1. INTRODUCTION

1.1 Introduction to the system

1.1.1 Project Title

Digital File Manipulation.

1.1.2 Category

Windows Application.

1.1.3 Overview

The application provides various manipulation operations on different types of digital files like Images, Audio, Video, Text, Doc, Pdf, xls, csv.

1.2 Background

1.2.1 Introduction of the Company

Not Applicable.

1.2.2 Brief note on Existing System

In existing system, when we cut text from image the file is saved as image itself because we are cropping the image. To highlight the text in image or pdf , we have to use external tools. Image color recognition is difficult in existing system and a lot of time is consumed by existing system. Symbol and Mathematical equations are difficult recognize. User has to use different existing software to achieve the tasks.

1.3 Objective of the System

- Objectives are the pre-determined goals or outcomes of the system process.
- The main objective of the system is to develop an application which can be used to manipulate various digital files like image, video, audio, text, doc, pdf, xls and csv. And to provide one stop solution for different types of digital file manipulations.

1.4 Scope of the System

- Scope is the limitation that a process faces from the beginning to the end.
- Image recognition, symbol and mathematical equation recognition is difficult.
- Clarity of image is must for text recognition.
- Clarity Audio and Video is must for extracting text and to extract audio from video.

1.5 Structure of the System

1.5.1 Text Manipulation in an Image

1.5.1.1 Extracting Text from Images

The user provides a clear image as input and the application retrieves the text in that image and saves it in a text file.

1.5.1.2 Extracting Text from Region of Interest

The user provides a clear image as input and selects a region in that image and the application will retrieve the text in that image.

1.5.1.3 Highlighting Text

The user provides a clear image and a word or a sentence as input, and the application highlights that word/sentence in the image ,else gives a prompt message telling 'There is no such word or sentence'.

1.5.2 Spread Sheet Manipulation

1.5.2.1 Excel to Text

The user provides an Excel file as input and the application converts it into a text file and gives it as output.

1.5.2.2 CSV to Text

The user provides a CSV file as input and the application converts it into a text file and gives it as output.

1.5.2.3 Excel to CSV

The user provides an Excel file and the excel sheet name as inputs and the application converts it into a CSV file and gives it as output.

1.5.2.4 Search a Column

The user provides an Excel file required column name as input and the application converts the specified column data into a text file and gives it as output.

1.5.3 Video-Audio Manipulation

1.5.3.1 Text To Audio

The user provides a text file as input and the application will convert it into audio file, which is the output.

1.5.3.2 Extracting Audio from Video

The user provides a video as input and the application retrieves the audio in that video and provides it as output.

1.5.2.3 Trim Audio

The user provides an audio, start and end time as input and the application will crop the audio which is the output.

1.5.4 File Manipulation

1.5.4.1 PDF to TEXT

The user provides a PDF file as input and the application converts it into a text file and gives it as output.

1.5.4.2 DOCX to Text

The user provides a DOCX file as input and the application converts it into a text file and gives it as output.

1.5.4.3 PDF to DOCX

The user provides a PDF file as input and the application converts it into a DOCX file and gives it as output.

1.5.4.4 TEXT to DOCX

The user provides a Text file as input and the application converts it into a DOCX file and gives it as output.

1.6 System Architecture

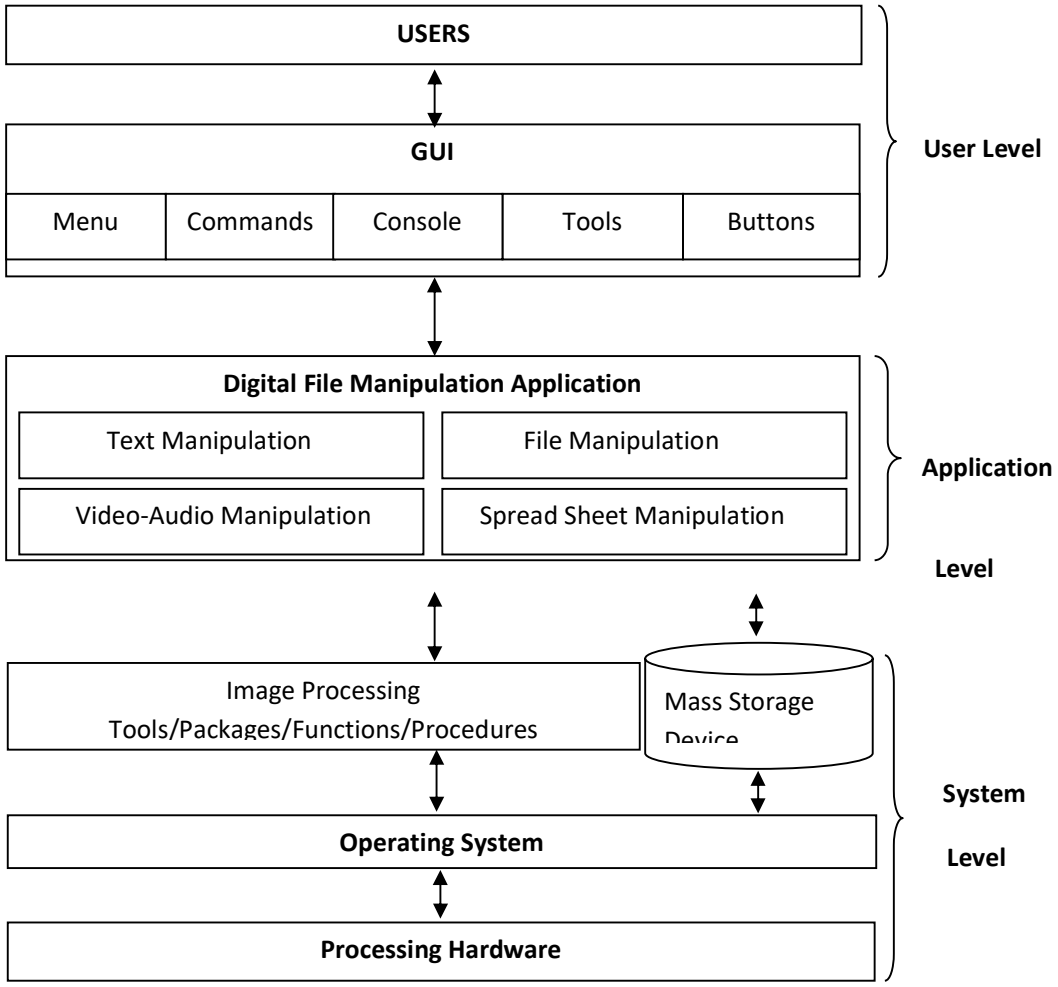


Fig no: 1.1 System Architecture

1.7 End User

The End-User can be anyone who requires the functionalities provided by this system. They can give image as an input and perform operations like extracting text, highlighting text. The user can give video and audio as input and perform operations like extracting audio from video, extracting text from video and extract text from audio. They can also input a doc/pdf/text file and convert them to text/doc/pdf.

1.8 Software/Hardware need for the development

Software:

Python, Pytessaract, ffmpeg, Windows OS.

Hardware:

The system must have more than 2GB of RAM.

1.9 Software/Hardware need for the implementation

Software:

Pytessaract, ffmpeg, Windows OS.

Hardware:

The system must have more than 2GB of RAM, and the screen resolution must be more than 1600X900 pixels.

2. SOFTWARE REQUIREMENT SPECFICAION

2.1. Introduction

Software Requirement Specification is a description of a software to be developed. It layouts functional and non-functional requirements and may include the set of use cases that describes the user interaction that they must provide. It helps the client to understand their own needs. It provides justification of final product. It helps to bridge the communication gap between developer and user.

The Purpose of Software Requirement Specification document is to provide a detailed overview of the software product and its parameter and goals. It specifies all the requirement of the application.

The boundaries of software products are defined set of requirements. The software development team designing implements tests and delivers these requirements to you. A requirement is an atomic unit of software product from the view point of the user. As a rule, requirements are always correct, unambiguous, verifiable and traceable requirements are numbered and prioritized.

2.2. Overall Description

2.2.1. Product perspective

2.2.1.1. System interfaces:

This application runs in the latest version of python, ffmpeg and pytesseract on windows OS.

2.2.1.2 User interfaces

This applications GUI provides menus, toolbars, buttons, frames, containers, grids allowing for easy control by a keyboard and a mouse.

2.2.1.3 Hardware interfaces

Not Applicable.

2.2.1.4 Software interfaces

Not Applicable.

2.2.1.5 Communications interfaces

This application requires internet connectivity as communication interface.

2.2.1.6 Interfaces with Server

Not Applicable.

2.2.2. Product Functions

The product function relates physical output of a production process to physical input as features of production.

The general function of this application is to manipulate the digital files as per the user requirements. This application provides various functionalities like text manipulation on the image, spreadsheet manipulation, video-audio manipulations and file conversions like converting text, pdf and word document with each other.

2.2.3. User characteristics

The End User must have basic knowledge about the usage of the system, which is how to operate the application and the user must install the pytesseract and ffmpeg engine in their system.

2.2.4. General constraints

The software works on any windows computer system. And it requires clear image for manipulation and audio quality must be good and video quality must be good.

The system must have more than 2GB of RAM, and the screen resolution must be more than 1600X900 pixels. The system must have external softwares called pytesseract and ffmpeg and requires internet connectivity.

2.2.5. Assumptions and Dependencies

These factors are not design constraints on the software but any changes to these factors can affect the requirement in the SRS.

The image quality must be good for image and text manipulation. Video and Audio quality must be good. The system must have more than 2GB of RAM and must run on latest version of OS and the system must have pytesseract engine and ffmpeg software.

2.3. Special Requirements (Software / Hardware - if any)

This application requires a software called pytesseract which is an external engine and ffmpeg software.

2.4. Functional requirements

In the functional requirement section, the functional capabilities of the system are described. In this organization, the functional capabilities for all the modes of operation of the software are given. For each functional requirement, the required inputs, desired outputs and processing requirements will have to be specified.

2.4.1 Text Manipulation in an Image

2.4.1.1 Extracting Text from Images

a. Input: Input is a clear image.

b. Process: Process include extracting text from the given image.

c. Output: Output is the text extracted from the image.

2.4.1.2 Extracting Text from Region of Interest

a. Input: Input is a clear image.

b. Process: Process include extracting text from the region of interest from the given image.

c. Output: Output is the text extracted from the specified region in the image

2.4.1.3 Highlighting Text

a. Input: Input is a clear image as input.

b. Process: Process include Highlighting text in the given image.

c. Output: Output is the text highlighted in the image or a message saying that “the given text is not found”.

2.4.2 Spread Sheet Manipulation

2.4.2.1 Excel to Text

a. Input: Inputs is an Excel file.

b. Process: Process include extracting text from the given Excel file to save it as a text file.

c. Output: Output is a text file.

2.4.2.2 Csv to Text

a. Input: Input is a Csv file.

b. Process: Process include extracting text from the given Csv file to save it as a text file.

c. Output: Output is a text file.

2.4.2.3 Excel to Csv

a. Input: Input is an Excel file and the sheet name.

b. Process: Process include extracting text from the given Excel file to save it as a Csv file.

c. Output: Output is a Csv file.

2.4.2.4 Search a Column:

a. Input: Input is an Excel File and the required column name.

b. Process: Process include finding the given column name in the excel file and writing that column data on a text file.

c. Output: Output is a Text file.

2.4.3 Video-Audio Manipulation

2.4.3.1 Text To Audio

a. Input: Input is a text file.

b. Process: Process include converting the text to audio

c. Output: Output is audio file.

2.4.3.2 Extracting Audio from Video

a. Input: Input is a video with clear audio.

b. Process: Process include extracting audio from the video.

c. Output: Output is the audio file extracted from the video.

2.4.3.3 Trim Audio

a. Input: Input is a clear audio file.

b. Process: Process include trimming the audio file to given start and end time.

c. Output: Output is the Audio file.

2.4.4 File Manipulation

2.4.4.1 Pdf to Text

a. Input: Input is a pdf file.

b. Process: Process include extracting text from the given pdf file to save it as a text file.

c. Output: Output is a text file.

2.4.4.2 Docx to Text

a. Input: Input is a Docx file.

b. Process: Process include extracting text from the given Docx file to save it as a text file.

c. Output: Output is a text file.

2.4.4.3 Pdf to Docx

a. Input: Input is a pdf file.

b. Process: Process include extracting text from the given pdf file to save it as a Docx file.

c. Output: Output is a Docx file.

2.4.4.4 Text to Docx

a. Input: Input is a text file.

b. Process: Process include converting the text file to Docx file.

c. Output: Output is a Docxx file.

2.5. Design Constraints

The client environment may restrict the designer to include some design constraints that must be followed.

2.5.1. Hardware Constraint

The system must have more than 2GB of RAM, and the screen resolution must be more than 1600X900 pixels.

2.5.2. Software Constraint

The system must have Windows OS, ffmpeg and pytesseract .

2.5.3. Fault Tolerance

Fault tolerance requirements can place a major constraint on how the system is to be designed. Fault tolerance requirements often make the system more complex and expensive, so they should be minimized.

If any fault occurs due to blur image or blur audio or video, then the application shows an error message stating the fault and asks the user to input clear image, audio or video.

2.5.4. Security

Currently security requirements have become essential and major for all types of systems. Security requirements place restrictions on the use of certain commands, control access to database, provide different kinds of access, requirements for different people, require the use of passwords and cryptography techniques, and maintain a log of activities in the system.

2.5.5. Standard Compliance

It specifies the requirements for the standard the system must follow. The standards may include the report format, Type of Navigations,Naming Conventions for Button, access keys, shortcut keys.

2.6. System Attributes

- **Availability**

  Availability refers to the percentage of time that the infrastructure, system, or solution remains operational under normal circumstances in order to serve its intended purpose.

- **Portability**

  Portability, in relation to software, is a measure of how easily an application can be transferred from one computer environment to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits.

- **Reliability**

  Reliability refers to the probability that the system will meet certain performance standards in yielding correct output for a desired time duration.

- **Maintainability**

  Maintainability refers to the ease with which you can repair, improve and understand software code. Software maintenance is a phase in the software development cycle that starts after the customer has received the product.

- **Scalability**

  Software scalability is an attribute of a tool or a system to increase its capacity and functionalities based on its users' demand. Scalable software can remain stable while adapting to changes, upgrades, overhauls, and resource reduction.

2.7. Other Requirements (if any)

Not Applicable.

3. System Design (Functional Design)

3.1 Introduction

- System design is the process of defining the architecture, Module interface and data for a system to satisfy specified requirements.
- The purpose of the design phase is to plan the solution of the problem Specified by the requirements documents.
- This is the first step that moving from problem domain to the solution domain.
- The design of the system is essentially a blueprint or a plan for a solution for the system.

3.2 Assumption and Constraints

An assumption is a condition you think to be true and constraints is fixed limitations of project development.

- All the functional requirement collected from client are sufficient for the project life-cycle.
- All the Non functional and Specific requirement specified in SRS are well enough for the development of system.
- Time constraint.

3.3 Functional decomposition

Functional decomposition is the process of taking a complex process and breaking it down into its smaller, simpler parts. Using functional decomposition large or complex functionalities are more easily understood. It is mainly used during project analysis phase, so each phase can be viewed as software. So this has modular with some sub modules.

3.3.1 System software architecture.

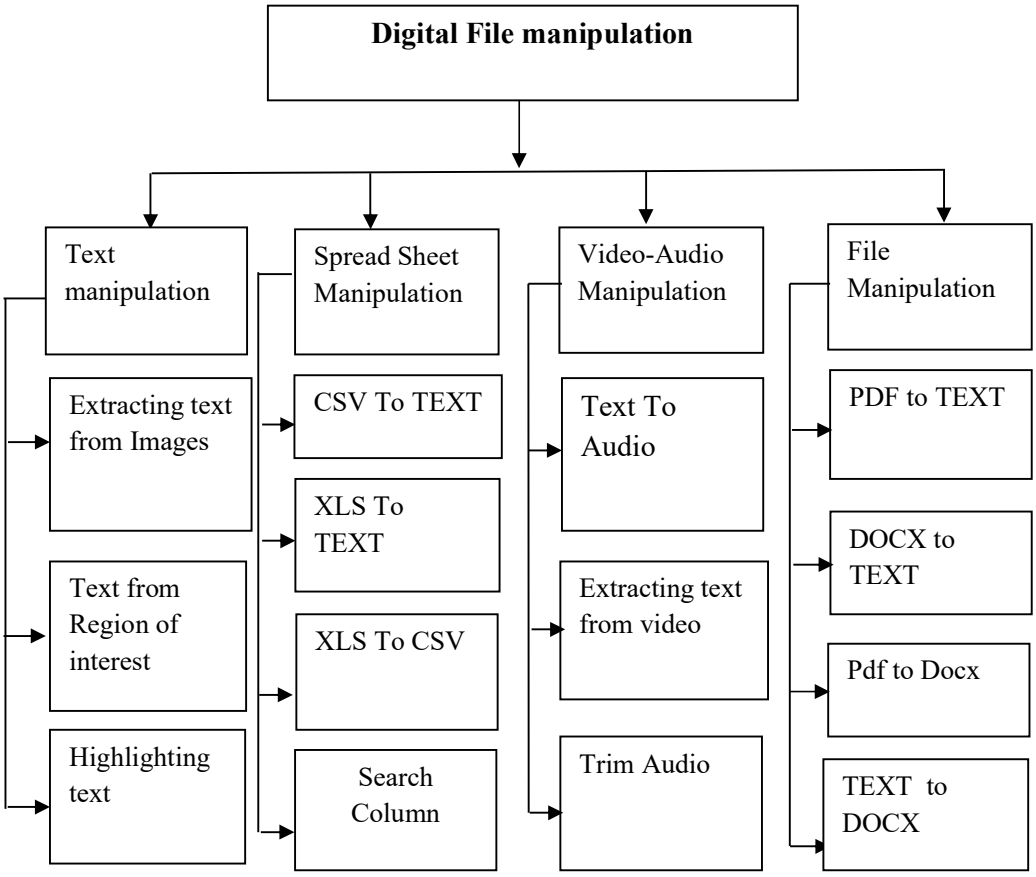


Fig no: 3.1 Software Architecture

3.3.2 System Technical Architecture

Fig no: 3.2 System Technical Architecture

3.3.3 System Hardware Architecture

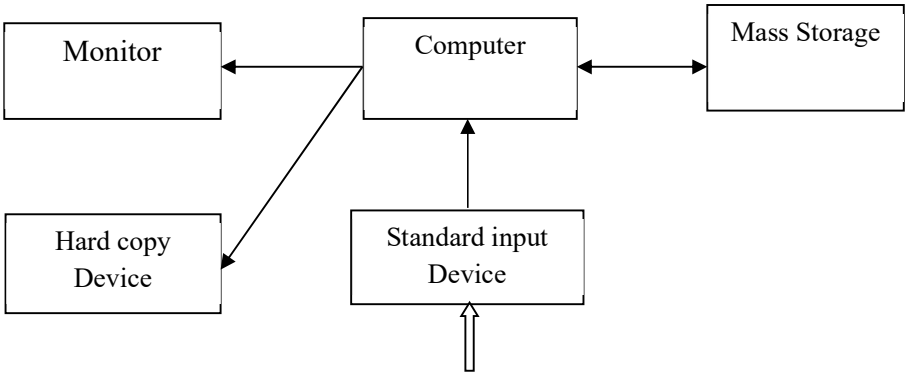


Fig no: 3.3 System Hardware Architecture

3.3.4 External Interfaces

Not applicable

3.4 Description of the programs

3.4.1 Context flow Diagram

In CFD entire system is considered as a single process. Context flow Diagram shows input and outputs of the system. It shows all the external entities that interact with the system and how the data flow between this external entities and system.

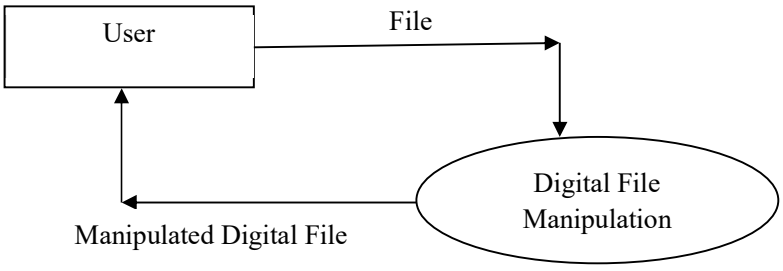


Fig no: 3.4 Context Flow Diagram

3.4.2 Data Flow Diagram (DFD's Level 1,Level 2,Level 3)

Data Flow Diagram(DFD)

Data flow diagram shows the flow of the data through system. Data flow diagram also called the data flow graphs. It views a system as a function that transforms the inputs into desired outputs. It aims to capture the transformation that takes place within a system to the input data so that eventually the output data is produced.

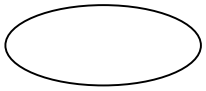
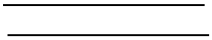
SYMBOLS	NAMES	DESCRIPTION
	Process	It performs transformation of the data from one state to another.
	Source/Sink	It represents the external entity that may either source or sink.
	Flow of data	It represents the flow of data from source to destination.
	Data Source/Data	It is placed where data is stored.

