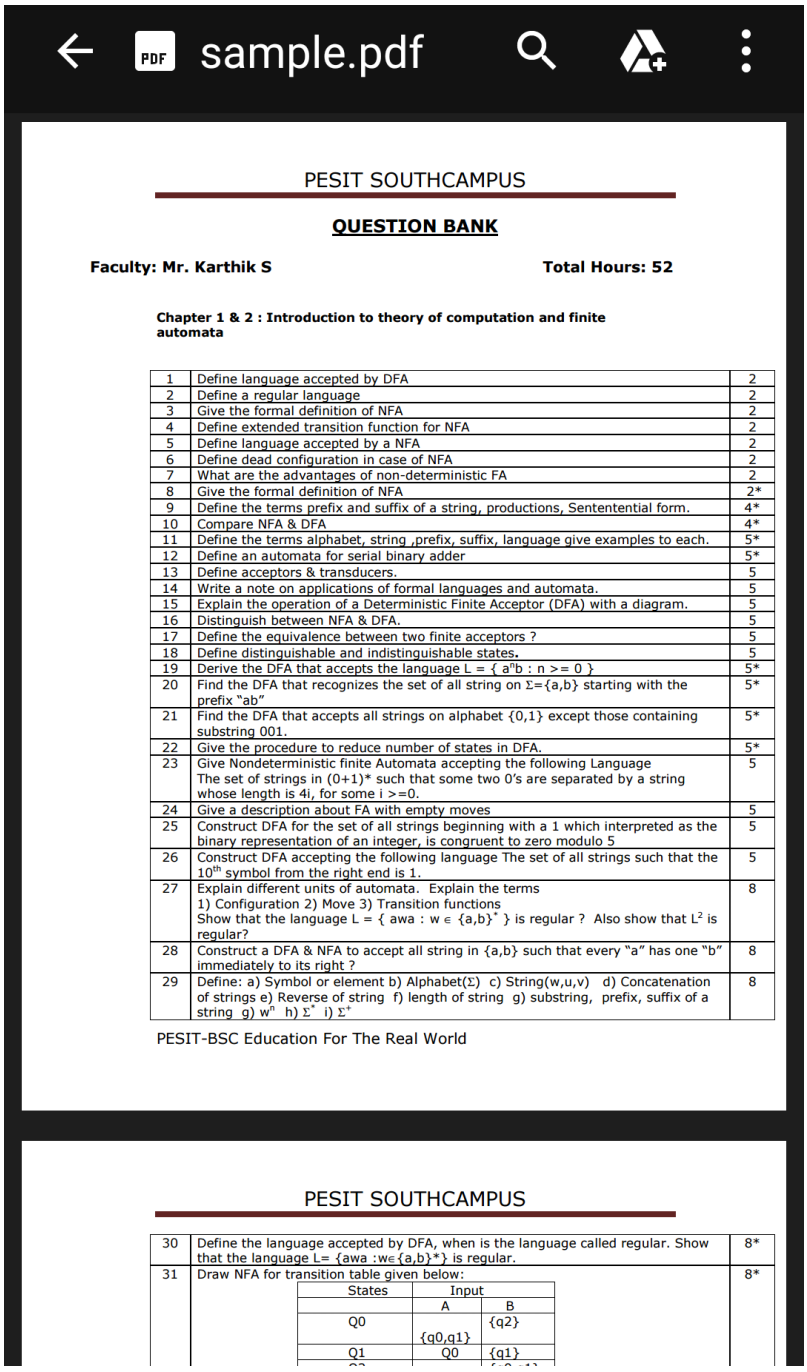

Fig 4.12 Sample.doc file converted to Sample.pdf

Fig 4.13 Help page - PDF to Word

Fig 4.14 Help page – Word to PDF

5. Conclusion and Scope for future enhancements

This project is to provide easy interface for users to access the app for converting their files from one format to another that is from a word file to a pdf file and vice versa. The key objective is to develop a user friendly application which can be accessed by every person who is using android phones. This application is also very useful for the people who continuously work with the large word and pdf documents.

Future work related to this project can focus on converting various other files as excel files, ppt files etc into other required formats along with additional features added into the application. A single platform can be provided for creating. Modifying and converting them into respective formats that is useful and simple to use.

REFERENCES

Textbooks

1. Matthew MacDonald, ASP.NET: The Complete Reference, First Edition, Tata McGraw – Hill Education, August 2002.
2. Bill Phillips and Brian Hardy, Android Programming: The Big Nerd Ranch Guide, First Edition, Big Nerd Ranch Guides, 28 March 2013.
3. Dave Smith and Jeff Friesen, Android Recipes: A Problem-Solution Approach, Second Edition, Apress, December 3, 2012

Websites

1. http//developer.android.com/tools/help/emulator.html

2. http//androiddevelopers.blogspot.com/

3. http://www.c-sharpcorner.com/

4. http://www.c-sharpcorner.com/UploadFile/07c1e7/convert-pdf-to-word-using-C-Sharp/

5. https://msdn.microsoft.com/en-us/library/dd264733.aspx

6. http://www.w3schools.com/aspnet/default.asp

7. http://www.w3schools.com/css/css_link.asp

8. http://www.tutorialspoint.com/asp.net/asp.net_server_controls.htm

9. https://msdn.microsoft.com/en-us/library/tbze79kd(v=vs.80).aspx