Table no: 3.1 Data Flow Diagram

Level 0 DFD

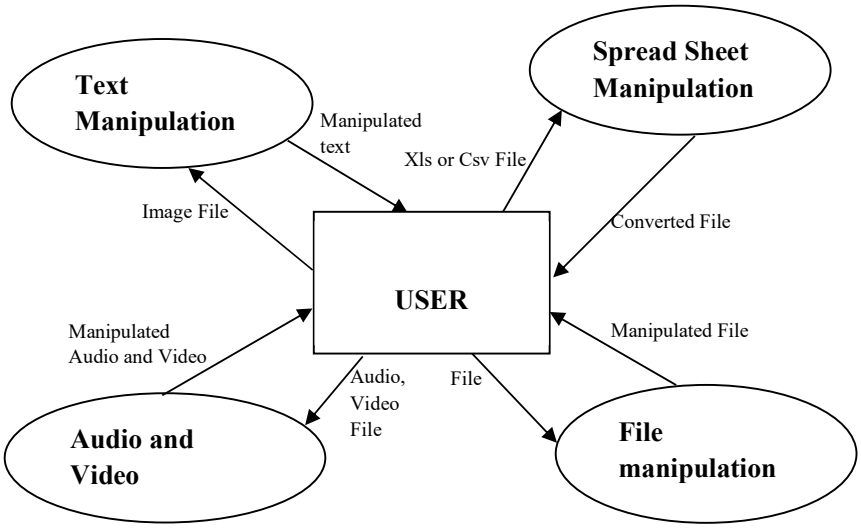


Fig no: 3.5 Level 0 DFD

Level 1 DFD

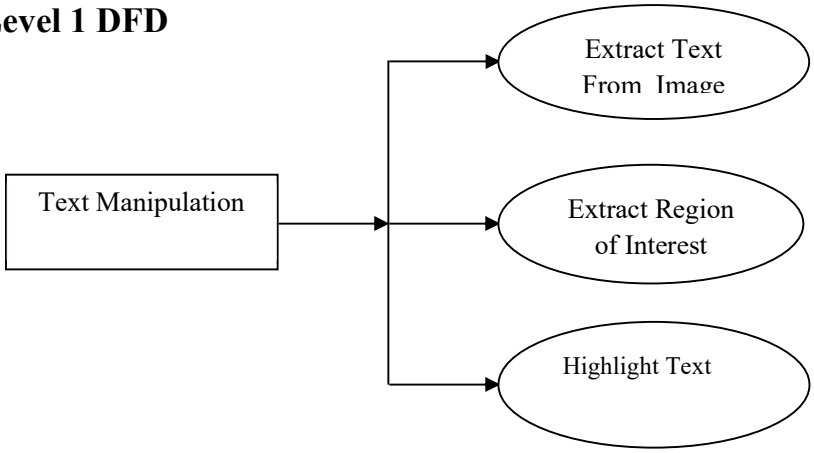


Fig no: 3.6 Text Manipulation Level 1 DFD

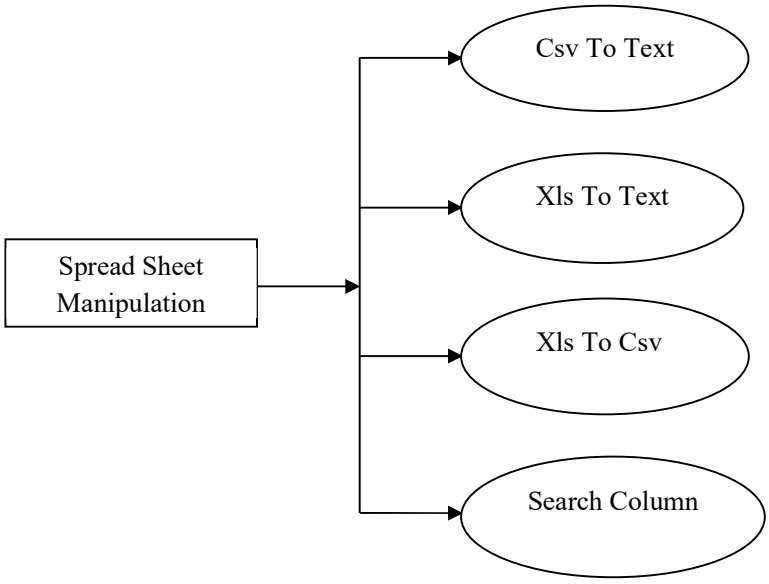


Fig no: 3.7 Spreadsheet Manipulation Level 1 DFD

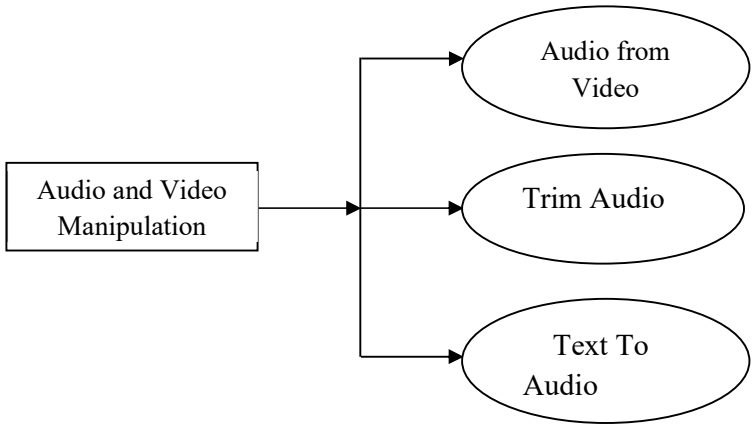


Fig no: 3.8 Audio and Video Manipulation Level 1 DFD

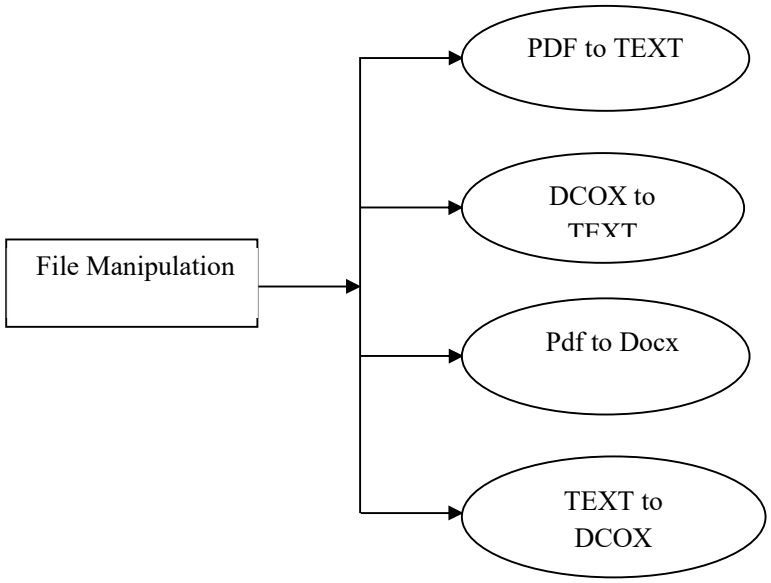


Fig no: 3.9 File Manipulation Level 1 DFD

3.5 Description of Components

3.5.1 Functional Component-1 Text Manipulation

3.5.1.1 Extracting Text From Images

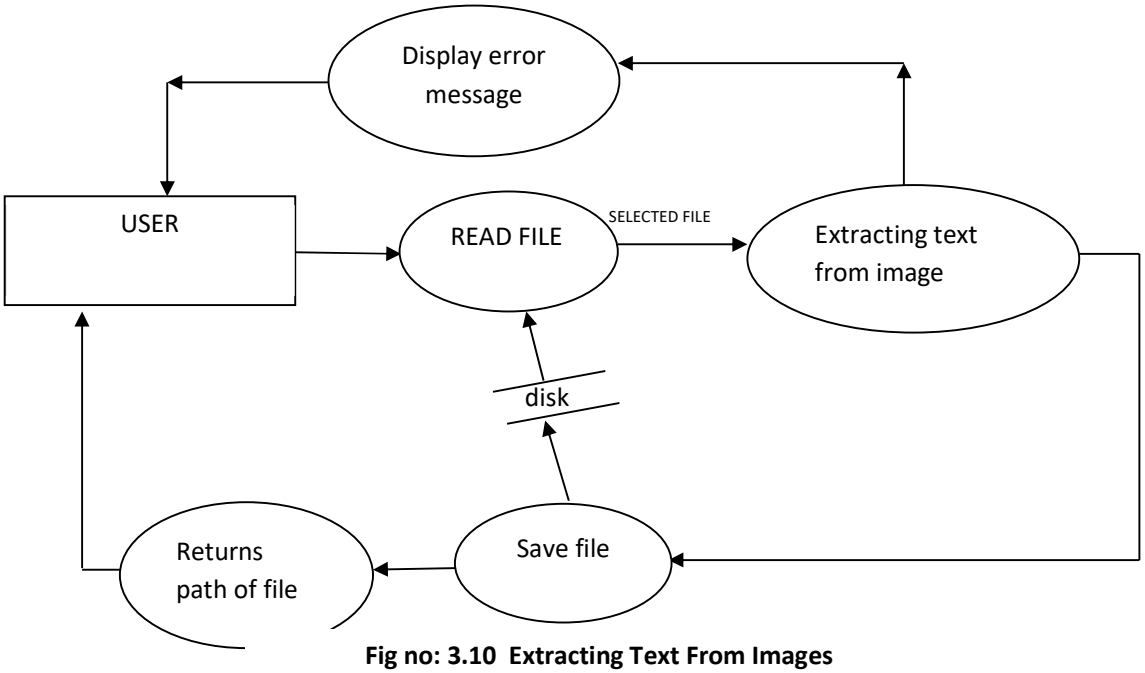


Fig no: 3.10 Extracting Text From Images

3.5.1.1.1 Input

User Provide the Clear Images

3.5.1.1.2 Process Definition

Application retrieves the text in the image

3.5.1.1.3 Output

Application saves the text in Text file format.

3.5.1.1.4 Interface with other functional component

Independent module

3.5.1.1.5 Resource Allocation

System Internal Storage

3.5.1.1.6 User interface

Buttons ,Frames ,Entry box, Message box, Label.

3.5.1.2 Extracting Text from Region of Interest

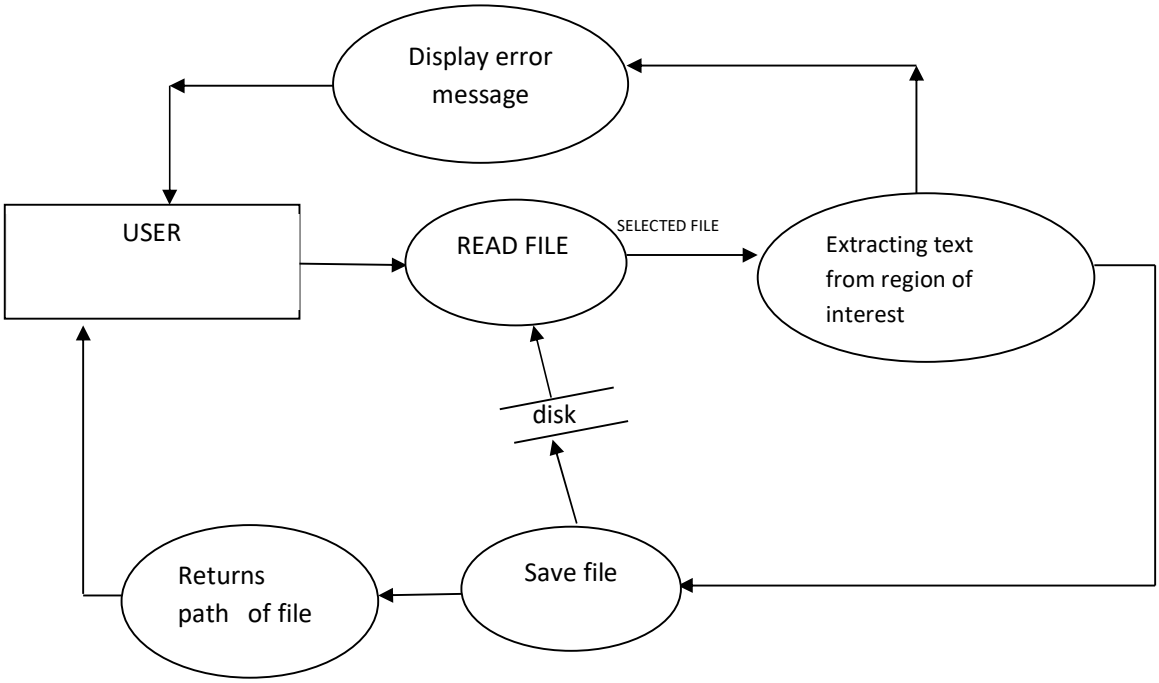


Fig no: 3.11 Extracting Text From Region of Interest

3.5.1.2.1 Input

User Provide the clear Image

3.5.1.2.2 Process Definition

Here user specifies the region in that image.

3.5.1.2.3 Output

Application Will retrieve the text in that Image

3.5.1.2.4 Interface with other functional component

Independent module

3.5.1.2.5 Resource Allocation

System Internal Storage

3.5.1.2.6 User interface

Button, Frames, Entry box, Message box, Label.

3.5.1.3 Highlighting Text

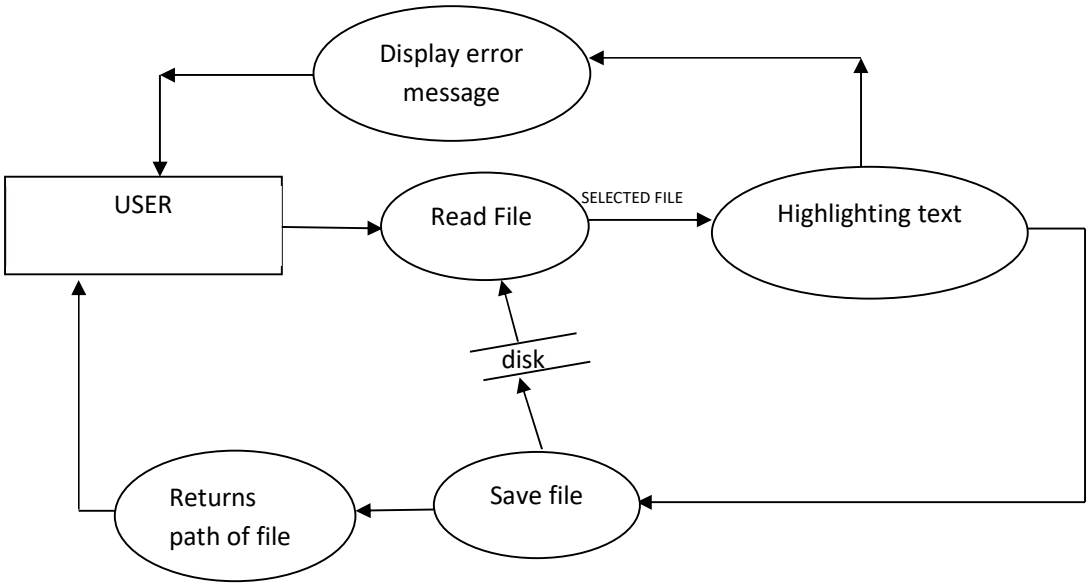


Fig no: 3.12 Highlighting Text

3.5.1.3.1 Input

User Provide the clear Image and word or sentence.

3.5.1.3.2 Process Definition

Here user highlight the specified text in an Image.

3.5.1.3.3 Output

Application will return the image with highlighted text.

3.5.1.3.4 Interface with other functional component

Independent module

3.5.1.3.5 Resource Allocation

System Internal Storage.

3.5.1.3.6 User Interface

Button, Frames, Entry box, Message box, Label.

3.5.2 Functional Component 2 Spread Sheet Manipulation

3.5.2.1 Csv To Text

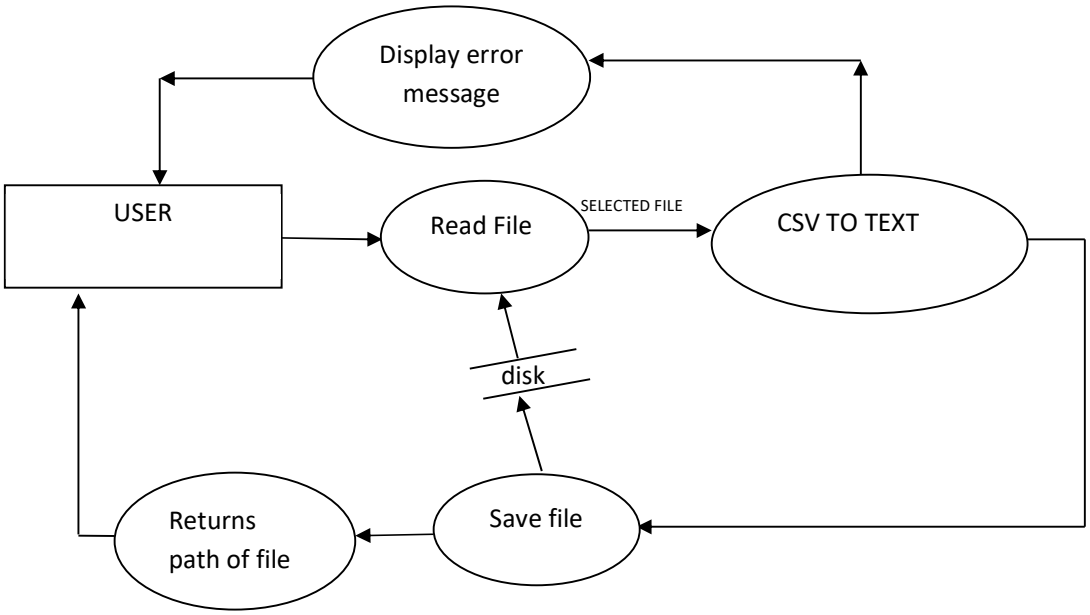


Fig no: 3.13 Csv To Text

3.5.2.1.1 Input

User Provides a Csv File using independent module.

3.5.2.1.2 Process Definition

Application process the Csv file and converts to Text file.

3.5.2.1.3 Output

Application will returns the text file and create the text file in original location.

3.5.2.1.4 Interface with other functional component

Independent module

3.5.2.1.5 Resource Allocation

System Internal Storage

3.5.2.1.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.2.2 Xls To Text

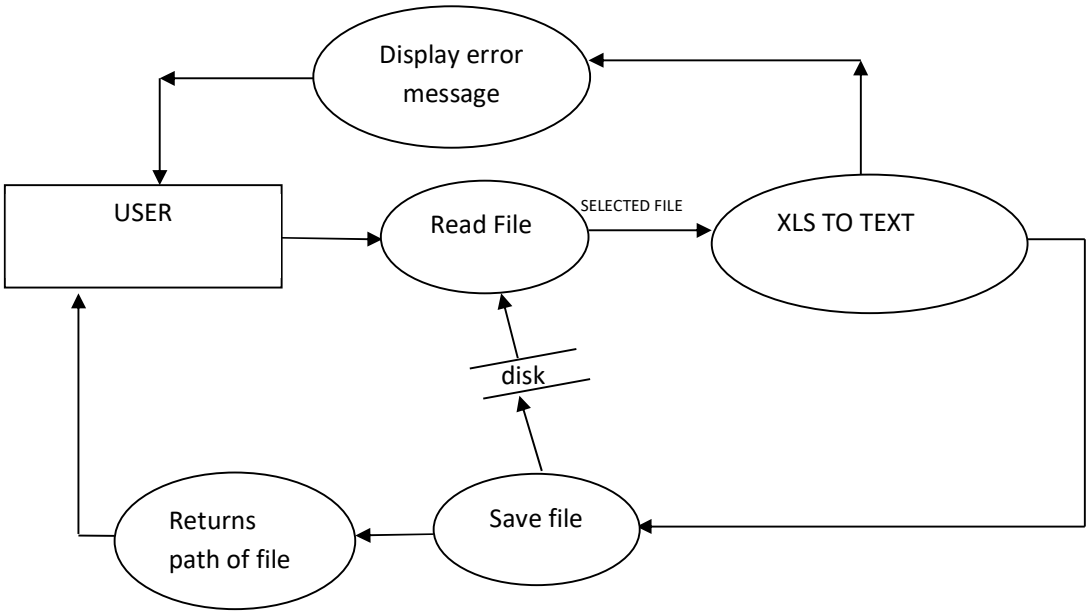


Fig no: 3.14 Xls To Text

3.5.2.2.1 Input

User Provide the Xls file using independent module.

3.5.2.2.2 Process Definition

Application process the Xls file and convert to Text format.

3.5.2.2.3 Output

Application will returns the text file and create the text file

In the original location.

3.5.2.2.4 Interface with other functional component

Independent module

3.5.2.2.5 Resource Allocation

System Internal Storage

3.5.2.2.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.2.3 Xls To Csv

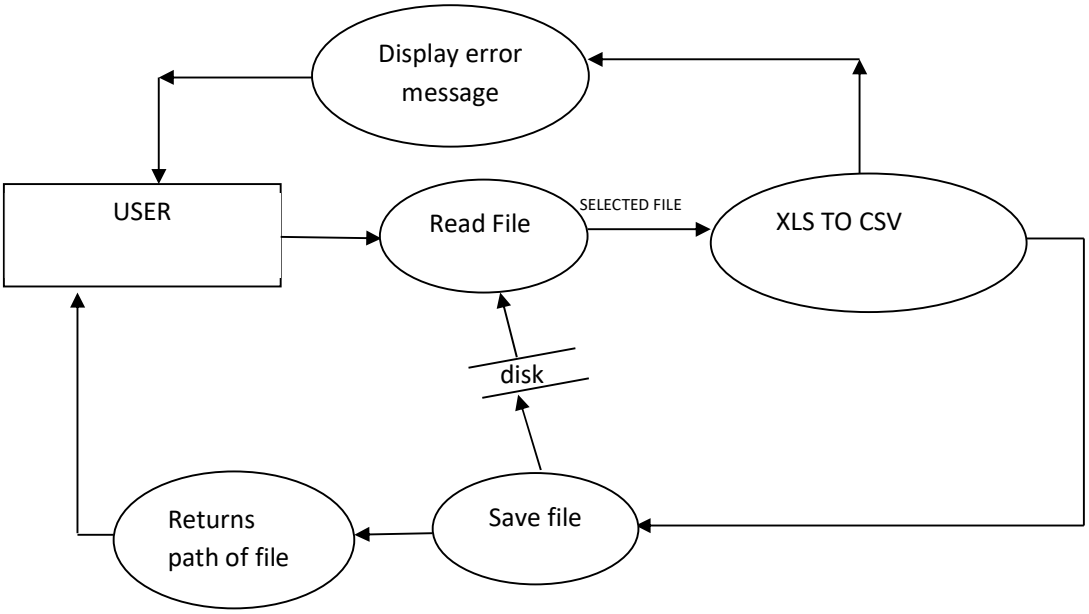


Fig no: 3.15 Xls To Csv

3.5.2.3.1 Input

User Provide the Xls file using independent module.

3.5.2.3.2 Process Definition

Application converts the Xls file to Csv file.

3.5.2.3.3 Output

Application return the Csv file and create the Csv file in original location.

3.5.2.3.4 Interface with other functional component

Independent module

3.5.2.3.5 Resource Allocation

System Internal Storage

3.5.2.3.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.2.4 Search column

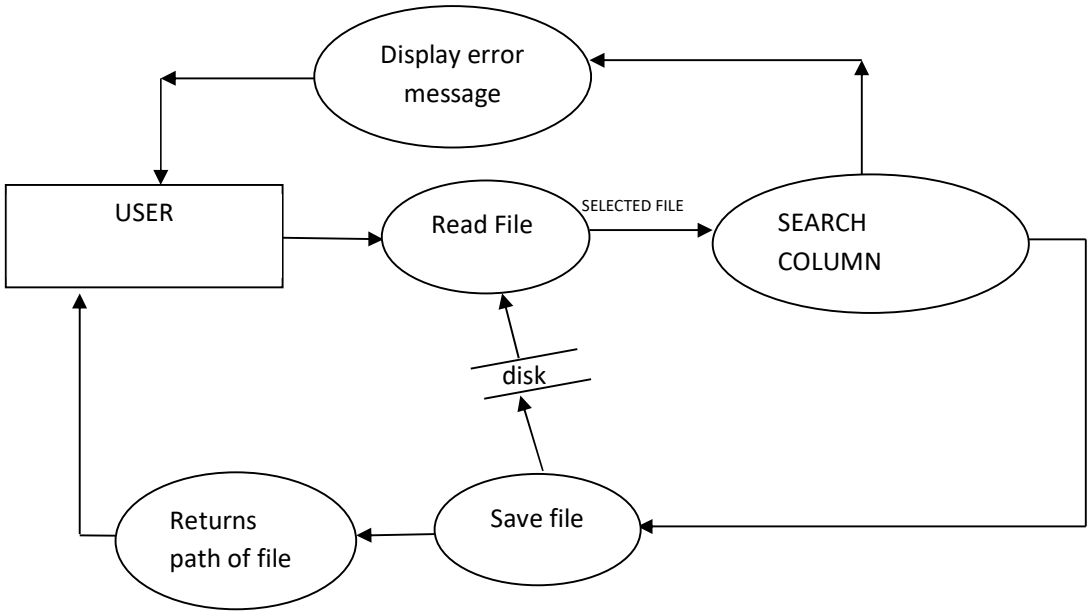


Fig no: 3.16 Search Column

3.5.2.4.1 Input

User Provide the Xls file using independent module.

3.5.2.4.2 Process Definition

Application process the xls file and search the columns..

3.5.2.4.3 Output

Application return the searched columns.

3.5.2.4.4 Interface with other functional component

Independent module

3.5.2.4.5 Resource Allocation

System Internal Storage

3.5.2.4.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.3 Functional Component 3 Video and Audio Manipulation

3.5.3.1 Text To Audio

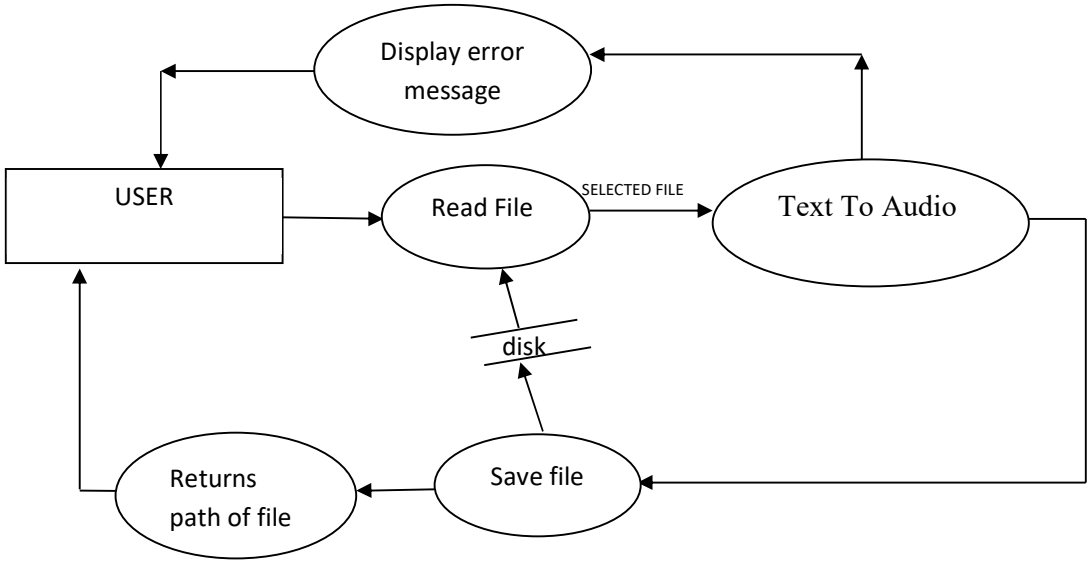


Fig no: 3.17 Text To Audio

3.5.3.1.1 Input

User Provide the Text file as input.

3.5.3.1.2 Process Definition

Application converts the text to audio.

3.5.3.1.3 Output

Application will save the audio file.

3.5.3.1.4 Interface with other functional component

Independent module

3.5.3.1.5 Resource Allocation

System Internal Storage

3.5.3.1.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.3.2 Extracting the Audio from Video

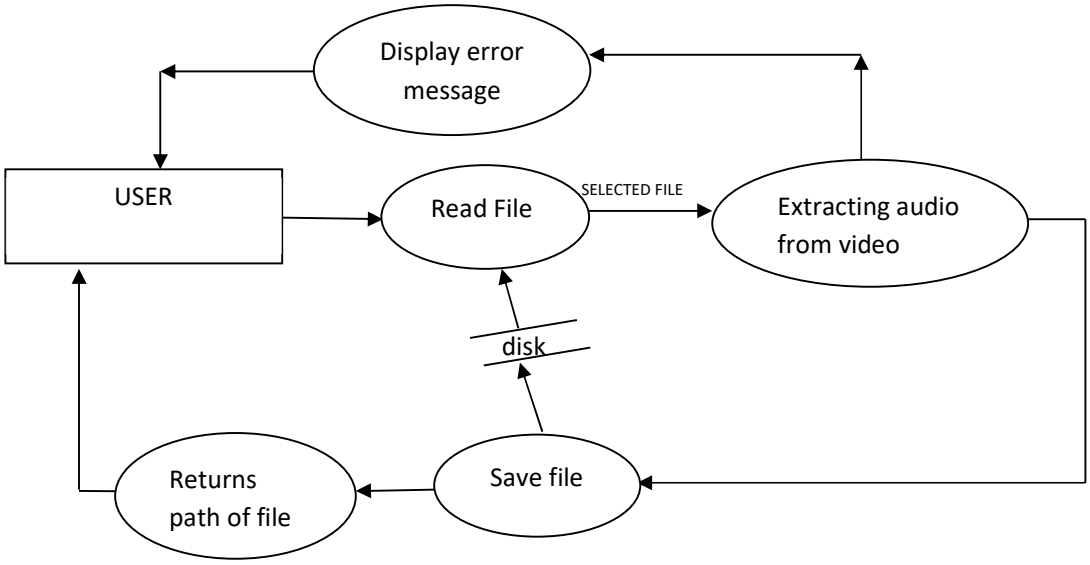


Fig no: 3.18 Extracting the Audio from Video

3.5.3.2.1 Input

User Provide the Video as input in this application.

3.5.3.2.2 Process Definition

Application will retrieves audio in the video.

3.5.3.2.3 Output

Application will produce the audio as output.

3.5.3.2.4 Interface with other functional component

Independent module

3.5.3.2.5 Resource Allocation

System Internal Storage

3.5.3.2.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.3.3 Trim Audio

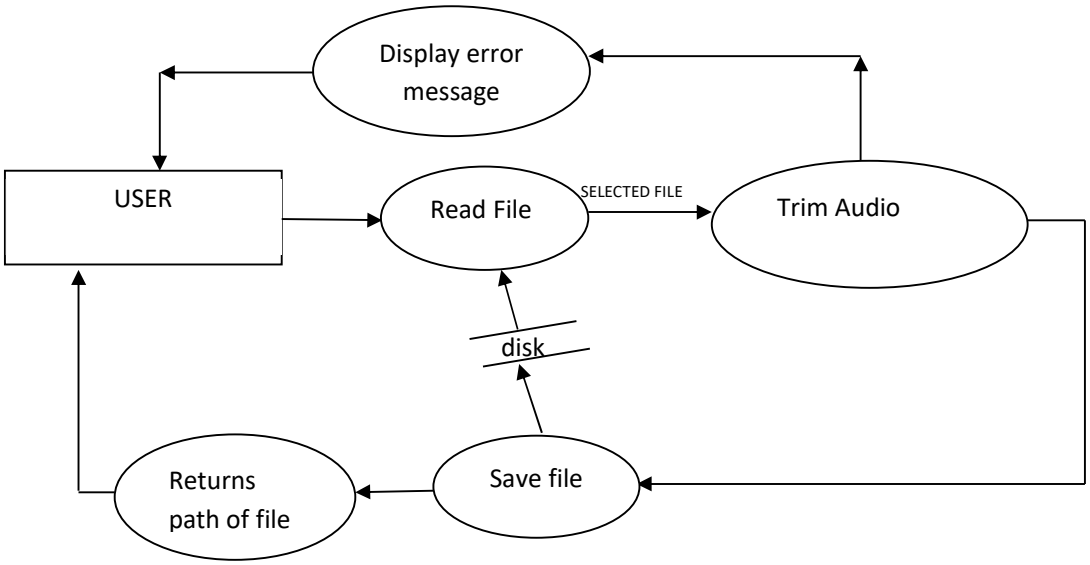


Fig no: 3.19 Trim Audio

3.5.3.3.1 Input

User Provide the audio as input to this application.

3.5.3.3.2 Process Definition

Application retrieves the words spoken in the audio.

3.5.3.3.3 Output

Application will writes the text in text file.

3.5.3.3.4 Interface with other functional component

Independent module

3.5.3.3.5 Resource Allocation

System Internal Storage

3.5.3.3.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.4 Functional Componnet4 File Manipulation

3.5.4.1 PDF to TEXT

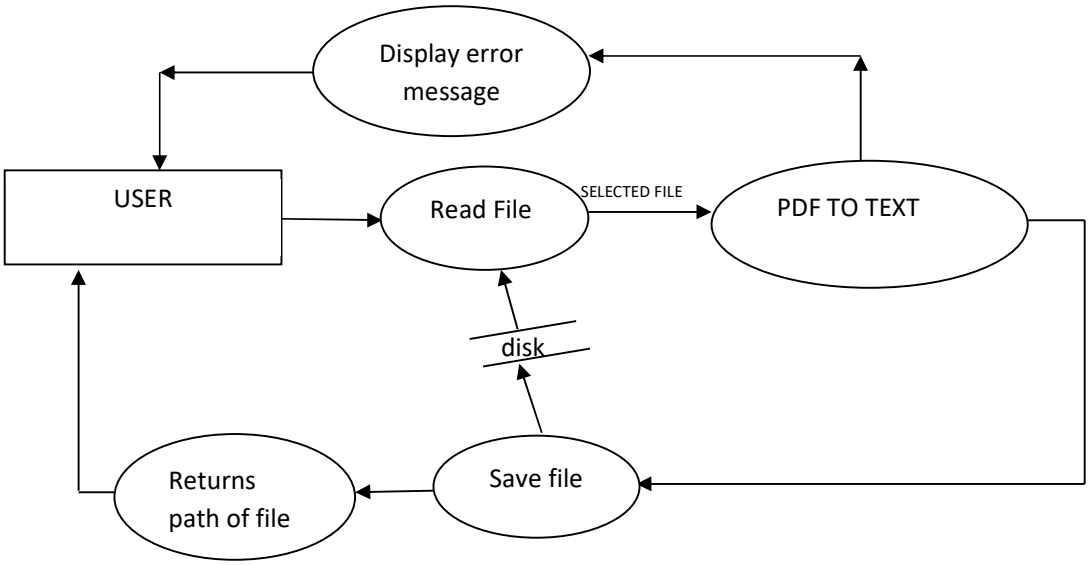


Fig no: 3.20 PDF to TEXT

3.5.4.1.1 Input

User Provide the PDF file as input.

3.5.4.1.2 Process Definition

Application will converts into text file.

3.5.4.1.3 Output

Application will produce the text file as output and create a text file in particular folder.

3.5.4.1.4 Interface with other functional component

Independent module

3.5.4.1.5 Resource Allocation

System Internal Storage

3.5.4.1.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.4.2 DCOX to TEXT

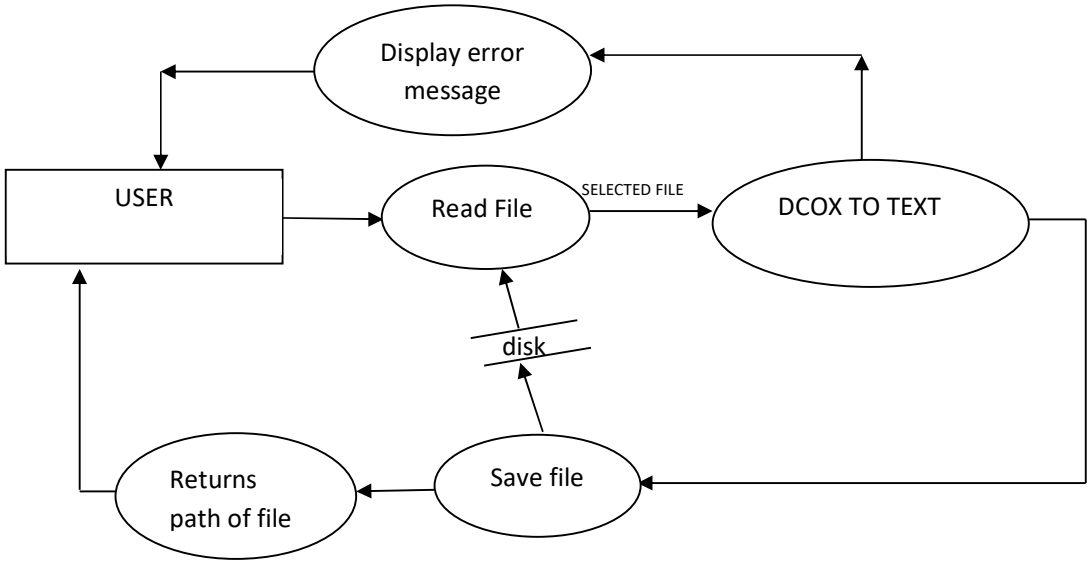


Fig no: 3.21 DCOX to TEXT

3.5.4.2.1 Input

User Provide the word file as input to this application.

3.5.4.2.2 Process Definition

Application process the word file and converts to text file format.

3.5.4.2.3 Output

Application will creates the text file in particular folder.

3.5.4.2.4 Interface with other functional component

Independent module

3.5.4.2.5 Resource Allocation

System Internal Storage

3.5.4.2.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.4.3 Pdf to Docx

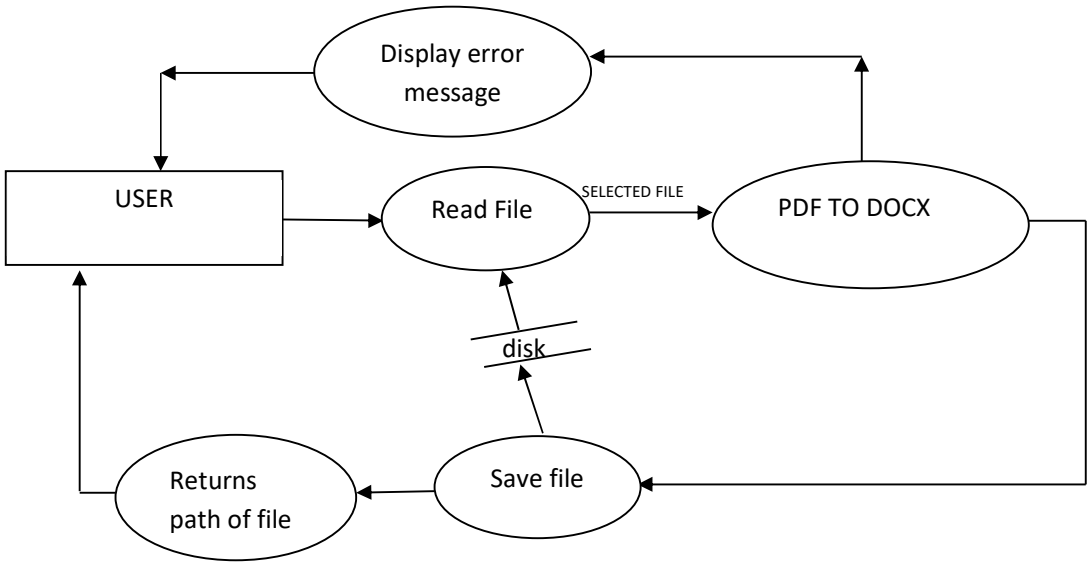


Fig no: 3.22 PDF to TEXT

3.5.4.3.1 Input

User Provide the pdf file as input.

3.5.4.3.2 Process Definition

Application converts into docx format.

3.5.4.3.3 Output

Application will create the docx file in particular folder.

3.5.4.3.4 Interface with other functional component

Independent module

3.5.4.3.5 Resource Allocation

System Internal Storage

3.5.4.3.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

3.5.4.4 TEXT to DOCX

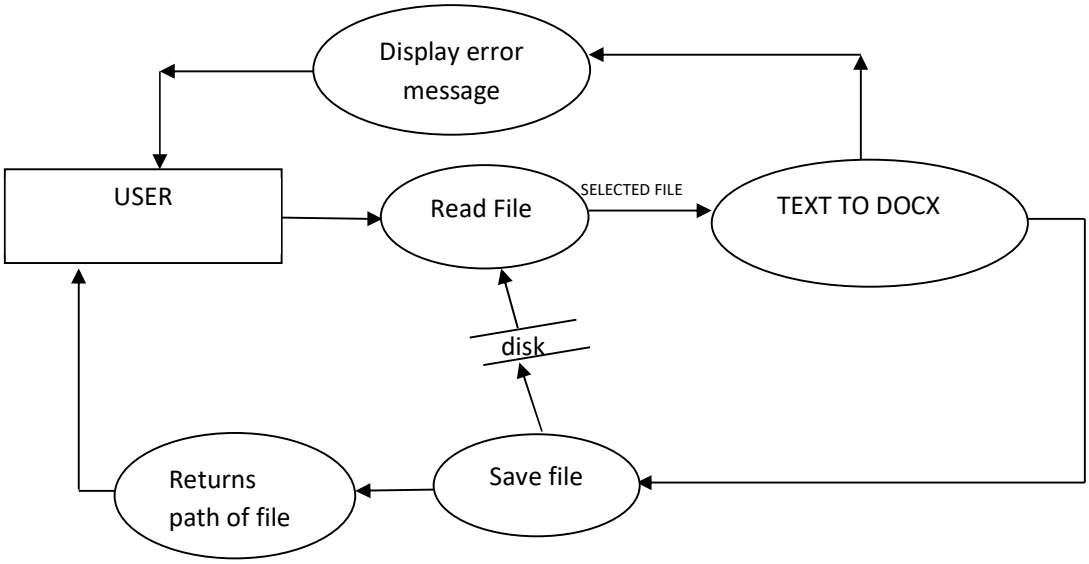


Fig no: 3.23 TEXT TO DOCX

3.5.4.4.1 Input

User Provide the text file as input.

3.5.4.4.2 Process Definition

Application converts into Dcox format.

3.5.4.4.3 Output

Application will create the Dcox file in particular folder.

3.5.4.4.4 Interface with other functional component

Independent module

3.5.4.4.5 Resource Allocation

System Internal Storage

3.5.4.4.6 User interface

Buttons , Frames ,Entry box, Message box, Label.

4.DETAILED DESIGN

4.1 Introduction:

During detailed design, the internal logic of each modules specified in system design is decided. During this phase further details of the modules are decided. Design of each of the modules usually specified in a high level description language which is independent of the language in which software eventually be implemented.

4.2 Structure of software package:

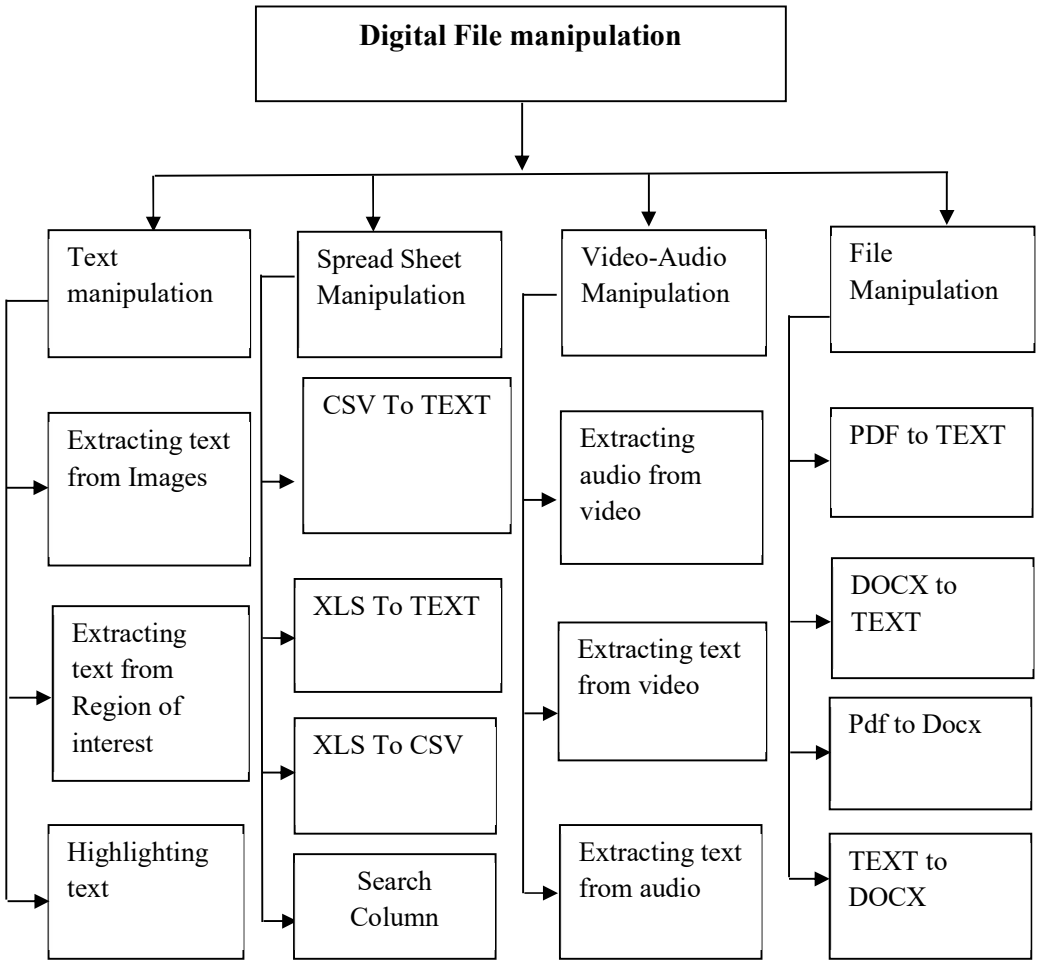


Fig no: 4.1 Structure of Software Package

4.3 Module decomposition of software:

Structure chart:

Structure chart is a top-down modular design, consist of squares representing different models in a system and lines .Structure chart shows how program has been partitioned into manageable modules hierarchy and organization of those modules and communicational interface.

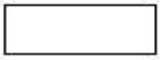
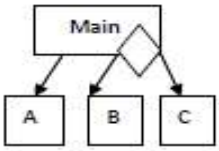
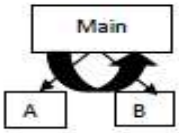
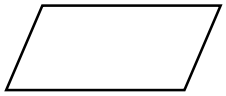
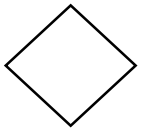
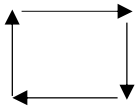
Symbol	Name	Process
	Data flow	Show the direction flow of data.
	Control flow	Shows the direction of flow control.
	Processing	Shows manipulation, calculation and processing.
	Module Invocation	It represent subordinate module being invoked by superior ordinate module.
	Condition invocation	It indicates that the invocation of subordinates Module depends on the evaluation of a condition.
	Iteration	It represent the iteration

Table no: 4.1 Structure Chart

Flow chart:

Flow chart is a graphical representation of solution to the given problems. A Flowchart is pictorial representation of an algorithm, workflow or process. The diagrammatic representation illustrates a solution model to given problem. It uses the following symbol.

Symbol	Name	Purpose
	Terminator	It indicates the start and end process
	Input/Output	Input/Output data.
	Decision	It represents a comparison or question that determines an alternate path to be followed.
	Flow direction	Shows the direction of data flow.
	Processing	It represents manipulation, calculation, or information processing.
	Direction access storage	File storage.

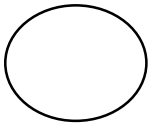
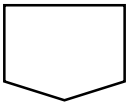
	Preparation(Looping)	An instruction or group of instruction.
	In-page	
	Off-page	
	Delay	

Table no: 4.2 Flow Chart

Chart

4.3.1 Text manipulation:

4.3.1.1 Extracting Text from image:

4.3.1.1.1 Input:

Single Image with jpg, jpeg, png, webp format from disk.

4.3.1.1.2 Procedural details:

Flow Chart:

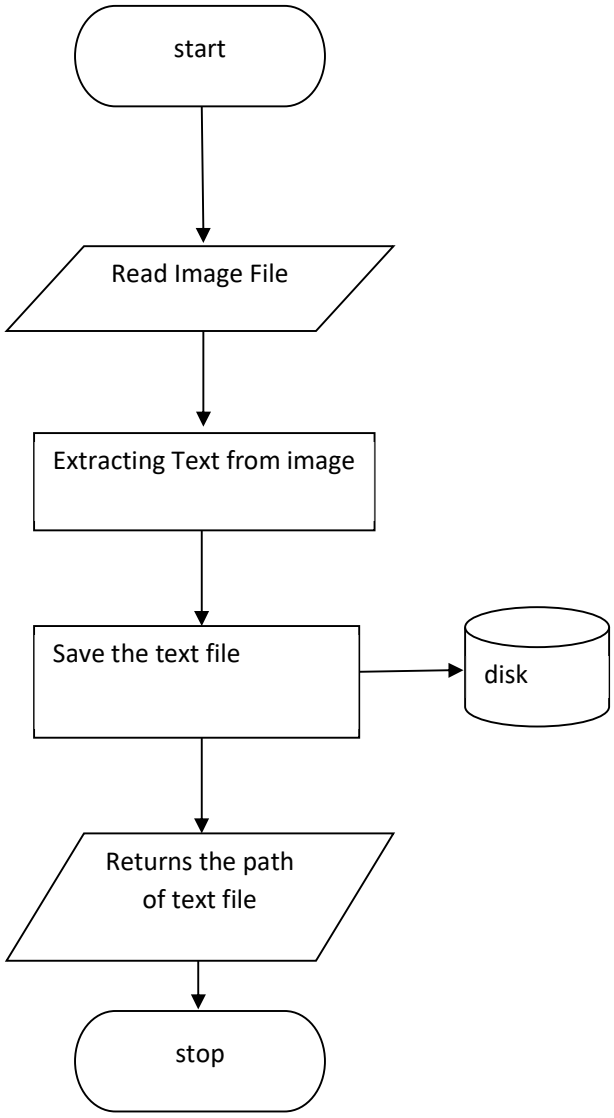


Fig no: 4.2 Flow Chart of Extracting text from image

Structure Chart:

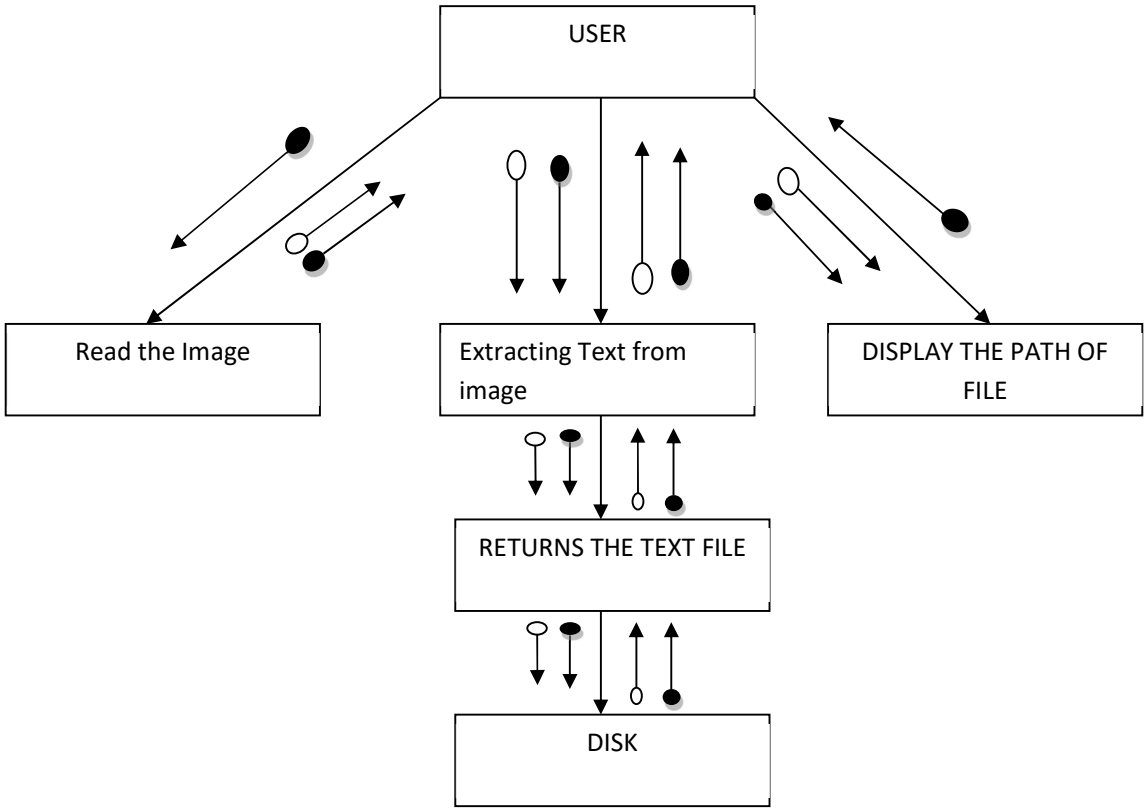


Fig no: 4.3 Structure Chart of Extracting text from image

4.3.1.1.3 File I/O interfaces

Not applicable

4.3.1.1.4 Outputs

Extracted text file.

4.3.1.1.5 Implementation aspects

Not applicable.

4.3.1.2 Extracting Region of interest:

4.3.1.2.1 Input:

Single Image with jpg, jpeg, png, webp format from disk.

4.3.1.2.2 Procedural details:

Flow Chart

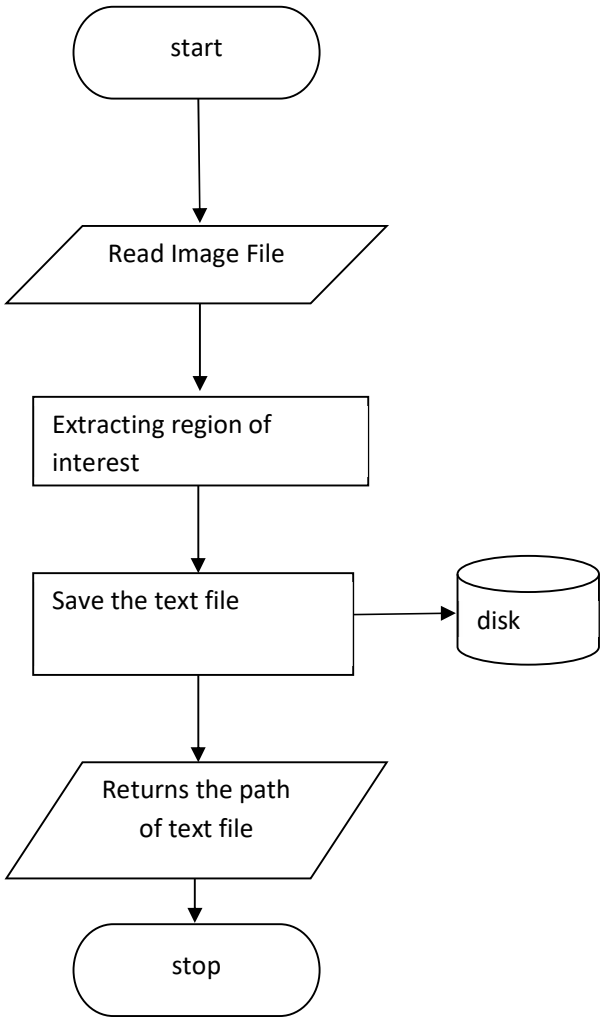


Fig no: 4.4 Flow Chart of Extracting text from Region of interest

Structure Chart:

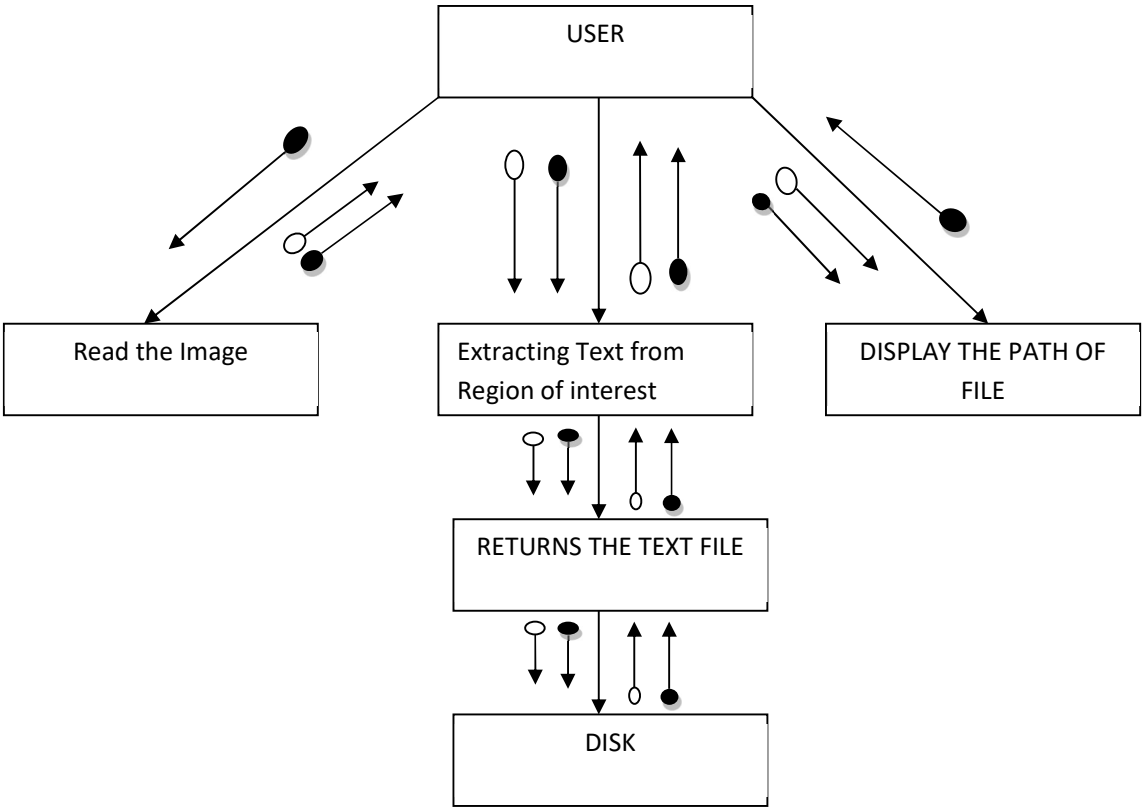


Fig no: 4.5 Structure Chart of Extracting text from Region of

4.3.1.2.3 File I/O interfaces

Not applicable

4.3.1.2.4 Outputs

Extracted text file.

4.3.1.2.5 Implementation aspects

Not applicable.

4.3.1.3 Highlighting text from Image:

4.3.1.3.1 Input:

Single Image with jpg, jpeg, png, webp format from disk.

4.3.1.3.2 Procedural details:

Flow Chart:

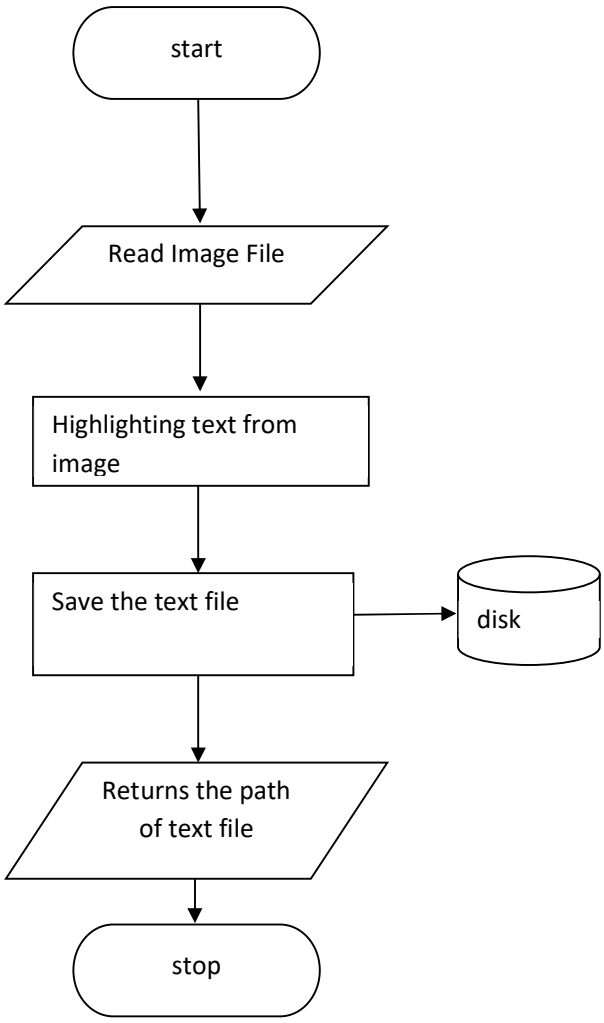


Fig no: 4.6 Flow Chart Highlighting Text From image

Structure Chart:

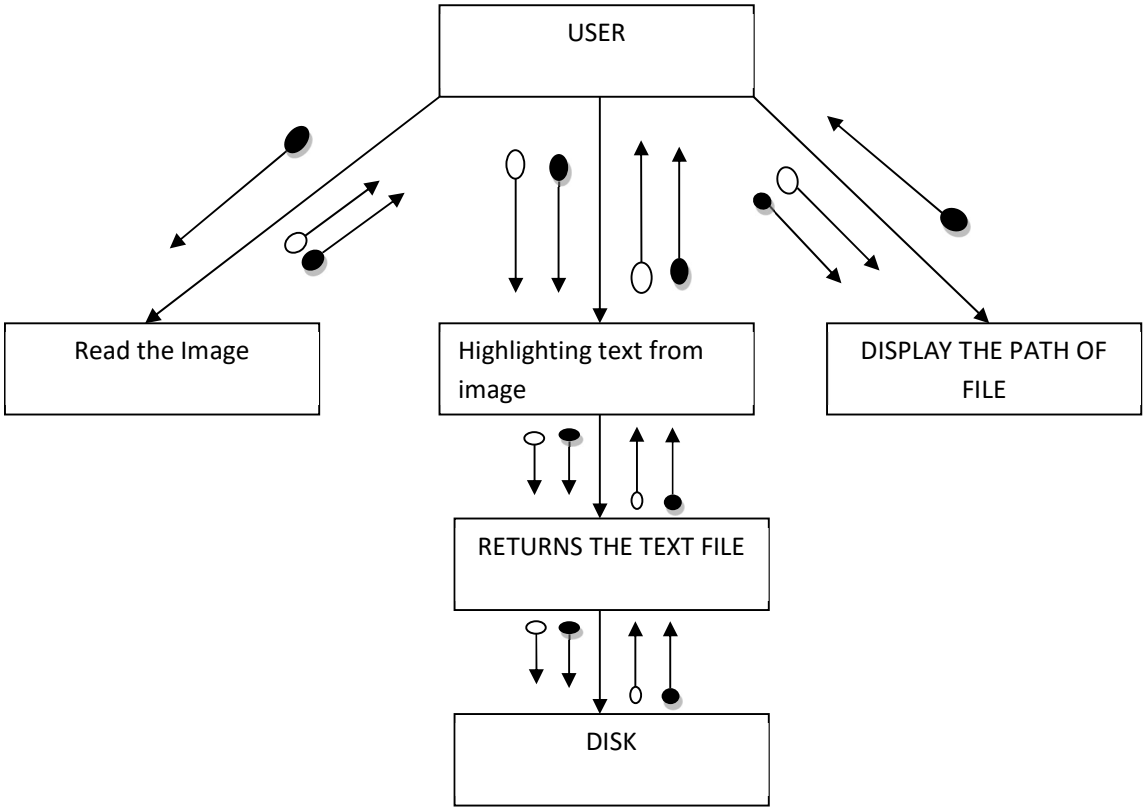


Fig no: 4.7 Structure Chart Highlighting Text From image

4.3.1.3.3 File I/O interfaces

Not applicable

4.3.1.3.4 Outputs

Extracted text file.

4.3.1.3.5 Implementation aspects

Not applicable.

4.3.2. Spreadsheet manipulation:

4.3.2.1 CSV to TEXT:

4.3.2.1.1 Input:

Single .csv format file from disk.

4.3.2.1.2 Procedural details:

Flow chart:

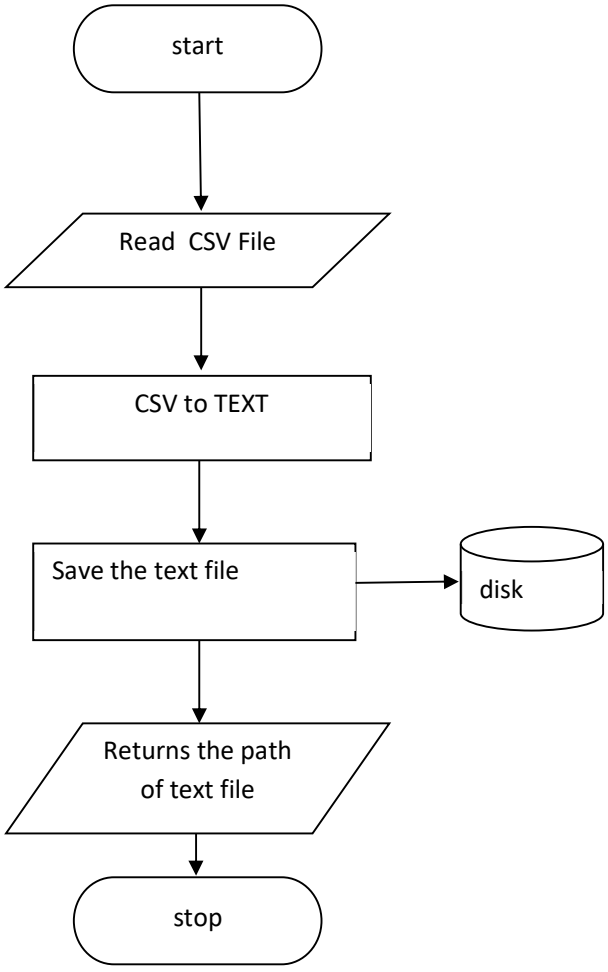


Fig no: 4.8 Flow Chart CSV TO TEXT

Structure Chart:

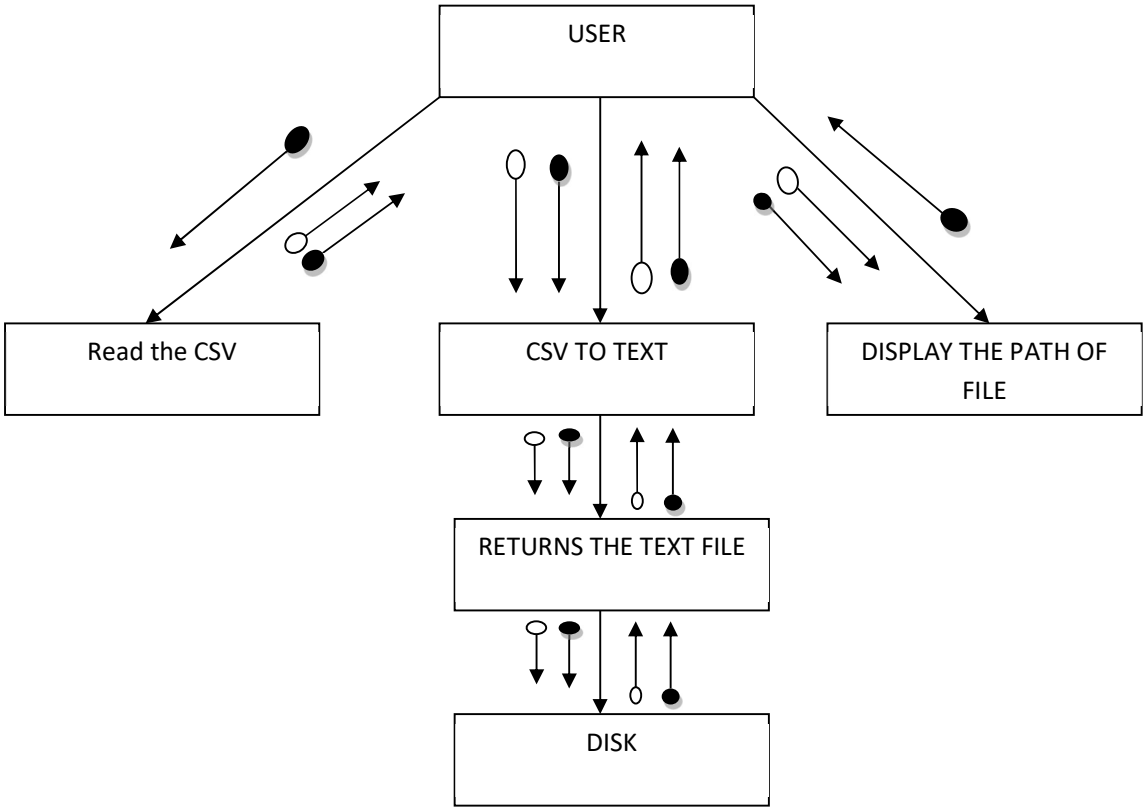


Fig no: 4.89 Structure Chart CSV TO TEXT

4.3.2.1.3 File I/O interfaces

Not applicable

4.3.2.1.4 Outputs

Extracted text file from csv file.

4.3.2.1.5 Implementation aspects

Not applicable.

4.3.2.2 XLS to TEXT:

4.3.2.2.1 Input:

Single .xlsx format file from disk.

4.3.2.2.2 Procedural details:

Flow chart:

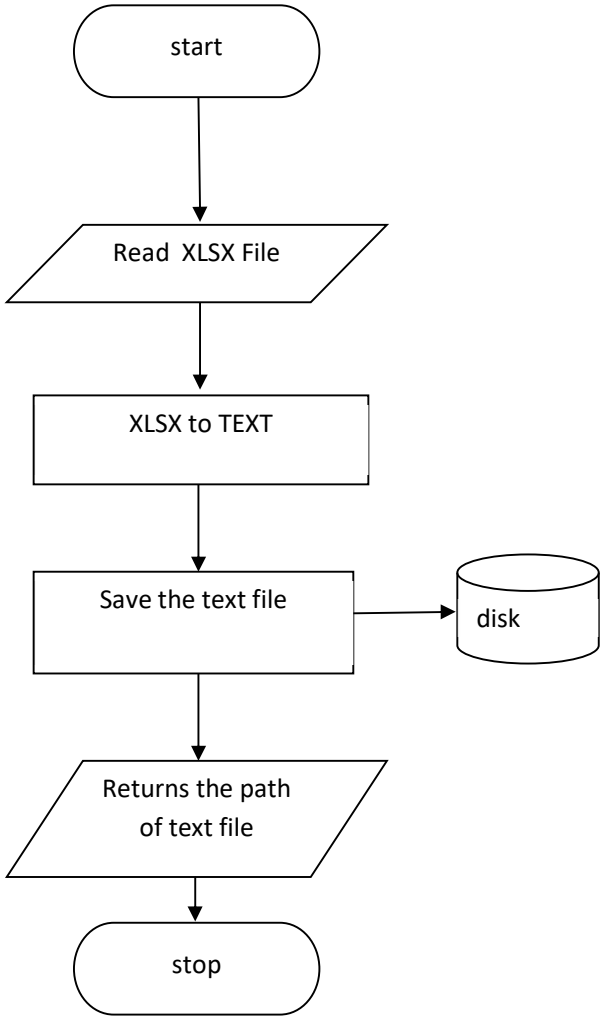


Fig no: 4.10 Flow Chart XLSX TO TEXT

Structure Chart:

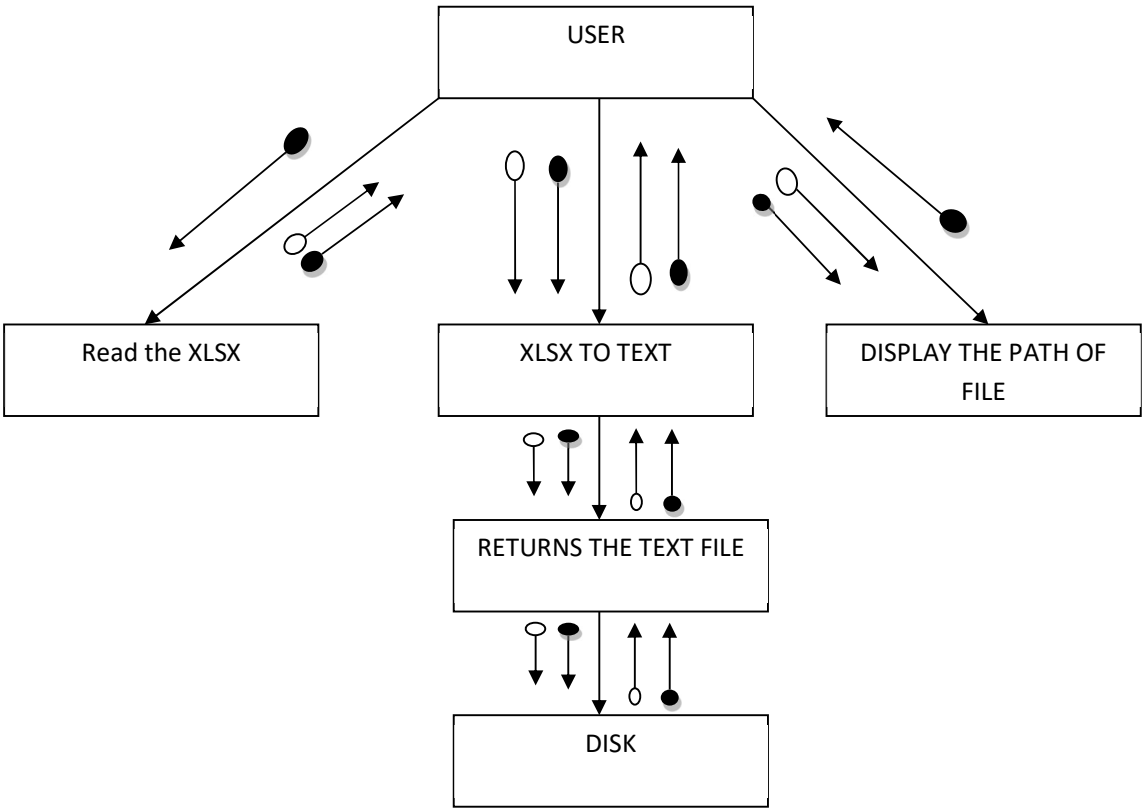


Fig no: 4.11 Structure Chart XLSX TO TEXT

4.3.2.2.3 File I/O interfaces

Not applicable

4.3.2.2.4 Outputs

Extracted text file from xlsx file.

4.3.2.2.5 Implementation aspects

Not applicable

4.3.2.3 XLS to CSV:

4.3.2.3.1 Input:

Single .xlsx format file from disk.

4.3.2.3.2 Procedural details:

Flow chart:

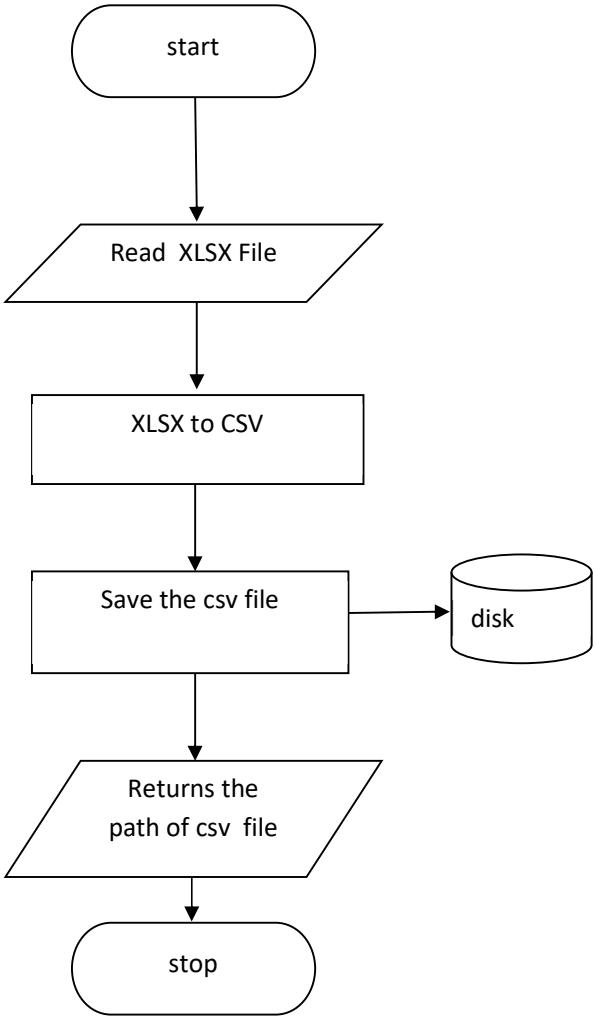


Fig no: 4.12 Flow Chart XLSX TO CSV

Structure Chart:

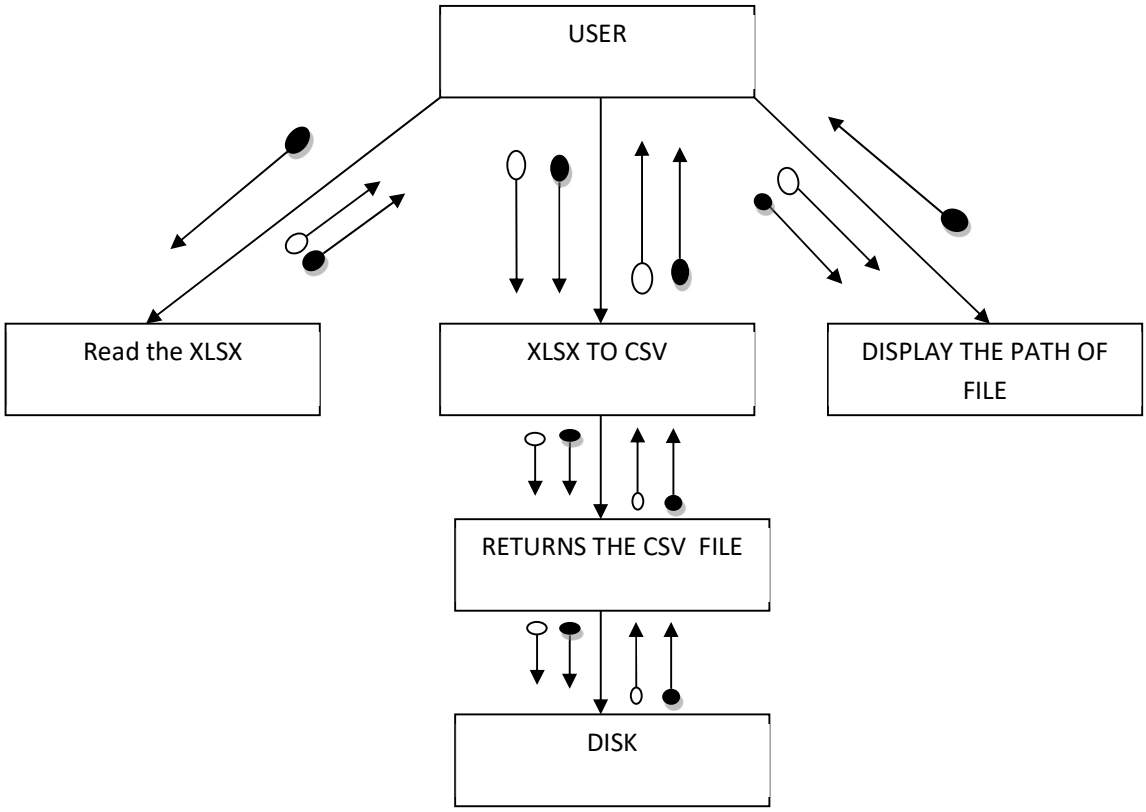


Fig no: 4.13 Structure Chart XLSX TO CSV

4.3.2.3.3 File I/O interfaces

Not applicable

4.3.2.3.4 Outputs

Extracted csv file from xlsx file.

4.3.2.3.5 Implementation aspects

Not applicable

4.3.2.4 Search Column:

4.3.2.4.1 Input:

Single .xlsx format file from disk.

4.3.2.4.2 Procedural details:

Flow chart:

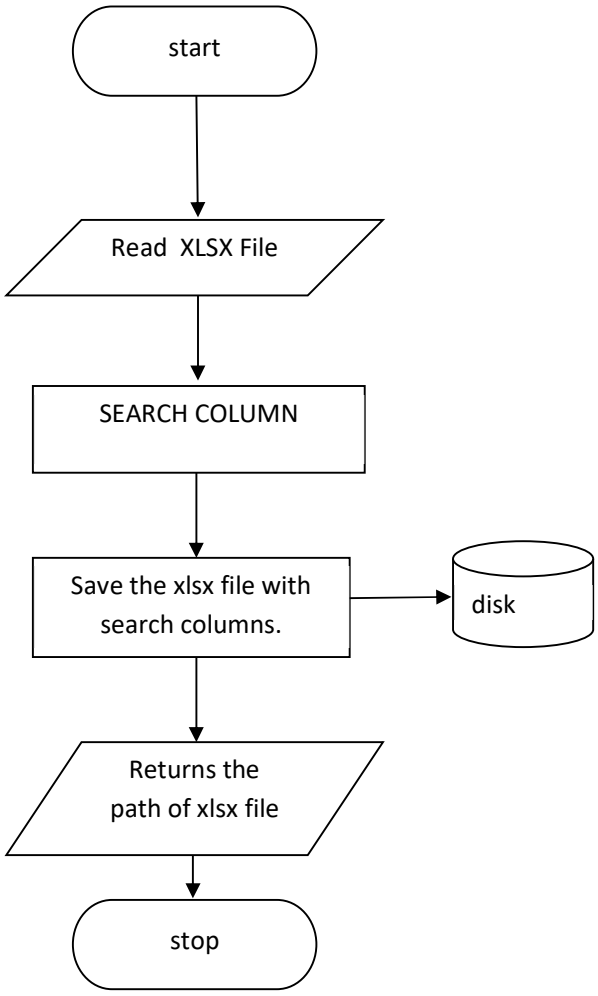


Fig no: 4.14 Flow Chart Search Column

Structure Chart:

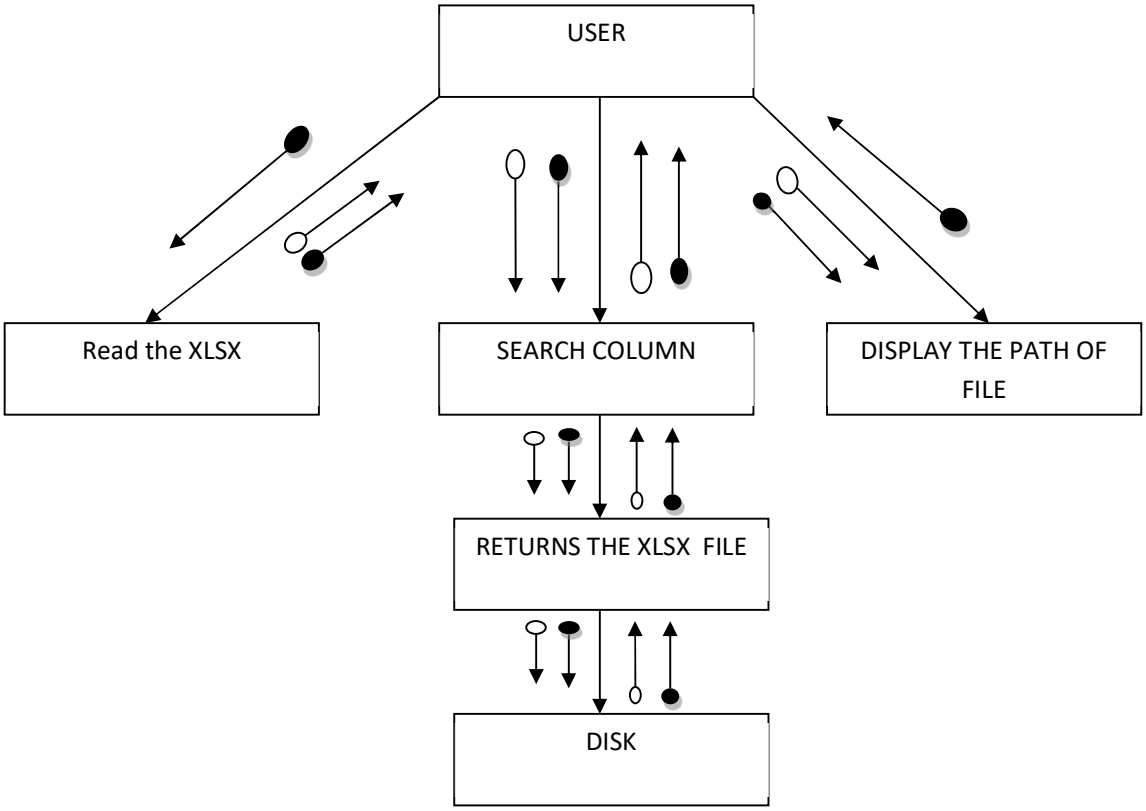


Fig no: 4.15 Structure Chart Search Column

4.3.2.4.3 File I/O interfaces

Not applicable

4.3.2.4.4 Outputs

Saved xlsx file with searched columns.

4.3.2.4.5 Implementation aspects

Not applicable

4.3.3. Audio and Video manipulation:

4.3.3.1 Extracting audio from Video:

4.3.2.1.1 Input:

Single video file from disk.

4.3.2.1.2 Procedural details:

Algorithm:

Step 1: start

Step 2: input video file

Step 3: extracting the audio from video

Step 4: save the audio file to disk

Step 5: Returns the path of stored audio file

Step 6: stop

Flow Chart:

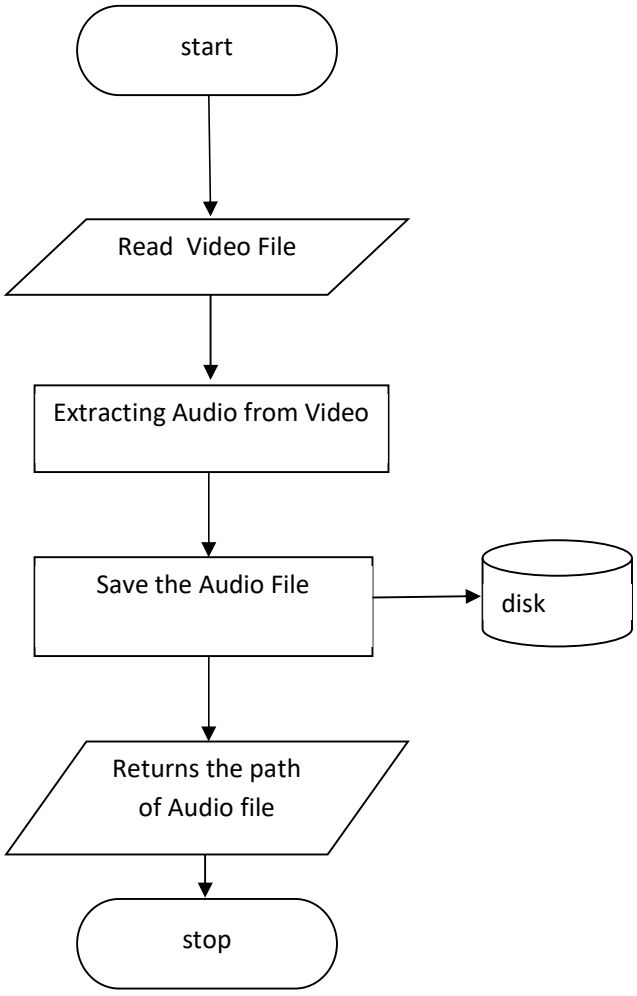


Fig no: 4.16 Flow Chart Extracting Audio From Video

4.3.2.1.3 File I/O interfaces

Not applicable

4.3.2.1.4 Outputs

Extracted audio file from video..

4.3.2.1.5 Implementation aspects

Not applicable

4.3.3.2 Text To Audio:

4.3.2.2.1 Input:

Single text file from disk.

4.3.2.2.2 Procedural details:

Algorithm:

Step 1: start

Step 2: input text file

Step 3: converting text to audio

Step 4: save the audio file to disk

Step 5: Returns the path of stored text file

Step 6: stop

Flow Chart:

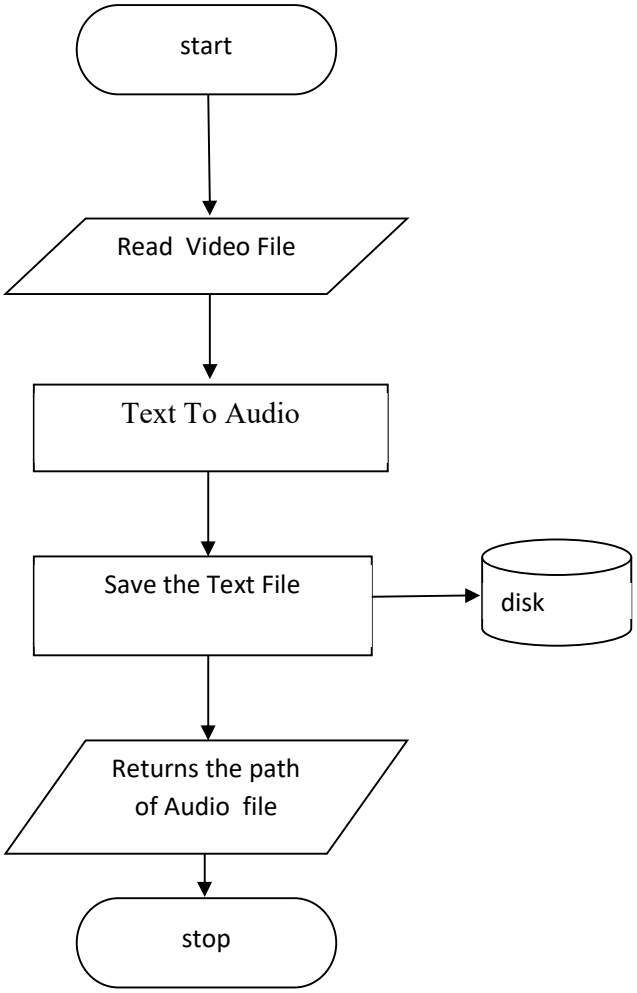


Fig no: 4.17 Flow Chart Text To Audio

4.3.2.2.3 File I/O interfaces

Not applicable

4.3.2.2.4 Outputs

Converted Audio file.

4.3.2.2.5 Implementation aspects

Not applicable

4.3.3.3 Trim Audio:

4.3.3.1.1 Input:

Single audio file from disk.

4.3.3.1.2 Procedural details:

Algorithm:

Step 1: start

Step 2: input audio file, start and end time.

Step 3: cropping the audio

¿ Step 4: save the audio file to disk

Step 5: Returns the path of stored audio file

Step 6: stop

Flow Chart:

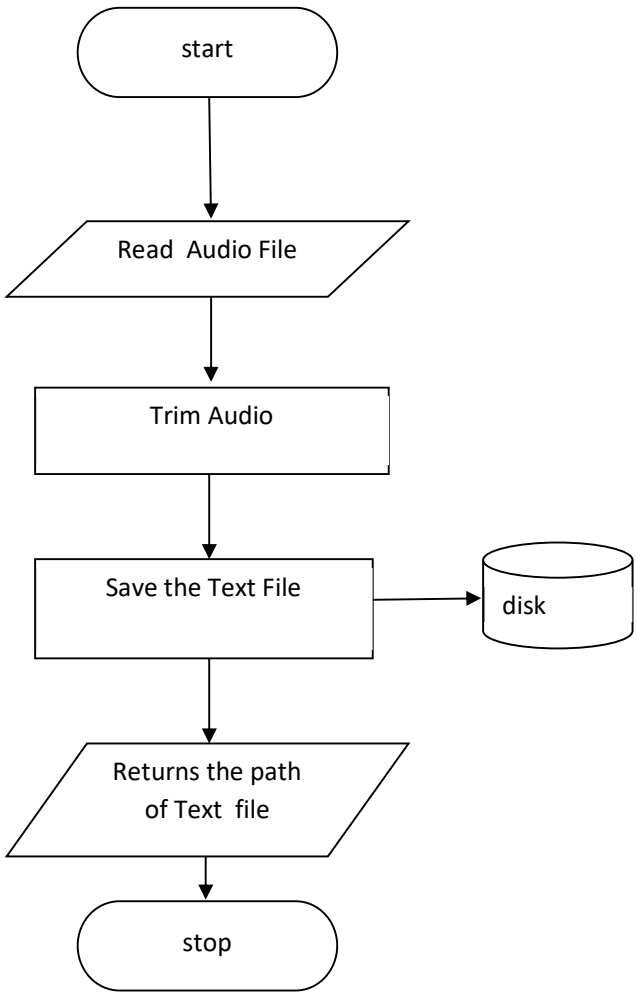


Fig no: 4.18 Flow Chart Trim Audio

4.3.3.1.3 File I/O interfaces

Not applicable

4.3.3.1.4 Outputs

Extracted audio file from audio.

4.3.3.1.5 Implementation aspects

Not applicable

4.3.4 File Manipulation

4.3.4.1 PDF TO TEXT :

4.3.4.1.1 Input:

User provide the single pdf file from disk.

4.3.4.1.2 Procedural details:

Structure Chart:

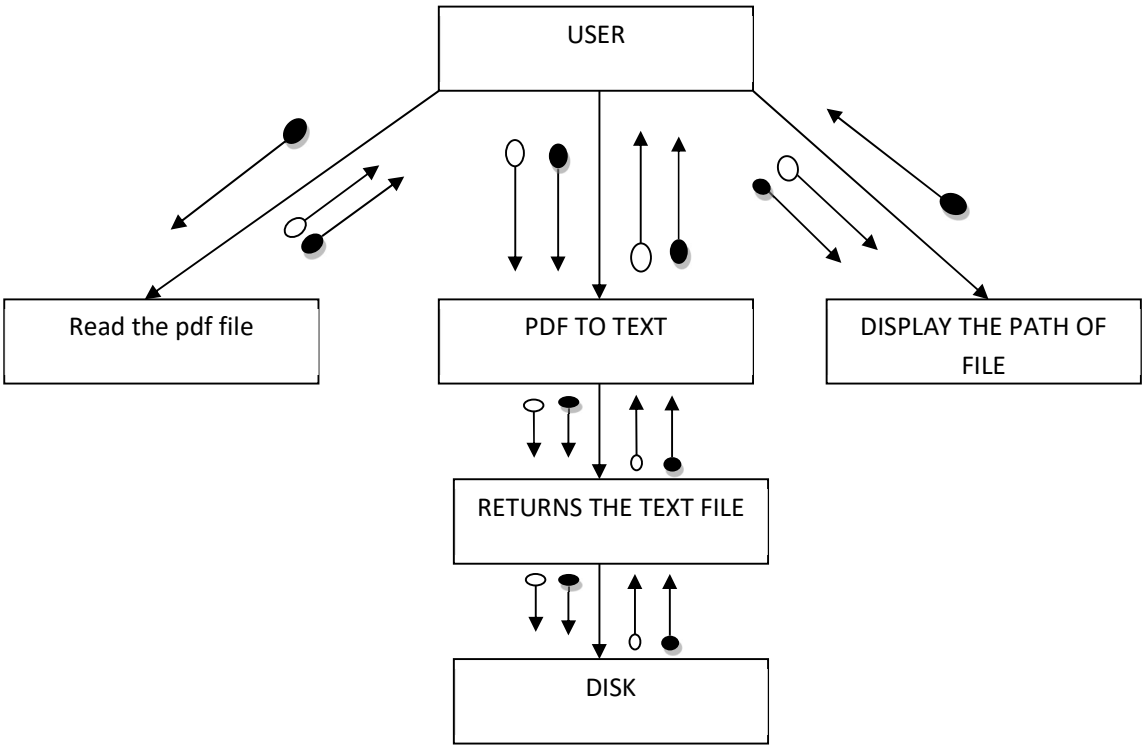


Fig no: 4.19 Structure Chart PDF TO TEXT

Flow Chart:

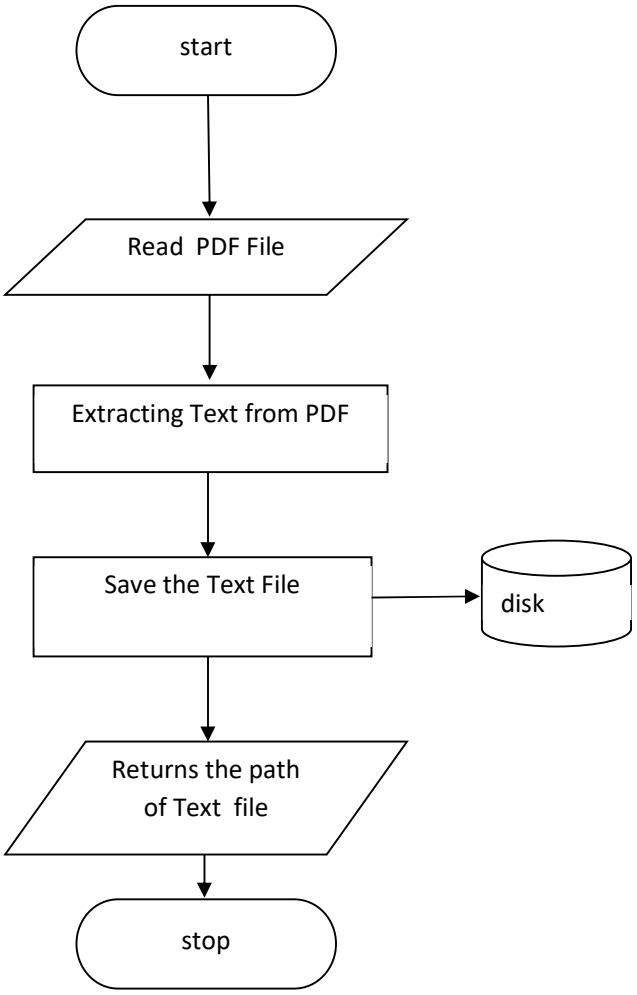


Fig no: 4.20 Flow Chart PDF TO TEXT

4.3.4.1.3 File I/O interfaces

Not applicable

4.3.4.1.4 Outputs

Converted text file from pdf file

4.3.4.1.5 Implementation aspects

Not applicable

4.3.4.2 DOCX TO TEXT :

4.3.4.2.1 Input:

User provide the single DOCX file..

4.3.4.2.2 Procedural details:

Structure Chart:

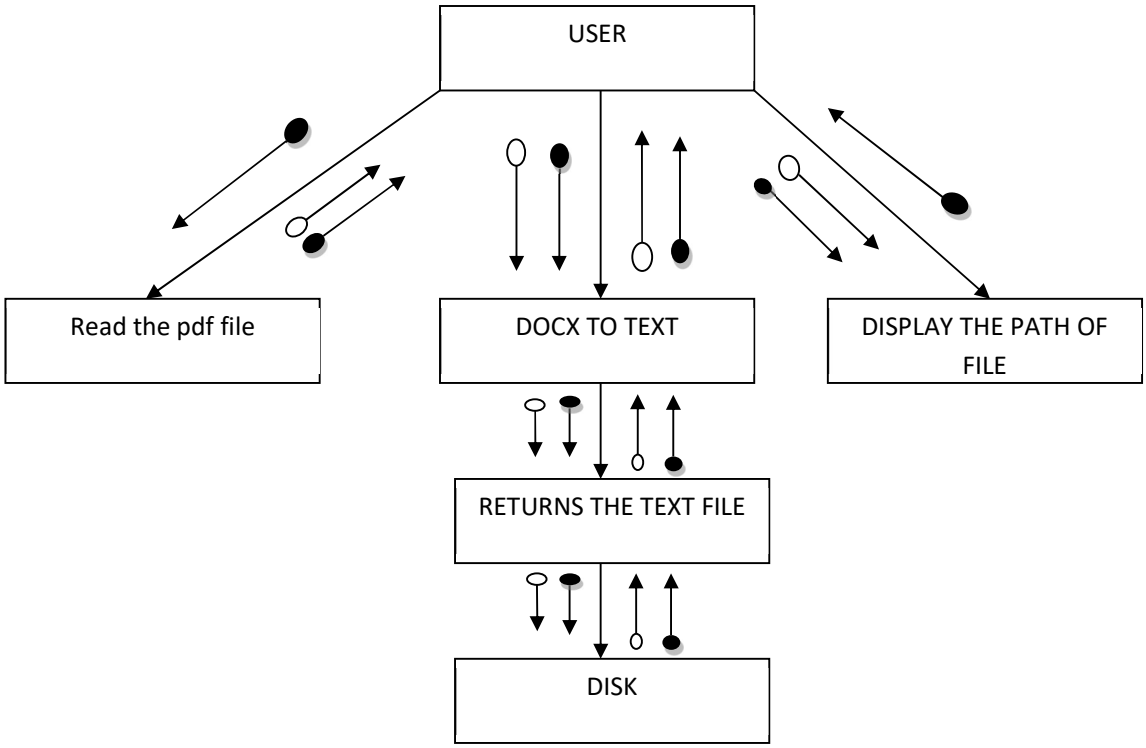


Fig no: 4.21 Structure Chart DOCX TO TEXT

Flow Chart:

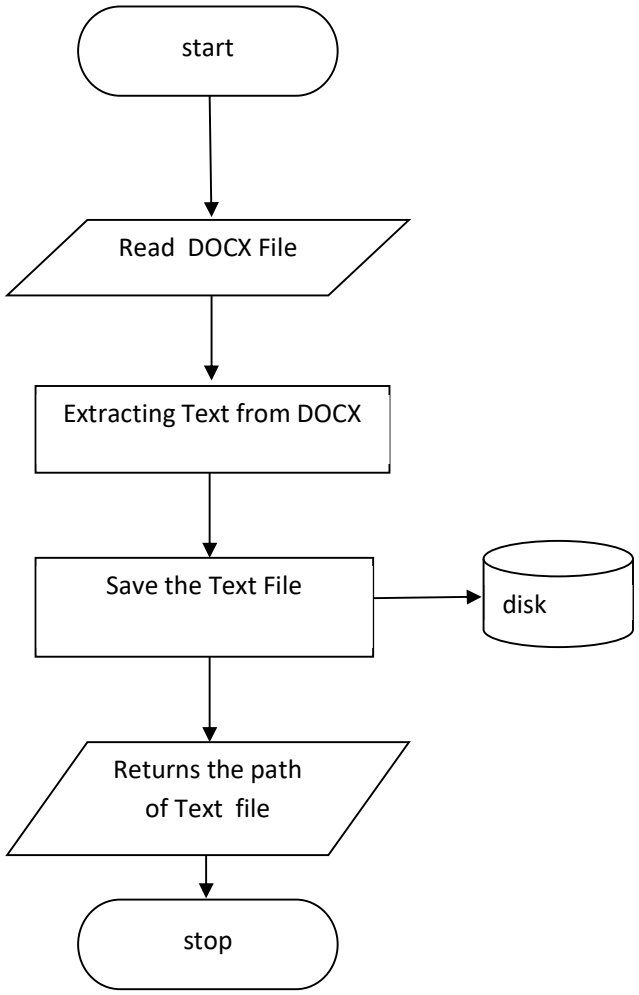


Fig no: 4.22 Flow Chart DOCX TO TEXT

4.3.4.2.3 File I/O interfaces

Not applicable

4.3.4.2.4 Outputs

Converted text file from docx file

4.3.4.2.5 Implementation aspects

Not applicable

4.3.4.3 Pdf to Docx:

4.3.4.3 .1 Input:

User provide the single Pdf file..

4.3.4.3.2 Procedural details:

Structure Chart:

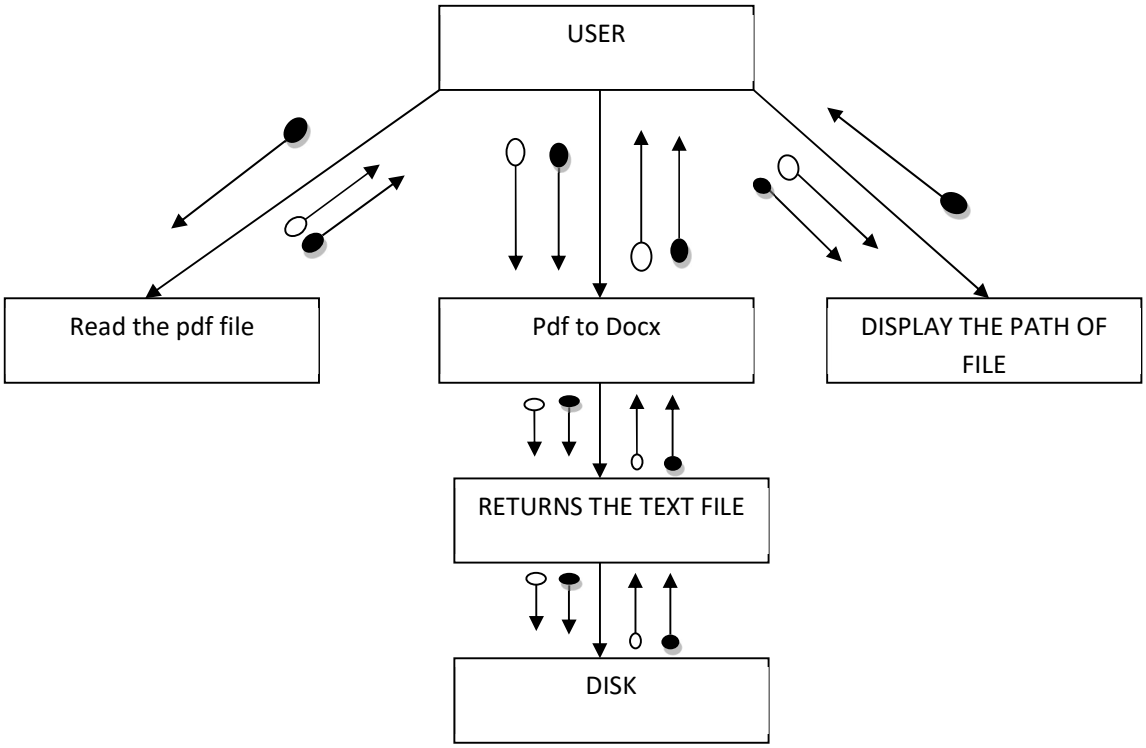


Fig no: 4.23 Structure Chart PDF TO DOCX

Flow Chart:

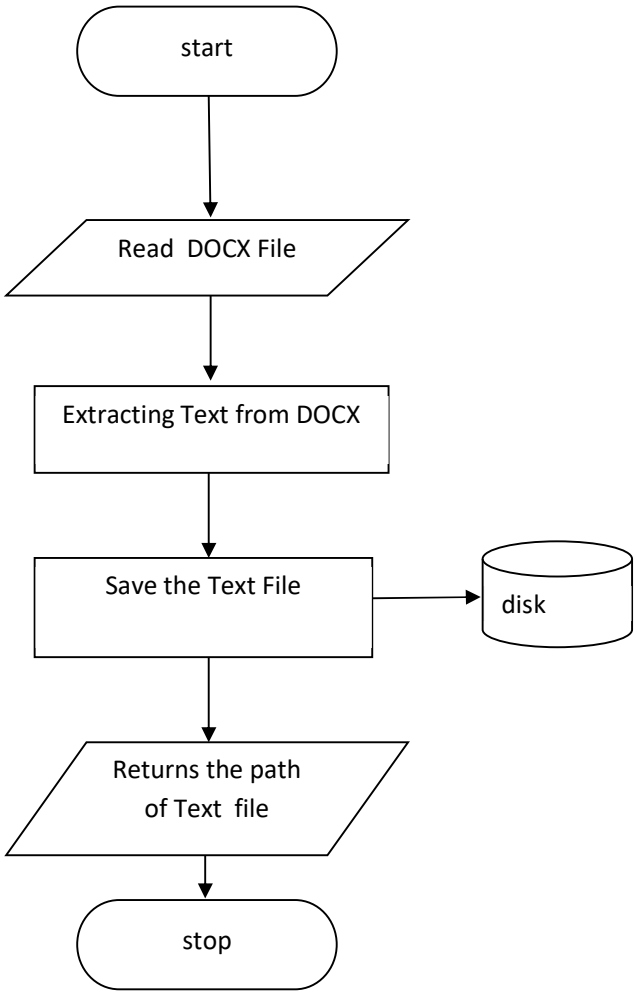


Fig no: 4.24 Flow Chart PDF TO DOCX

4.3.4.3.3 File I/O interfaces

Not applicable

4.3.4.3.4 Outputs

Converted Docx file from Pdf file.

4.3.4.3.5 Implementation aspects

Not applicable

4.3.4.4 TEXT TO DOCX:

4.3.4.4 .1 Input:

User provide the single Text file..

4.3.4.4.2 Procedural details:

Structure Chart :

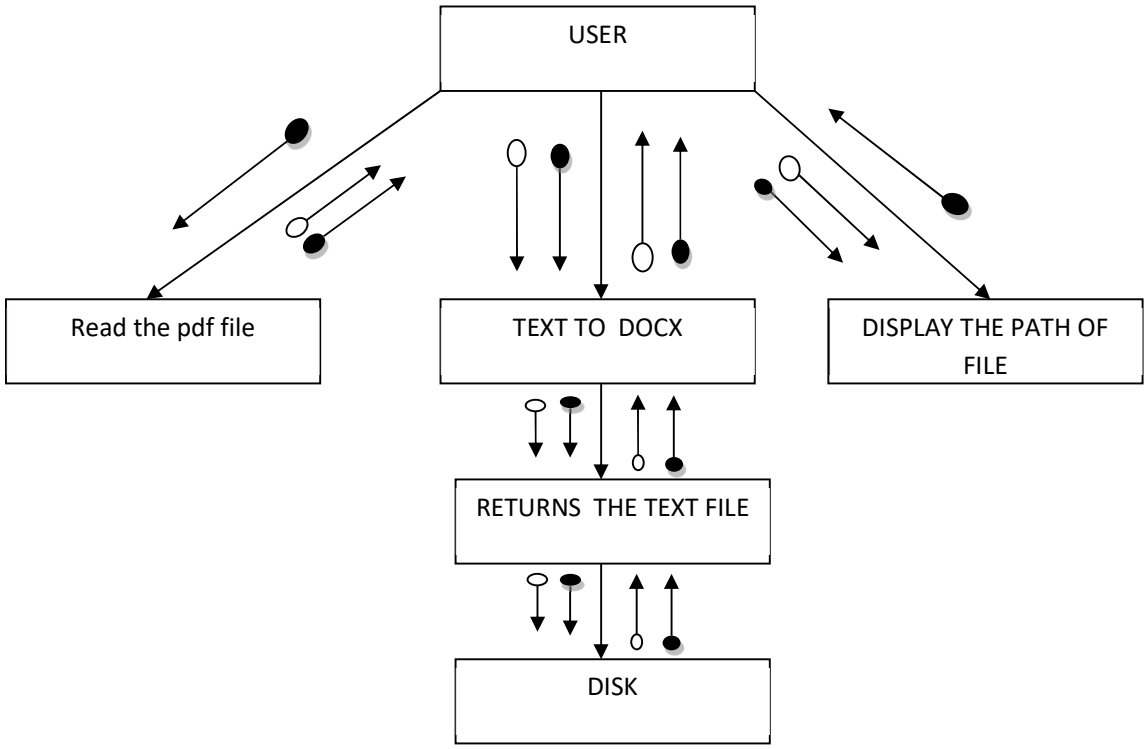


Fig no: 4.25 Structure Chart TEXT TO DOCX

Flow Chart:

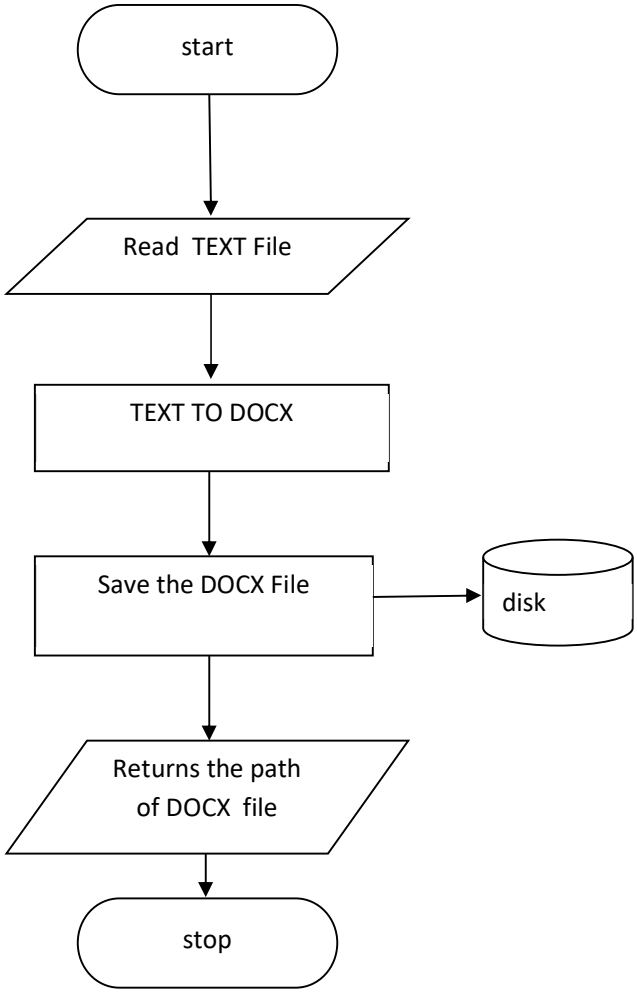


Fig no: 4.26 Flow Chart TEXT TO DOCX

4.3.4.4.3 File I/O interfaces

Not applicable

4.3.4.4.4 Outputs

Converted docx file from text file.

4.3.4.4.5 Implementation aspects

Not applicable

5. Program code listing

```
from textwrap import indent

from tkinter import Button,Frame,Label,StringVar,Toplevel

from tkinter import Entry,Tk,filedialog,messagebox,ttk

from tkinter import FLAT,SE,SOLID,SUNKEN,SW,CENTER,END,LEFT

from tkinter.filedialog import asksaveasfile

from PIL import ImageTk, Image

from winsound import MessageBeep,MB_ICONHAND,MB_OK

from os import startfile

from textmanipulation import *

from spreadsheetmanipulation import *

from audiovideomanipulation import *

from filemanipulation import *

class Design:

    def __init__(self, master):

        self.designwindow = master

        self.height = self.designwindow.winfo_screenheight()

        self.width = self.designwindow.winfo_screenwidth()

        self.designwindow.geometry("%dx%d+%d+%d" % (self.width, self.height,0, 0 ))

        if self.width < 720 and self.height < 1000:

            messagebox.showerror("Screen Resolution Problem","Sorry for the inconvienence
File can't be run on your system")

else:

            self.designwindow.title("Digital File Manipulation")
```

```
        self.designwindow.configure(bg="black")

        self.frame1 = Frame(self.designwindow, height=self.height,

         width=self.width , bg="black")

    self.frame1.pack(side=LEFT)

    front_window_img = Image.open("./res/frontlogo.jpg")

    front_window_img =front_window_img.resize((self.width, self.height)) # two
bracket for image sampling

    front_window_img = ImageTk.PhotoImage(front_window_img)

    label_img = Label(self.frame1, image=front_window_img)

    label_img.pack()

    label_img.img = front_window_img

    label_img.config(image=label_img.img)

    text_manipulation_button = Button(self.designwindow, width=30,height=3,
text="Text Manipulation in an Image",fg="white", bg="black", font=('arial', 12, 'bold'),
borderwidth=8,command=self.text_manipulation,activebackground="black",activeforegr
ound="white")

text_manipulation_button .place(relx=0.75,rely=0.02)

spreadsheet_manipulation_button = Button(self.designwindow, width=30, height=3,
text="spreadsheet manipulation",fg="white", bg="black", font=('arial', 12, 'bold'),
borderwidth=8,command=self.spreadsheet_manipulate,activebackground="black",activef
oreground="white")




spreadsheet_manipulation_button.place(relx=0.75,rely=0.22)
```

```
 video_audio__manipulation_button = Button(self.designwindow, width=30, height=3,
text="Video Audio Manipulation",fg="white", bg="black", font=('arial', 12, 'bold'),
borderwidth=8,


command=self.video_audio_manipulation,activebackground="black",activeforeground="
white")

video_audio__manipulation_button.place(relx=0.75, rely=0.42)

file_manipulation_button = Button(self.designwindow, width=30, height=3, text="File
Manipulation",fg="white", bg="black", font=('arial', 12, 'bold'),
command=self.file_manipulate,


borderwidth=8,activebackground="black",activeforeground="white")

file_manipulation_button .place(relx=0.75, rely=0.62)

exit_button = Button(self.designwindow, width=30, height=3, text="EXIT", fg="white",
bg="red", font=('arial', 12, 'bold'), borderwidth=8, command=lambda:
self.designwindow.destroy(),activebackground="black",activeforeground="white")

 exit_button.place(relx=0.75, rely=0.82)

   def text_manipulation(self):

       self.module1 = Toplevel(self.designwindow)

       self.mod1 = Window1(self.module1, self.designwindow)

       self.designwindow.withdraw()

 def spreadsheet_manipulate(self):

       self.module2 = Toplevel(self.designwindow)

       self.mod2 = Window2(self.module2, self.designwindow)

       self.designwindow.withdraw()


   def file_manipulate(self):
```

```
        self.module4 = Toplevel(self.designwindow)
        self.mod4 = Window4(self.module4, self.designwindow)
        self.designwindow.withdraw()

    def video_audio_manipulation(self):
        self.module3 = Toplevel(self.designwindow)
        self.mod3 = Window3(self.module3, self.designwindow)

  self.designwindow.withdraw()
class Window1:
    def __init__(self, master, mainwindow):
        self.view_file1=""
        self.view_file2=""
        self.view_file3=""
        self.var = StringVar()
        self.first_window = master
        self.first_window.title("Text Manipulation")

        self.design_window = mainwindow
        self.window1_height = self.design_window.winfo_screenheight()
        self.window1_width = self.design_window.winfo_screenwidth()
        self.window1_img = Image.open("./res/topleveltextmanipulation.jpg")
        self.window1_img = self.window1_img.resize((self.window1_width,
self.window1_height))
        self.window1_img = ImageTk.PhotoImage(self.window1_img)
```

```
label = Label(self.first_window, image=self.window1_img)

label.pack()

self.first_window.protocol("WM_DELETE_WINDOW",self.on_exit)

self.window1_img.img =self.window1_img

label.config(image=self.window1_img.img)

extracting_text_from_img_btn1 = Button(self.first_window, text="Extracting Text
from Image", width=30,height=3, font=('arial', 12, 'bold'), command=lambda:
self.upload_extracting_text_from_image(),bg='#FF005E', fg="white",borderwidth=8)



 extracting_text_from_img_btn1.place(relx=0.015,rely=0.2)

extracting_text_from_region_of_interest_btn2 = Button(

    self.first_window, text="Extracting Text from Region of Interest", width=30,
height=3,font=('arial', 12, 'bold'),bg='#FF005E', fg="white",

command=lambda:self.upload_extracting_text_from_region_of_interest(),
borderwidth=8)

extracting_text_from_region_of_interest_btn2.place(relx=0.015,rely=0.4)

highlighting_text_btn3 = Button(self.first_window, text="Highlighting Text ",
width=30, height=3, font=('arial', 12, 'bold'),  bg='#FF005E', fg="white",
command=lambda: self.upload_highlighting_text(), borderwidth=8)

highlighting_text_btn3.place(relx=0.015,rely=0.6)

exit_btn4 = Button(self.first_window, text="Exit", width=30, height=3,

command=self.exit_window, font=('arial', 12, 'bold'), bg='#0055FF', fg="white",

borderwidth=8)

exit_btn4.place(relx=0.015,rely=0.8)

self.window1_btn1_frame=Frame(self.first_window)

self.window1_btn2_frame=Frame(self.first_window)

self.window1_btn3_frame=Frame(self.first_window)
```

```
    def exit_window(self):
        self.design_window.deiconify()
        self.first_window.destroy()
    def on_exit(self):
        self.first_window.destroy()


            self.design_window.deiconify()



def upload_extracting_text_from_image(self):
        self.window1_btn2_frame.destroy()
        self.window1_btn3_frame.destroy()



            self.window1_btn1_frame=Frame(self.first_window,width=400,
height=400,bg="purple")
        self.window1_btn1_frame.place(relx=0.4, rely=0.2)
        self.window1_btn1_frame.config(relief=FLAT)
        # Image for button
        self.window1_btn1_img= Image.open("./res/file.png")
        self.window1_btn1_img = self.window1_btn1_img.resize((150, 120))
        self.window1_btn1_img = ImageTk.PhotoImage(self.window1_btn1_img)
        # upload button
        button1 = Button(self.window1_btn1_frame, text="Upload", width=150,
height=120, fg="white", font=('arial', 10, 'bold'), image=self.window1_btn1_img,
command=lambda: self.extracting_text_from_image(), bg="black")
```

```
        button1.place(relx=0.5,rely=0.2,anchor="center")

        self.window1_btn1_img.img = self.window1_btn1_img

        button1.config(image=self.window1_btn1_img.img)

        # label for upload

        label1=Label(self.window1_btn1_frame, text="Drop file here", font=('arial black',
12, 'bold'), fg="black", width=15, height=1,bg="purple")


 label1.place(relx=0.3,rely=0.4)

        # instruction button

        button2 = Button(self.window1_btn1_frame, text="How to use", width=10,
height=2, fg="white", font=('arial black', 12, 'bold'), bg="black", command=lambda:
self.text_extract_instruct())


            button2.place(rely=0.98, relx=0.98, x=0, y=0, anchor=SE)

        #view Nutton



            view_button1 = Button(self.window1_btn1_frame, text="view", width=10,
height=2, font=('arial black', 12, 'bold'),

                          bg="black", fg="white", activebackground="black",
activeforeground="white",

                          command=lambda: self.view_text_file())

        view_button1.place(relx=0.01, rely=0.82)


    def text_extract_instruct(self):

        width=(self.window1_width/2)-100
```

```
        height=(self.window1_height/2)-150
        font=("Comic Sans MS",20,"bold")
        msg="Upload the clear image as input and save the file in desired location application retrive the text associated with the image"
        msg_window=Toplevel(self.first_window)
        msg_window.geometry("%dx%d+%d+%d" %(500,300,width,height))

        msg_window.resizable(0,0
        msg_window.focus_set()
        msg_window.iconbitmap("./res/hint.ico")
        msg_window.title("Instruction")
        msg_label=Label(msg_window,text=msg,font=font,wraplength=450)
        msg_label.place(relx=0, rely=0.1)
        msg_window.mainloop()

    def extracting_text_from_image(self):
        try:
            self.first_window.grab_set()
            image_file = filedialog.askopenfilename(initialdir="c:\\", title="select a file", filetypes=(

             ("jpg", "*.jpg"), ("jpeg", "*.jpeg"), ("png", "*.png"), ("webp", "*.webp"),("svg","*.svg"),("raw","*.raw"),("xcf","*.xcf"),("jpx","*.jpx"),
```

```
("tiff","*.tiff"),("cr2","*.cr2"),("bmp","*.bmp"),("jxr","*.jxr"),("psd","*.psd"),("ico","*.ic
o"),("heic","*.heic"),("dcm","*.dcm")),parent=self.first_window)
        if image_file:
            saving_file = asksaveasfile(initialfile='Untitled.txt', defaultextension=".txt",
                                        filetypes=[("txt", "*.txt")],parent=self.first_window)
            if saving_file:
                saving_file.close()


        progressbar = ttk.Progressbar(self.first_window, length=400,
mode="indeterminate")
                progressbar.place_forget()
                location = saving_file.name
                progressbar.place(relx=0.4,rely=0.7)
                progressbar.start()
                output = image_to_text(image_file, location)
                if output["output"] != "":
                    progressbar.stop()
                    progressbar.destroy()
                    MessageBeep(type=MB_OK)
                    messagebox.showinfo("success", "file is saved..  \nin "+
output['output_file'],parent=self.first_window) #path of saved file is output_file
                    self.view_file1=output["output_file"]
                elif output["error"] != "":
                    MessageBeep(type=MB_ICONHAND)
                    progressbar.stop()
```

```
                progressbar.destroy()
                messagebox.showerror("info",output["error"],parent=self.first_window)
            else:
                    messagebox.askokcancel("warning", "please save the file for
                    accessing content",parent=self.first_window)


else:
                 messagebox.askokcancel("warning", "please select the image file for
        converting",parent=self.first_window)
    except Exception as e:
        messagebox.showerror("error", e)
  def view_text_file(self):
        if self.view_file1!="":
            startfile(self.view_file1)
            self.view_file1=""
        else:
            messagebox.askokcancel("warning", "Please  Convert the file before
viewing",parent=self.first_window)

  # 2nd button Extracting Text from region of interest
  # ************************Button2**********************
  def upload_extracting_text_from_region_of_interest(self):
      self.window1_btn1_frame.destroy()
      self.window1_btn3_frame.destroy()
self.window1_btn2_frame=Frame(self.first_window,width=400,height=400,bg="purple")
```

```
        self.window1_btn2_frame.place(relx=0.4, rely=0.2)

        self.window1_btn2_frame.config(relief=SOLID)

        self.window1_btn2_img = Image.open("./res/file.png")

        self.window1_btn2_img = self.window1_btn2_img.resize((150, 120))

        self.window1_btn2_img = ImageTk.PhotoImage(self.window1_btn2_img)


 button1 = Button(self.window1_btn2_frame, text="Upload", width=150, height=120,
fg="white", font=('arial', 10, 'bold'), image=self.window1_btn2_img, command=lambda:
self.extracting_text_from_region_of_interest, bg="black")


button1.place(relx=0.5, rely=0.2, anchor="center")

  self.window1_btn2_img.img = self.window1_btn2_img

   button1.config(image=self.window1_btn2_img.img)

  label1 = Label(self.window1_btn2_frame, text="Drop file here", font=('arial black', 12,
'bold'), bg="purple",fg="black", width=15, height=1)

        label1.place(relx=0.3, rely=0.4)

        button2 = Button(self.window1_btn2_frame, text="How to use", width=10,
height=2, fg="white",font=('arial black', 12, 'bold'), bg="black", command=lambda:
self.instruction_region_of_interest())

        button2.place(rely=0.98, relx=0.98, x=0, y=0, anchor=SE)

        view_button2 = Button(self.window1_btn2_frame, text="view", width=10,
height=2,
                              font=('arial black', 12, 'bold'), bg="black", fg="white",

                              activebackground="black", activeforeground="white",

                              command=lambda: self.region_text_file())

        view_button2.place(relx=0.01, rely=0.82)


    def instruction_region_of_interest(self):

        width=(self.window1_width/2)-100
```

```
        height=(self.window1_height/2)-150
        font=("Comic Sans MS",20,"bold")
        msg="upload the proper image and save the file in desired location then application pops up the image to select the region of interest" \
            "On  selecting the region  pressing 'Enter' key two times will save the file as png and 'c' button to cancel the selection "\




        msg_window=Toplevel(self.first_window)
        msg_window.geometry("%dx%d+%d+%d" %(500,400,width,height))
        msg_window.resizable(0,0)
        msg_window.focus_set()


            msg_window.iconbitmap("./res/hint.ico")
        msg_window.title("Instruction")
        msg_label=Label(msg_window,text=msg,font=font,wraplength=450)
        msg_label.place(relx=0, rely=0.1)
        msg_window.mainloop()


    def extracting_text_from_region_of_interest(self):
        try:
            self.first_window.grab_set()
            image_file = filedialog.askopenfilename(initialdir="c:\\", title="select a file", filetypes=(
                ("jpg", "*.jpg"), ("jpeg", "*.jpeg"), ("png", "*.png"), ("webp", "*.webp"),("svg","*.svg"),("raw","*.raw"),("xcf","*.xcf"),("jpx","*.jpx"),
```

```
("tiff","*.tiff"),("cr2","*.cr2"),("bmp","*.bmp"),("jxr","*.jxr"),("psd","*.psd"),("ico","*.ic
o"),("heic","*.heic"),("dcm","*.dcm")),parent=self.first_window)
        if image_file:
            saving_file = asksaveasfile(initialfile='Untitled.txt',
                                        defaultextension=".txt", filetypes=[("txt",
"*.txt")],parent=self.first_window)


  if saving_file:
                saving_file.close()
                location= saving_file.name
                output = text_from_region_of_interest(image_file, location)
                progressbar = ttk.Progressbar(self.first_window, length=400,
mode="indeterminate")
                progressbar.place(relx=0.4, rely=0.78)
                progressbar.start()
                if output["output"] != "":
                    progressbar.stop()
                    progressbar.destroy()
                    #success sound
                    MessageBeep(type=MB_OK)
                    messagebox.showinfo("success", "file is saved..  \nin "+
output['output_file'],parent=self.first_window)
                    self.view_file2=output["output_file"]
                elif output["error"] != "":
                    progressbar.stop()
```

```
                progressbar.destroy()

                #failed sound

                MessageBeep(type=MB_ICONHAND)

                messagebox.showerror("warning",
output["error"],parent=self.first_window)

            else:


 messagebox.askokcancel("warning", "please save the file for accessing
content",parent=self.first_window)

        else:

            messagebox.askokcancel("warning", "please select the image file for
converting",parent=self.first_window)

    except Exception as e:

        messagebox.showerror("error", e)

  def region_text_file(self):

    if self.view_file2!="":

        startfile(self.view_file2)

        self.view_file2 = ""

    else:

        messagebox.askokcancel("warning", "Please  Convert the file before
viewing",parent=self.first_window)


  # 3rd button Highlighting text

  #**********************Button3*************************

  def upload_highlighting_text(self):

    self.window1_btn1_frame.destroy()
```

```
        self.window1_btn2_frame.destroy()

        self.window1_btn3_frame=Frame(self.first_window,width=500, height=500,
bg="purple")

        self.window1_btn3_frame.place(relx=0.4, rely=0.2)

        self.window1_btn3_frame.config(relief=SUNKEN)

        self.window1_btn3_img = Image.open("./res/file.png")


 self.window1_btn3_img = self.window1_btn3_img.resize((150, 120))

        self.window1_btn3_img = ImageTk.PhotoImage(self.window1_btn3_img)

        button1 = Button(self.window1_btn3_frame, text="Upload", width=150,
height=120, fg="white", font=(

            'arial', 10, 'bold'), image=self.window1_btn3_img, command=lambda:
self.highlighting_text(),

                     bg="black")

        button1.place(relx=0.5, rely=0.2, anchor="center")

        self.window1_btn3_img.img = self.window1_btn3_img

        label1 = Label(self.window1_btn3_frame, text="Drop file here", font=('arial black',
12, 'bold'), bg="purple",

                       fg="black", width=15, height=1)

        label1.place(relx=0.35, rely=0.35)

        button2 = Button(self.window1_btn3_frame, text="How to use", width=10,
height=2, fg="white",

                         font=('arial black', 12, 'bold'), bg="black", command=lambda:
self.instruction_for_highlight())

        button2.place(rely=0.98, relx=0.98, x=0, y=0, anchor=SE)

        self.view_button3 = Button(self.window1_btn3_frame, text="View", width=10,
height=2,font=('arial black', 12, 'bold'), bg="black",
```

```
fg="white",activebackground="black",activeforeground="white",command=lambda:
self.view_highlight_file())

        self.view_button3.place(rely=0.98, relx=0.01, x=0, y=0, anchor=SW)




    def instruction_for_highlight(self):

        width=(self.window1_width/2)-100

        height=(self.window1_height/2)-150

        font=("Comic Sans MS",20,"bold")

        msg="Upload the image and save the file then application ask for the word to search
if it's their then pop ups the image and save in the desired location as png file "

        msg_window=Toplevel(self.first_window)

        msg_window.geometry("%dx%d+%d+%d" %(500,300,width,height))

        msg_window.resizable(0,0)

        msg_window.focus_set()

        msg_window.iconbitmap("./res/hint.ico")

        msg_window.title("Instruction")

        msg_label=Label(msg_window,text=msg,font=font,wraplength=450)

        msg_label.place(relx=0, rely=0.1)

        msg_window.mainloop()


    def highlighting_text(self):

        try:
```

```
        self.first_window.grab_set()

        self.input = filedialog.askopenfilename(initialdir="c:\\", title="select a file",
filetypes=(

            ("jpg", "*.jpg"), ("jpeg", "*.jpeg"), ("png", "*.png"), ("webp",
"*.webp"),("svg","*.svg"),("raw","*.raw"),("xcf","*.xcf"),("jpx","*.jpx"),


("tiff","*.tiff"),("cr2","*.cr2"),("bmp","*.bmp"),("jxr","*.jxr"),("psd","*.psd"),("ico","*.ic
o"),("heic","*.heic"),("dcm","*.dcm")),parent=self.first_window)

        if self.input:

            saving_file = asksaveasfile(initialfile='Untitled.png',

                                        defaultextension=".png", filetypes=([("png",
"*.png")]),parent=self.first_window)

            if saving_file:

                saving_file.close()

                self.output = saving_file.name

                self.search_word_label = Label(self.window1_btn3_frame, text="Enter the
search word",

                                               font=("times new roman", 15, "bold"),bg="purple")

                self.search_word_label.place(relx=0.1,rely=0.5)

                self.search_word = Entry(self.window1_btn3_frame, width=20,
textvariable=self.var,

                                         font=("times new roman", 15, "bold"))

                self.search_word.place(relx=0.5,rely=0.5)

                self.ok_button = Button(self.window1_btn3_frame,
text="Ok",bg="black",fg="white", width=6,font=("times new roman", 15, "bold"),

                                        command=lambda: self.ok())

                self.ok_button.place(relx=0.4,rely=0.6)
```

```
            else:

                messagebox.askokcancel("warning", "please save the file for accessing
content",parent=self.first_window)

            else:

 messagebox.askokcancel("warning", "please select the image file for
converting",parent=self.first_window)

        except Exception as e:

            messagebox.showerror("error", e)


    def ok(self):

        try:

            self.word = self.search_word.get()

            if self.word == "":

                messagebox.showerror("Error", "please enter the
word",parent=self.first_window)

            else:


                output = highlighting_text(self.input, self.output, self.word)


                if output["output"] != "":


                    #success sound

                    MessageBeep(type=MB_OK)

                    messagebox.showinfo("success", "file is saved..  \nin "+
output['output_file'],parent=self.first_window)

                    self.view_file3=output["output_file"]
```

```
                self.search_word_label.destroy()
                self.search_word.destroy()
                self.ok_button.destroy()
            elif output["error"] != "":
                MessageBeep(type=MB_ICONHAND)
                messagebox.showerror("warning", output["error"],parent=self.first_window)
                self.search_word_label.destroy()
                self.search_word.destroy()
                self.ok_button.destroy()
                self.input=""
        except Exception as e:
            messagebox.showerror("error", e)
    def view_highlight_file(self):
        if self.view_file3!="":
            startfile(self.view_file3)
            self.view_file3 = ""
        else:
            messagebox.askokcancel("warning", "Please  Convert the file before viewing")
```

#Spread Sheet Manipulation

```
class Window2:
    def __init__(self, master, mainwindow):
```

```
#************Design Window2*******************
        self.view_file4=""
        self.view_file5=""
        self.view_file6=""
        self.view_file7=""
        self.var1 = StringVar
        self.var2 = StringVar
        self.var3 = StringVar
        self.second_window = master
        self.second_window.title("Spread Sheet Manipulation")
        self.design_window = mainwindow
        self.window2_height = self.design_window.winfo_screenheight()
        self.window2_width = self.design_window.winfo_screenwidth()
        self.window2_img = Image.open("./res/toplevelspreadsheetmanipulation.jpg")
        self.window2_img = self.window2_img.resize((self.window2_width,
self.window2_height))
        self.window2_img = ImageTk.PhotoImage(self.window2_img)
        label = Label(self.second_window, image=self.window2_img)
        label.pack()
        self.window2_img.img = self.window2_img
        label.config(image=self.window2_img.img)
        self.window2_btn1_frame = Frame(self.second_window)
        self.window2_btn2_frame = Frame(self.second_window)
```

```
        self.window2_btn3_frame = Frame(self.second_window)
        self.window2_btn4_frame = Frame(self.second_window)
        self.xls_to_txt = Button(
            self.second_window, text="Excel To Text", width=30, height=3,
            command=lambda: self.upload_xls_to_text_file(), font=('arial', 12, 'bold'),
            bg='#FF005E', fg="white",
            borderwidth=8)
        self.csv_to_text = Button(
            self.second_window, text="Csv To Text", width=30, height=3, font=('arial', 12,
'bold'),
            bg='#FF005E', fg="white", command=lambda: self.upload_csv_to_text(),
            borderwidth=8)
        self.xls_to_csv = Button(
            self.second_window, text="Excel To Csv ", width=30, height=3, font=('arial', 12,
'bold'),
            bg='#FF005E', fg="white", command=lambda: self.upload_xls_to_csv(),
            borderwidth=8)
        self.search_columns = Button(
            self.second_window, text="Search Columns", width=30, height=3, font=('arial',
12, 'bold'),
            bg='#FF005E', fg="white", command=lambda:
self.upload_search_columns_in_xls(),
            borderwidth=8)
        self.exit_btn2 = Button(self.second_window, text="Exit", width=30, height=3,
```

```
command=self.exit_window, font=('arial', 12, 'bold'), bg='#0055FF', fg="white",
                        borderwidth=8)
    self.xls_to_txt.place(relx=0.02,rely=0.2)
    self.csv_to_text.place(relx=0.02,rely=0.35)
    self.xls_to_csv.place(relx=0.02,rely=0.5)
    self.search_columns.place(relx=0.02,rely=0.65)
    self.exit_btn2.place(relx=0.02,rely=0.8)
    self.second_window.protocol("WM_DELETE_WINDOW", self.on_exit)


  def exit_window(self):
    self.design_window.deiconify()
    self.second_window.destroy()
  def on_exit(self):
    self.second_window.destroy()
    self.design_window.deiconify()


  # first button code xls to text
  #*********************Button1************************
  def upload_xls_to_text_file(self):
    self.window2_btn2_frame.destroy()
    self.window2_btn3_frame.destroy()
    self.window2_btn4_frame.destroy()
```

```
        self.window2_btn1_frame = Frame(self.second_window, width=400, height=400,
bg="purple")


        self.window2_btn1_frame.place(relx=0.4, rely=0.2)

        self.window2_btn1_frame.config(relief=SUNKEN)

        # Image for button

        self.window2_btn1_img = Image.open("./res/file.png")

        self.window2_btn1_img = self.window2_btn1_img.resize((150, 120))

        self.window2_btn1_img = ImageTk.PhotoImage(self.window2_btn1_img)

        # upload button

        button1 = Button(self.window2_btn1_frame, text="Upload", width=150,
height=120, fg="white", font=(

            'arial', 10, 'bold'), image=self.window2_btn1_img,

                    command=lambda: self.xls_to_text_file(),

                    bg="black")

        button1.place(relx=0.5, rely=0.2, anchor="center")

        self.window2_btn1_img.img = self.window2_btn1_img

        button1.config(image=self.window2_btn1_img.img)

        # label for upload

        label1 = Label(self.window2_btn1_frame, text="Drop file here", font=('arial black',
12, 'bold'),bg="purple",fg="black", width=15, height=1)

        label1.place(relx=0.3, rely=0.4)

        # instruction button

        button2 = Button(self.window2_btn1_frame, text="How to use", width=10,
height=2, fg="white",font=('arial black', 12, 'bold'), bg="black", command=lambda:
self.instruction_xls_to_text())
```

```
        button2.place(rely=0.98, relx=0.98, x=0, y=0, anchor=SE)
        view_button4 = Button(self.window2_btn1_frame, text="view", width=10,
height=2,
                              font=('arial black', 12, 'bold'), bg="black", fg="white",
                              activebackground="black", activeforeground="white",
                              command=lambda: self.view_xls_to_text_file())
        view_button4.place(relx=0.01, rely=0.82)


    def instruction_xls_to_text(self):
        width = (self.window2_width / 2) - 100
        height = (self.window2_height / 2) - 150
        font = ("Comic Sans MS", 20, "bold")
        msg = "upload the xls or xlsx file application will convert the uploaded file into text
file"
        msg_window = Toplevel(self.second_window)
        msg_window.geometry("%dx%d+%d+%d" % (500, 300, width, height))
        msg_window.resizable(0, 0)
        msg_window.focus_set()
        msg_window.iconbitmap("./res/hint.ico")
        msg_window.title("Instruction")
        msg_label = Label(msg_window, text=msg, font=font, wraplength=450)
        msg_label.place(relx=0, rely=0.1)
msg_window.mainloop()
```

```
    def xls_to_text_file(self):
        try:
            self.second_window.grab_set()
            xls_file = filedialog.askopenfilename(initialdir="c:\\", title="select a file",
filetypes=(
                ("xls file", "*.xls"), ("xlsx file", "*.xlsx")),parent=self.second_window)
            if xls_file:
                textfile = asksaveasfile(initialfile='Untitled.txt',
                                         defaultextension=".txt", filetypes=[("txt file",
"*.txt")],parent=self.second_window)
                if textfile:
                    textfile.close()
                    output_file = textfile.name


                    output = xls_to_text(xls_file, output_file)
                    if output["output"] != "":
                        MessageBeep(type=MB_OK)
                        messagebox.showinfo("success", "file is saved..  \nin "+
output['output_file'],parent=self.second_window)
                        self.view_file4 = output["output_file"]
                    elif output["error"] != "":
                        MessageBeep(type=MB_ICONHAND)

messagebox.showerror("warning",output["error"],parent=self.second_window)
                else:
```

```
                messagebox.showerror("warning", "Please save the
file",parent=self.second_window)
            else:
                messagebox.showerror("error!", "please select the xls
file",parent=self.second_window)
        except Exception as e:
            messagebox.showerror("error", e)


    def view_xls_to_text_file(self):
        if self.view_file4 != "":
            startfile(self.view_file4)
            self.view_file4 = ""
        else:
            messagebox.askokcancel("warning", "Please  Convert the file before
viewing",parent=self.second_window)
        # 2nd button csv to text
        #*******************Button2*************
    def upload_csv_to_text(self):
        self.window2_btn1_frame.destroy()
        self.window2_btn3_frame.destroy()
        self.window2_btn4_frame.destroy()
        self.window2_btn2_frame = Frame(self.second_window, width=400, height=400,
bg="purple")


       self.window2_btn2_frame.place(relx=0.4, rely=0.2)
        self.window2_btn2_frame.config(relief=SUNKEN)
```

```
        self.window2_btn2_img = Image.open("./res/file.png")

        self.window2_btn2_img = self.window2_btn2_img.resize((150, 120))

        self.window2_btn2_img = ImageTk.PhotoImage(self.window2_btn2_img)

        button1 = Button(self.window2_btn2_frame, text="Upload", width=150,
height=120, fg="white", font=(

            'arial', 10, 'bold'), image=self.window2_btn2_img,

                     command=lambda: self.csv_text(),

                     bg="black")

        button1.place(relx=0.5, rely=0.2, anchor="center")

        self.window2_btn2_img.img = self.window2_btn2_img

        button1.config(image=self.window2_btn2_img.img)

        label1 = Label(self.window2_btn2_frame, text="Drop file here", font=('arial black',
12, 'bold'), bg="purple",

                       fg="black", width=15, height=1)

        label1.place(relx=0.3, rely=0.4)

        button2 = Button(self.window2_btn2_frame, text="How to use", width=10,
height=2, fg="white",

                         font=('arial black', 12, 'bold'), bg="black", command=lambda:
self.instruction_csv_to_text())

        button2.place(rely=0.98, relx=0.98, x=0, y=0, anchor=SE)

        view_button4 = Button(self.window2_btn2_frame, text="view", width=10,
height=2,

                              font=('arial black', 12, 'bold'), bg="black", fg="white",

                              activebackground="black", activeforeground="white",

                              command=lambda: self.view_csv_to_text())

        view_button4.place(relx=0.01, rely=0.82)
```

```
    def instruction_csv_to_text(self):
        width = (self.window2_width / 2) - 100
        height = (self.window2_height / 2) - 150
        font = ("Comic Sans MS", 20, "bold")
        msg = "Upload the csv file then application convert the uploaded file into text file"
        msg_window = Toplevel(self.second_window)
        msg_window.geometry("%dx%d+%d+%d" % (500, 300, width, height))
        msg_window.resizable(0, 0)
        msg_window.focus_set()
        msg_window.iconbitmap("./res/hint.ico")
        msg_window.title("Instruction")
        msg_label = Label(msg_window, text=msg, font=font, wraplength=450)
        msg_label.place(relx=0, rely=0.1)
        msg_window.mainloop()
    def csv_text(self):
            try:
                self.second_window.grab_set()
                csv_file = filedialog.askopenfilename(initialdir="c:\\", title="select a file",
filetypes=(
                    ("csv file", "*.csv"), ("csv file", "*.csv")),parent=self.second_window)
                if csv_file:
                    text_file = asksaveasfile(initialfile='Untitled.txt',
defaultextension=".txt", filetypes=[("text file", "*.txt")],parent=self.second_window)
                    if text_file:
```

```
                text_file.close()
                output_file = text_file.name
                output = csv_to_text(csv_file, output_file)
                if output["output"] != "":
                    MessageBeep(type=MB_OK)
                    messagebox.showinfo("success", "file is saved..  \nin "+
output['output_file'],parent=self.second_window)
                    self.view_file5 = output["output_file"]
                elif output["error"] != "":
                    MessageBeep(type=MB_ICONHAND)
            messagebox.showerror("warningoutput["error"],parent=self.second_window)
              else:
                messagebox.showerror("error!", "please save the
file",parent=self.second_window)
          else:
            messagebox.showerror("error!", "please select the csv
file",parent=self.second_window)
        except Exception as e:
          messagebox.showerror("error", e)
  def view_csv_to_text(self):
    if self.view_file5 != "":
      startfile(self.view_file5)
      self.view_file5= ""
    else:messagebox.askokcancel("warning", "Please  Convert the file before
viewing",parent=self.second_window)
```

5 . TESTING

5.1 Introduction

Testing is the major quality control measures and during the software development it is used to detect errors that could have occurred during any of the phase like requirement analysis, design, coding. The goal of the testing is to uncover errors in the program.

5.2 Levels of Testing

Testing is done in different levels which includes the following.

- Unit Testing
- Integration Testing
- System testing
- Acceptance testing

  - **Unit Testing**

    In Unit testing each module gets tested during the coding phase itself. The purpose is to exercise the different parts of the module code to detect the coding errors.

  - **Integration Testing**

    After new testing the modules are gradually integrated into sub systems. It is performed to detect design errors by focusing on testing the interconnection between modules.

  - **System Testing**

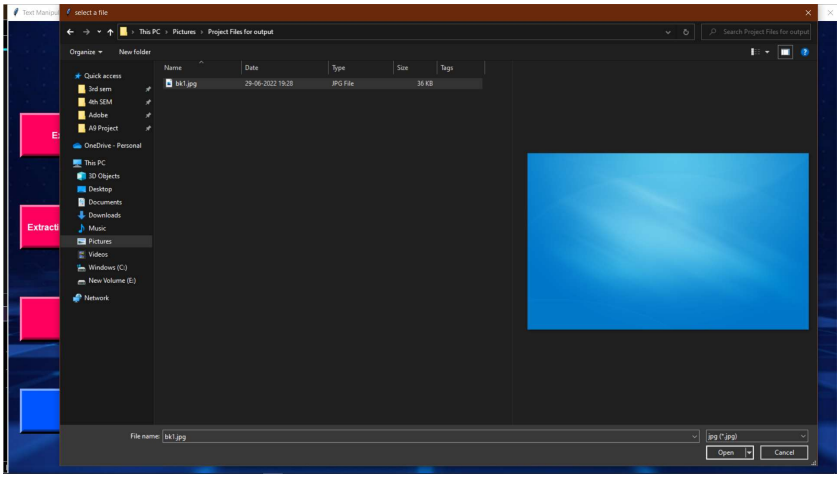
    System is tested against the system requirement if all the requirements are met and if the system performs as specified by the requirement.

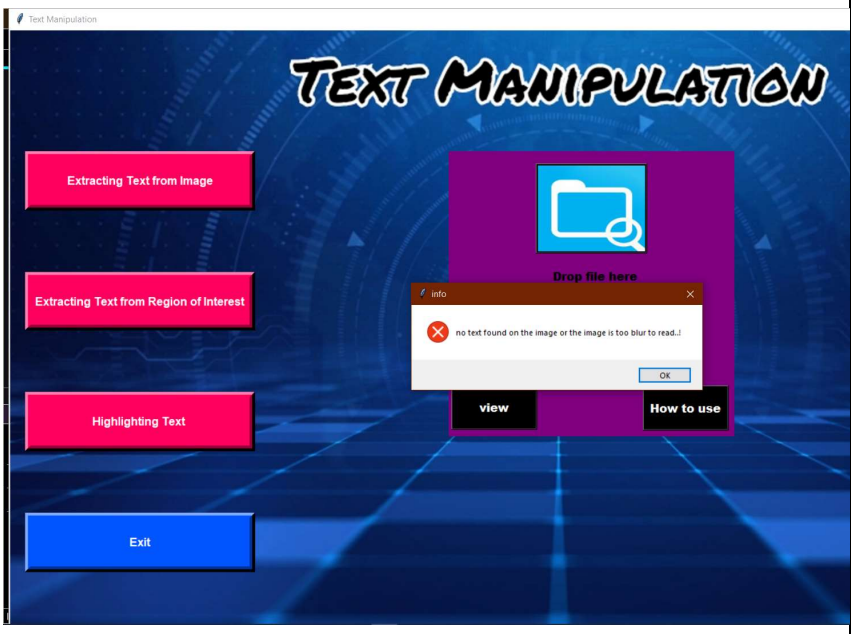
  - **Acceptance Testing**

    It is performed to demonstrate to the client on real life data of the client, the operation of the system.

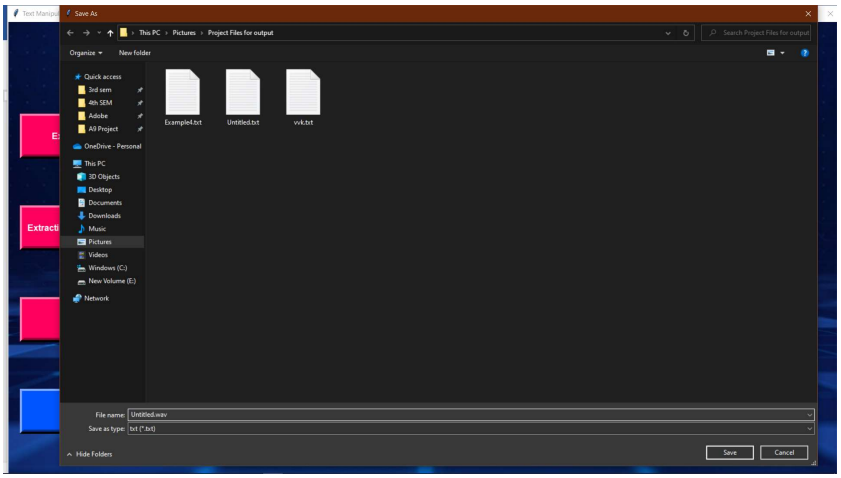
5.3 Test Case

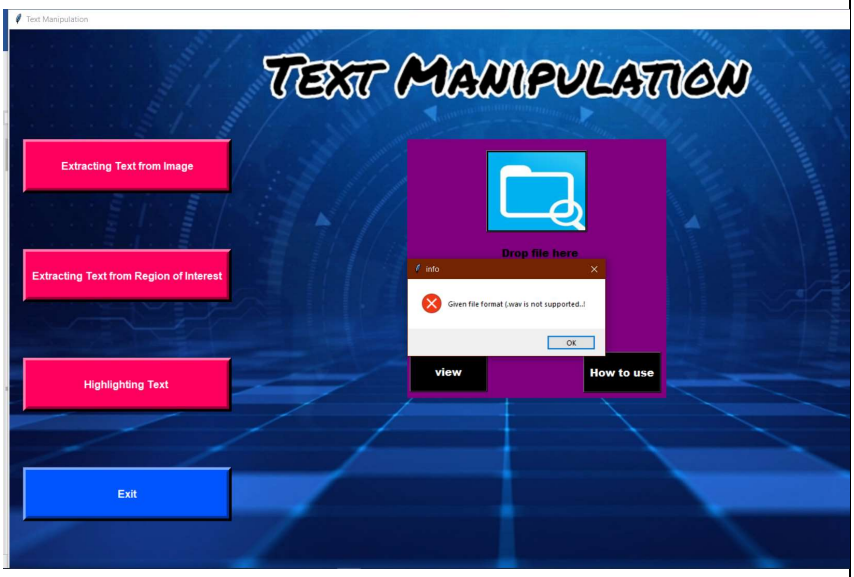
It is the input that tests the genuineness of the program and successful execution of the test case revels. that there are no errors in the program that are under testing. it is a set of conditions or variables under which tester will determine whether an application or software is working currently

Test case-ID	01
Test case Title	Extracting Text from Image
Purpose of testing	Testing for image contain text or not
Test Data	Image
Steps	Step 1: if Image contains text then Step 2: Save the text file and display location of file Step 3:ELSE Display Error message Invalid input: 

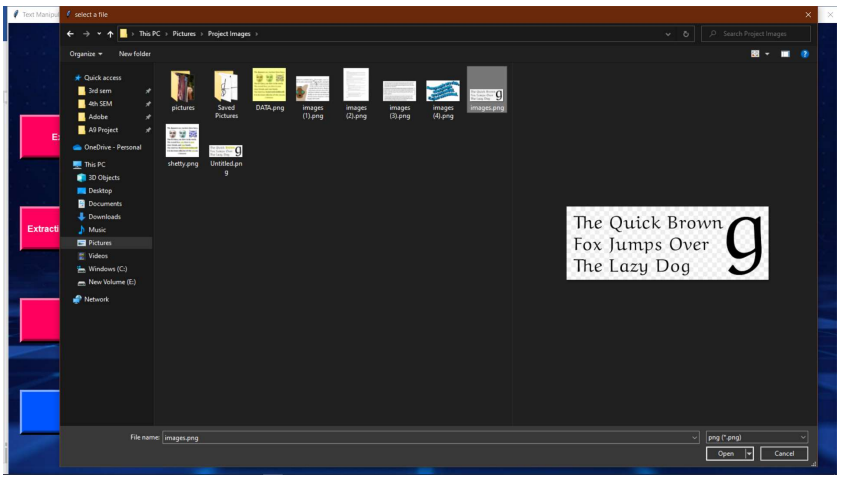


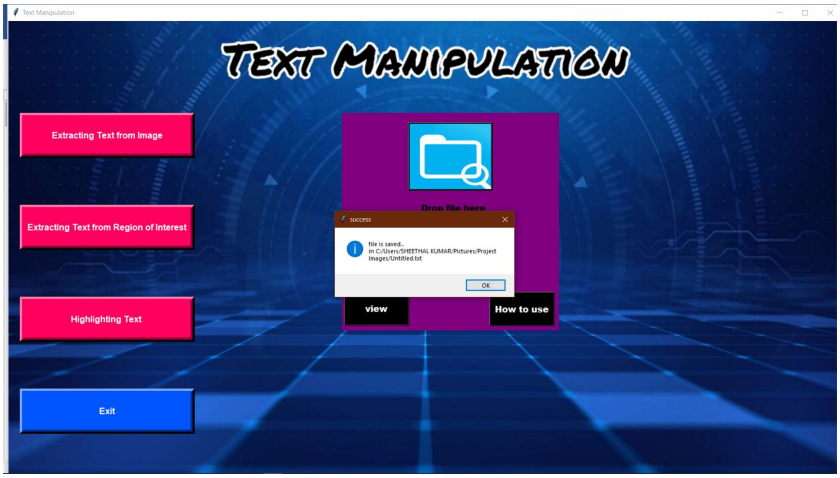
Invalid Extension for saving File:



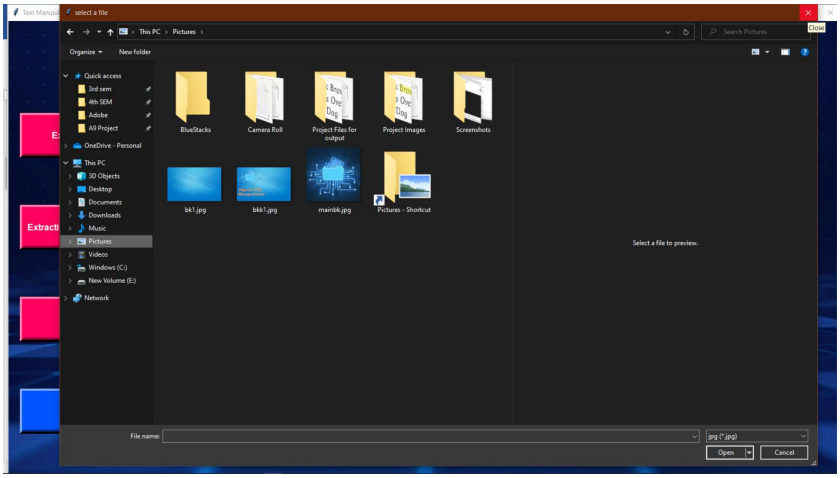

Text Manipulation
TEXT MANIPULATION
Extracting Text from Image
Extracting Text from Region of Interest
Highlighting Text
Exit
Drop file here
info
Given file format (.wav is not supported..!
OK
view
How to use

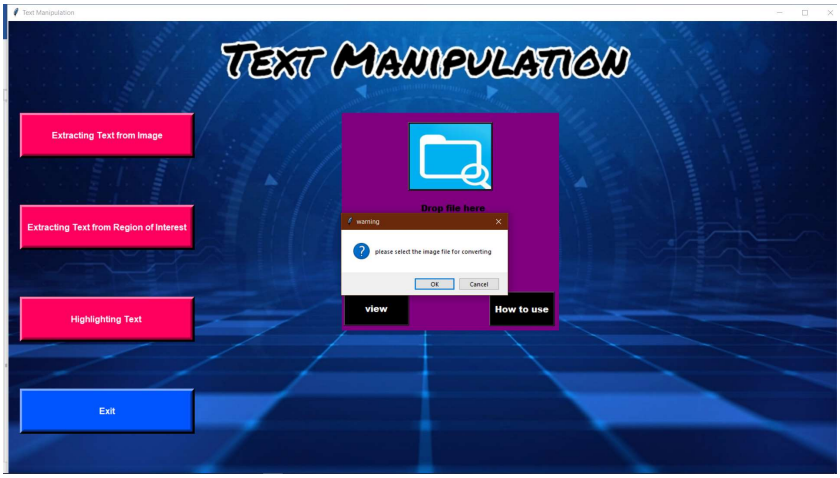
Valid input:


select a file
This PC
Pictures
Project Images
Organize
New folder
Quick access
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
New Volume (E:)
Network
pictures
Saved Pictures
DATA.png
images (1).png
images (2).png
images (3).png
images (4).png
images.png
shetty.png
Untitled.png
The Quick Brown
Fox Jumps Over
The Lazy Dog
g
File name: images.png
png (*.png)
Open
Cancel

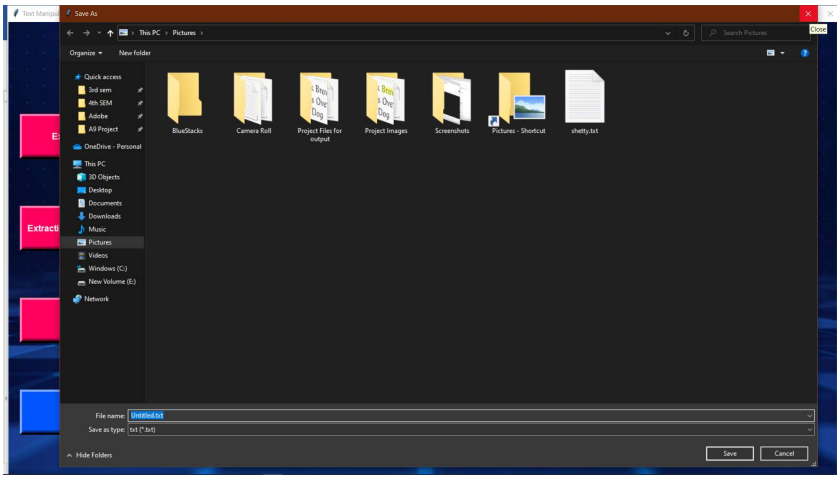


Invalid Selection:

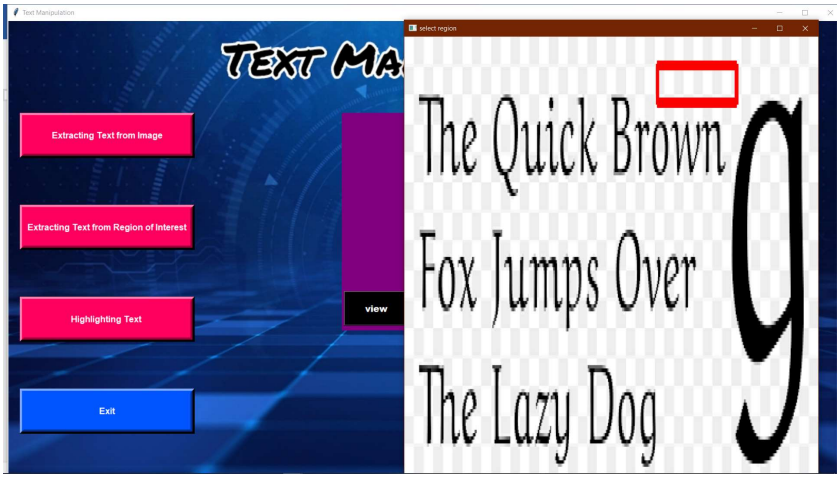


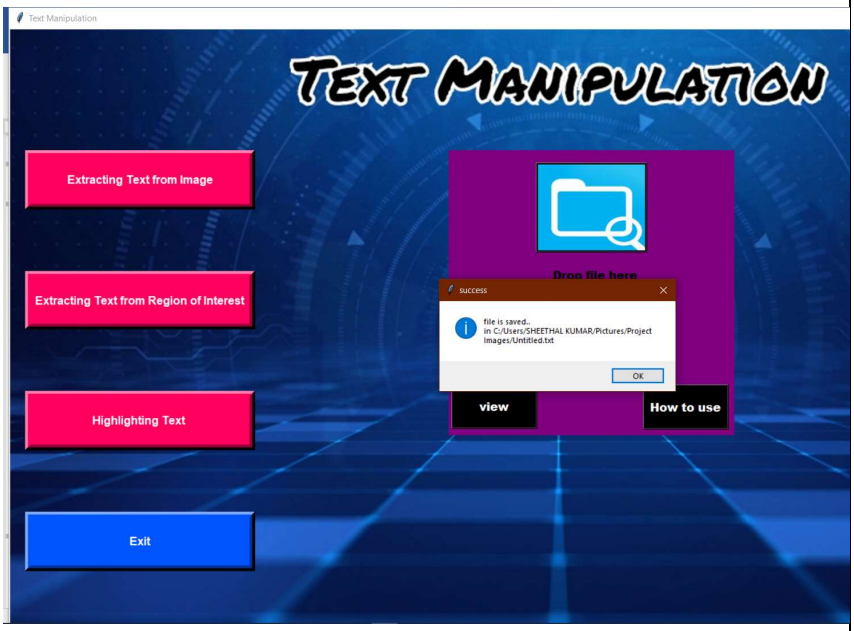


Invalid save:

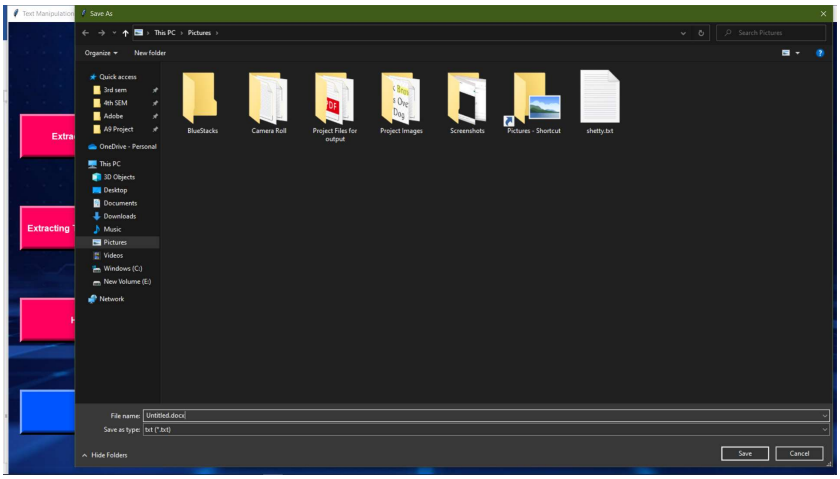


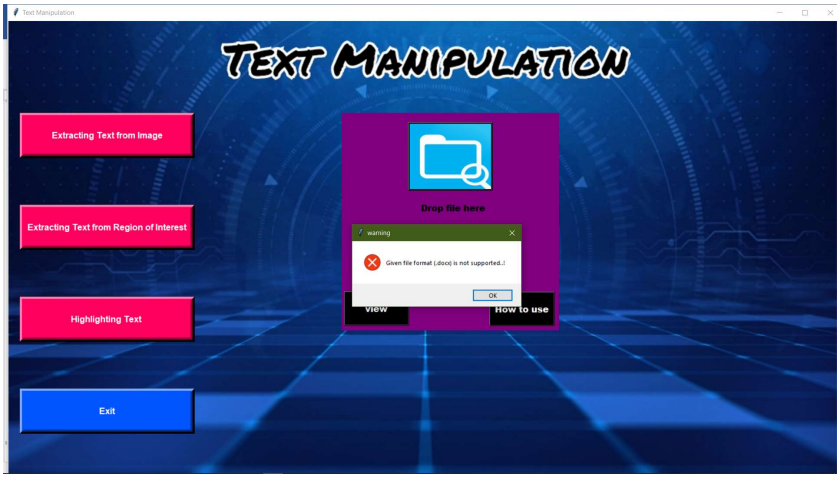
	 Text Manipulation TEXT MANIPULATION Extracting Text from Image Extracting Text from Region of Interest Highlighting Text Exit Drop file here warning please save the file for accessing content. OK Cancel view How to use

Test case-ID	02
Test case Title	Extracting Text from Region of Interest
Purpose of testing	Testing for region contains text or not
Test Data	Image
Steps	Step 1: if Image contains text then Step 2: Select the region, save the text file and display the location of file Step 3: ELSE Display Error message Invalid input:  

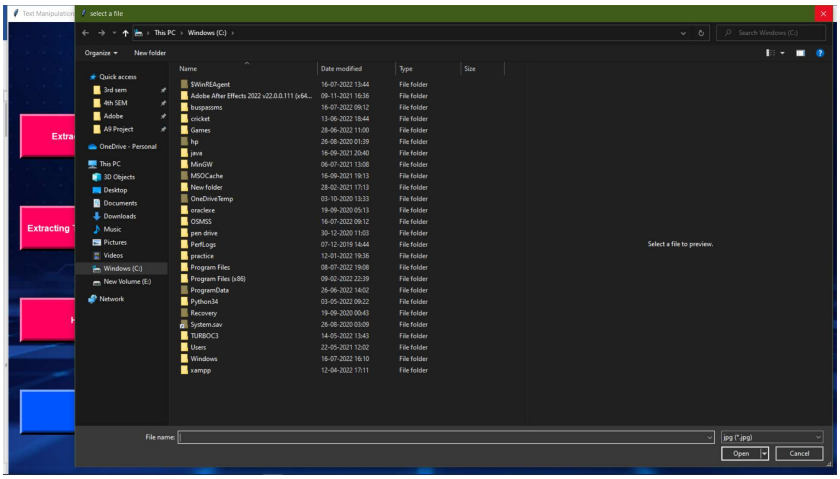
	Valid input:  

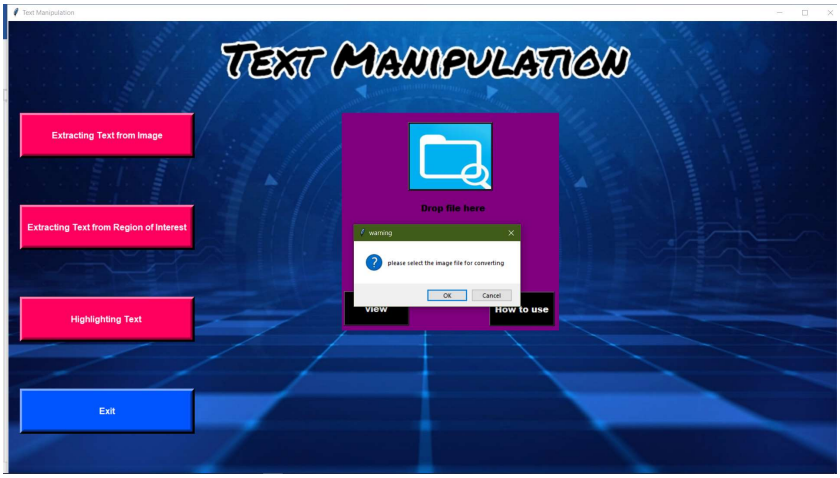
Invalid Extension for save file:



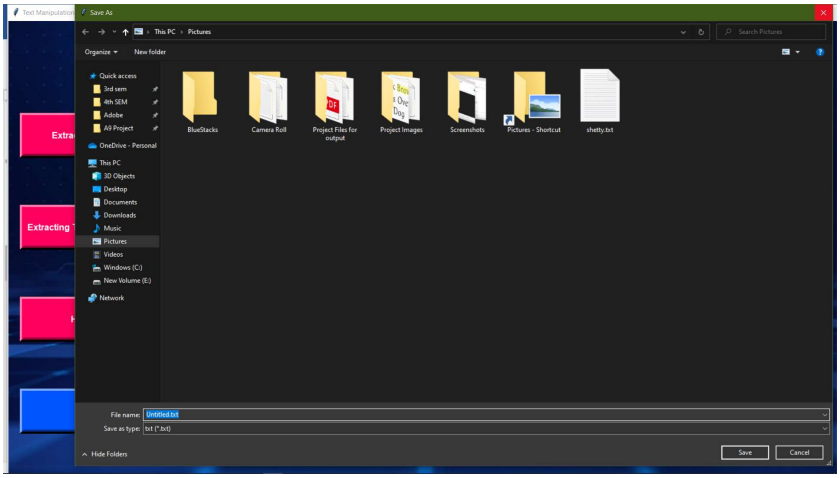


Invalid selection:

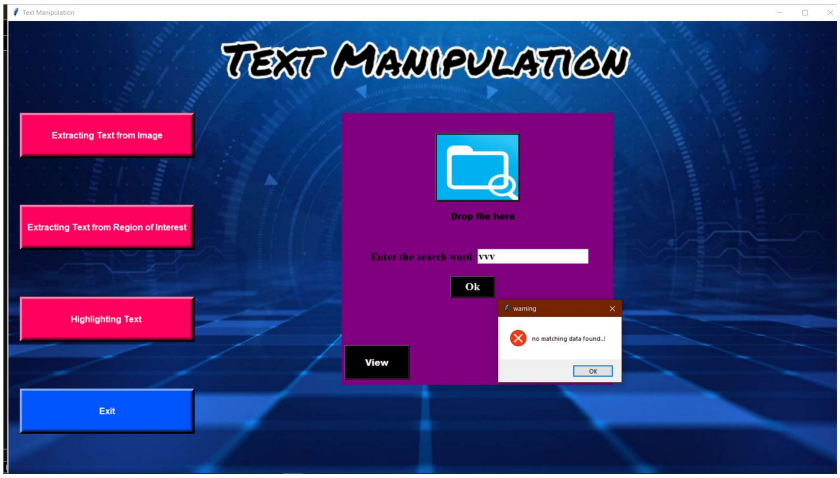




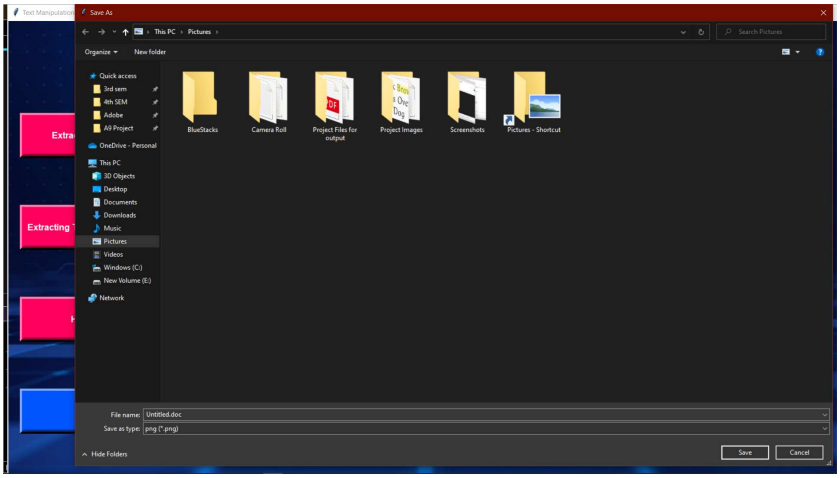
Invalid saving:



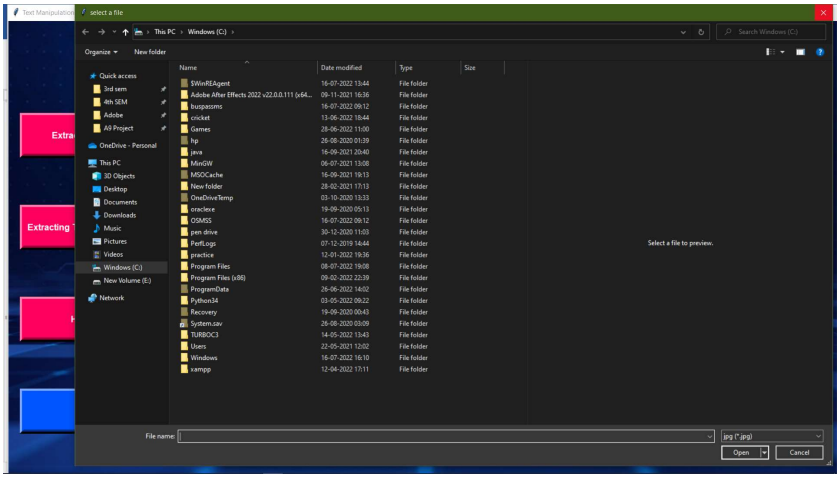
	 Text Manipulation TEXT MANIPULATION Extracting Text from Image Extracting Text from Region of Interest Highlighting Text Exit Drop file here warning please save the file for accessing content OK Cancel View How to use

Test case-ID	03
Test case Title	Highlighting Text
Purpose of testing	Testing for image contain the searched word or not
Test Data	Image
Steps	Step 1: if Image contains text then Step 2: Search for the word, save the file and display location of file Step 3:ELSE Display Error message Invalid input:  Valid input: 

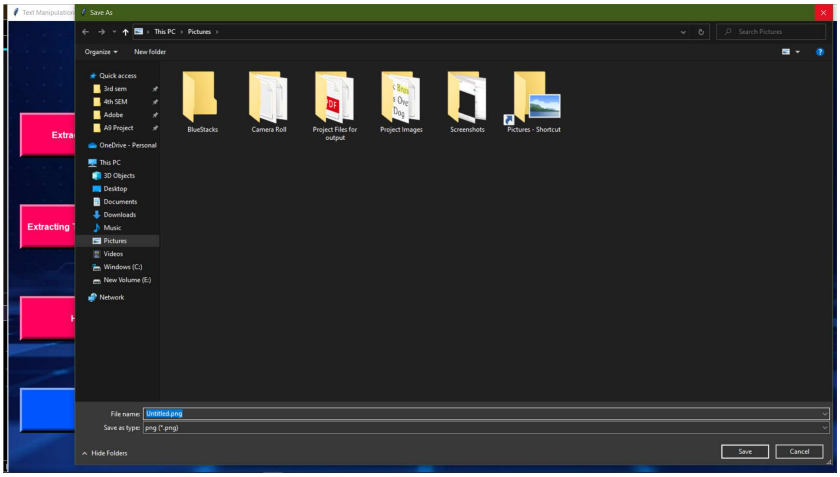
Invalid extension for save file:

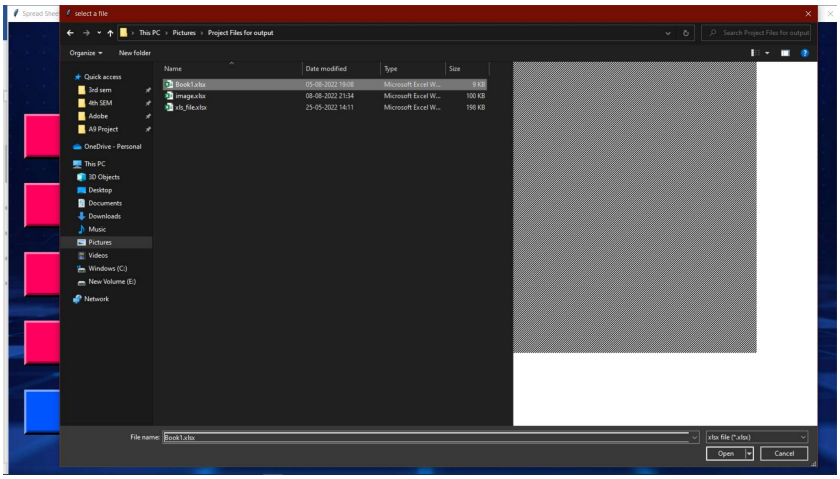


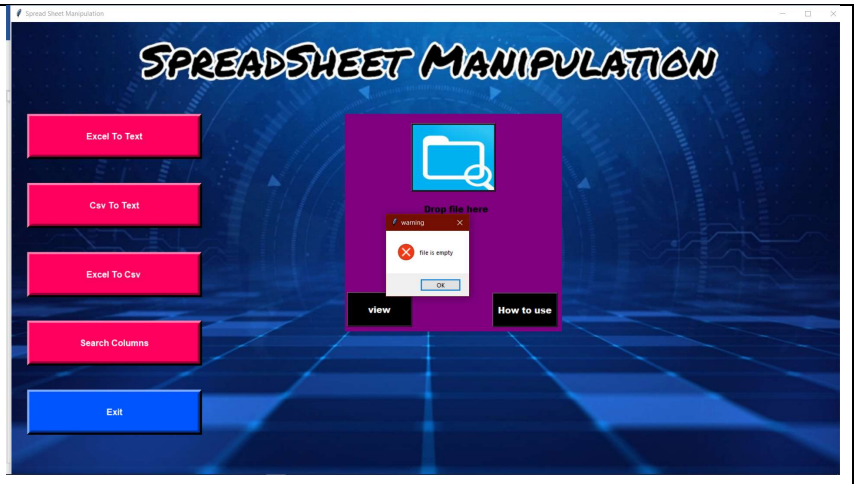
Invalid Selection:



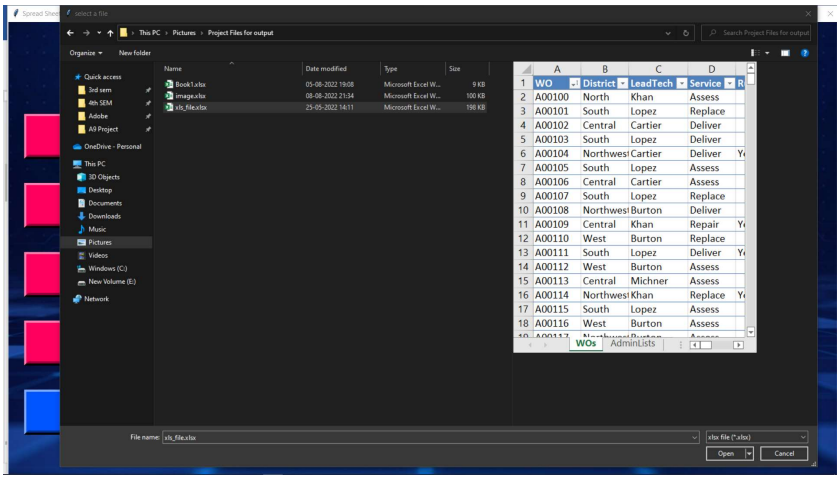
Invalid Saving:

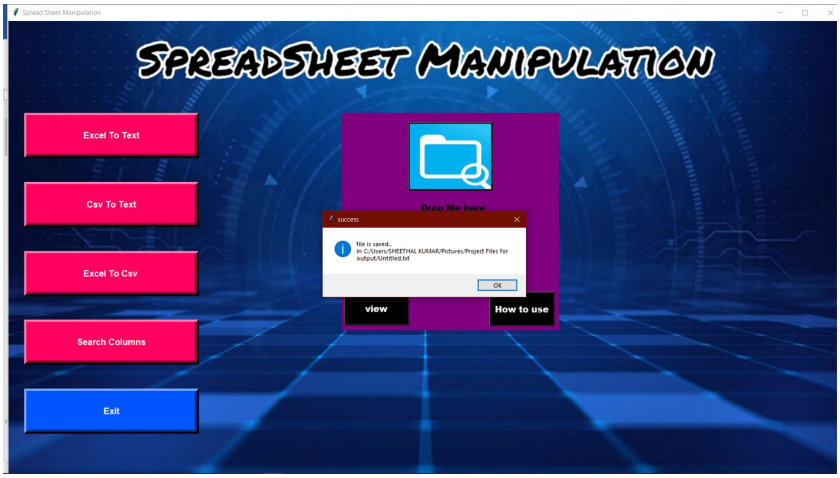


Test case-ID	04
Test case Title	Excel to Text
Purpose of testing	Testing for Excel file contains data or not
Test Data	Excel file
Steps	Step 1: if Excel file contains data then Step 2: Convert to text file and display location of file Step 3:ELSE Display Error message Invalid input: 

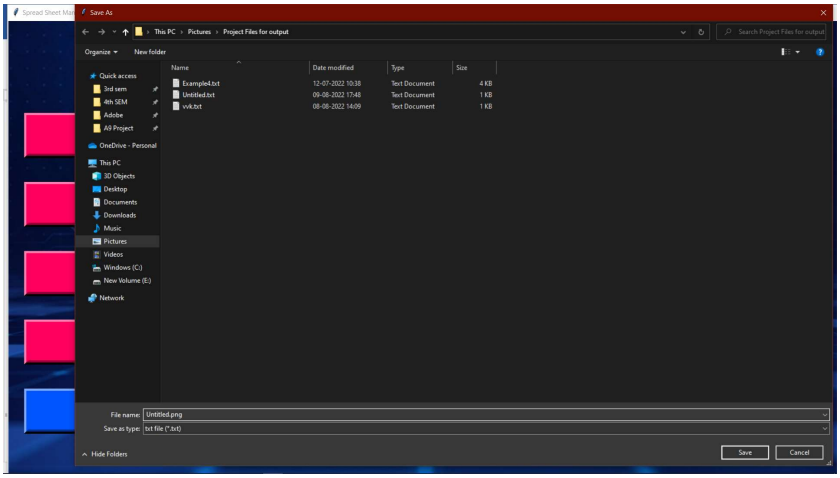

SPREADSHEET MANIPULATION
Excel To Text
Csv To Text
Excel To Csv
Search Columns
Exit
Drop file here
warning
file is empty
OK
view
How to use

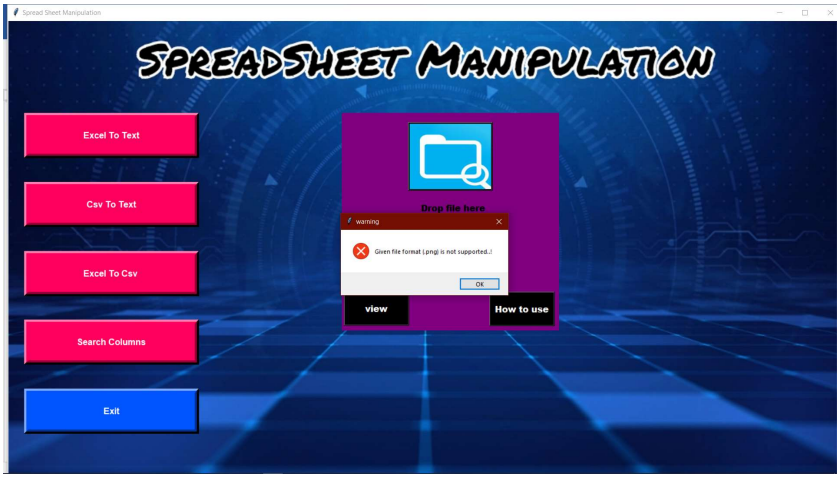
Valid input:


select a file
This PC › Pictures › Project Files for output
Organize
New folder
Name
Date modified
Type
Size
Book1.xlsx
image.xlsx
xls_file.xlsx
WO
District
LeadTech
Service
A00100 North Khan Assess
A00101 South Lopez Replace
A00102 Central Cartier Deliver
A00103 South Lopez Deliver
A00104 Northwest Cartier Deliver
A00105 South Lopez Assess
A00106 Central Cartier Assess
A00107 South Lopez Replace
A00108 Northwest Burton Deliver
A00109 Central Khan Repair
A00110 West Burton Replace
A00111 South Lopez Deliver
A00112 West Burton Assess
A00113 Central Michner Assess
A00114 Northwest Khan Replace
A00115 South Lopez Assess
A00116 West Burton Assess
WOs
AdminLists
File name:
xls_file.xlsx
xlsx file (*.xlsx)
Open
Cancel

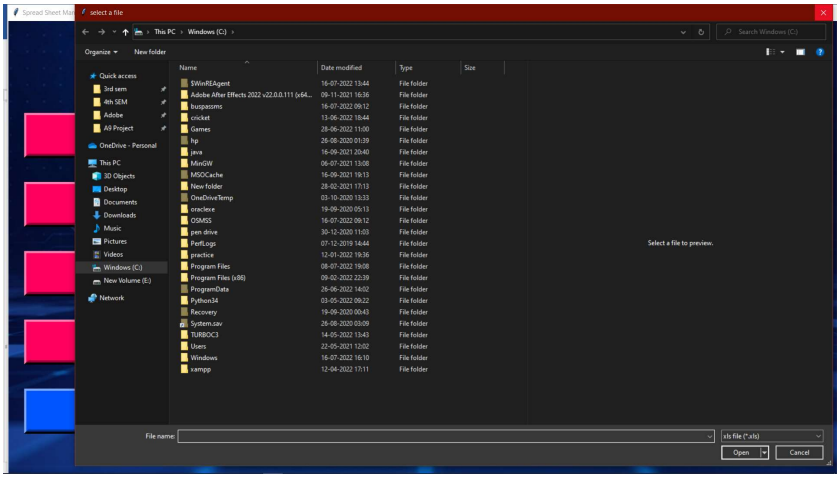

SPREADSHEET MANIPULATION
Excel To Text
Csv To Text
Excel To Csv
Search Columns
Exit
Drop file here
success
file is saved..
in C:/Users/SHEETHAL KUMAR/Pictures/Project Files for output/Untitled.txt
OK
view
How to use

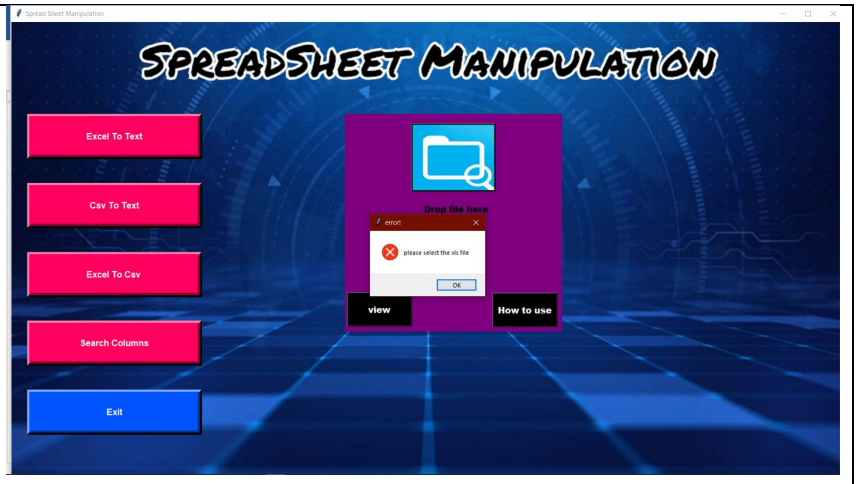
Invalid Extension:



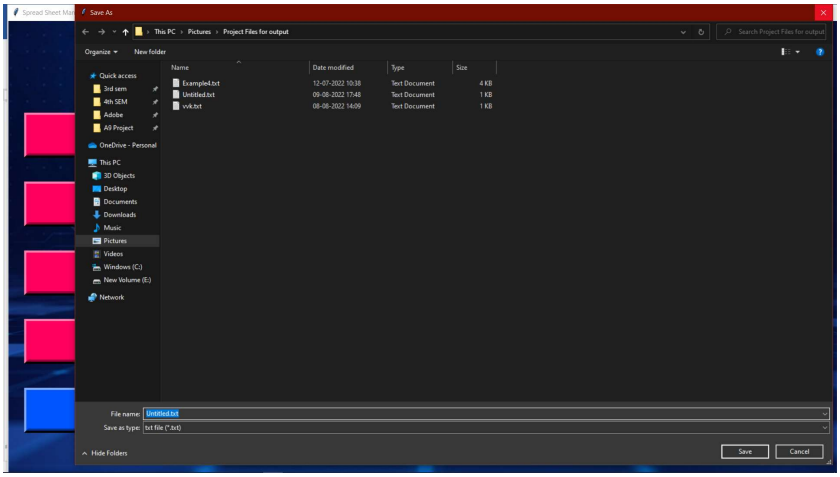


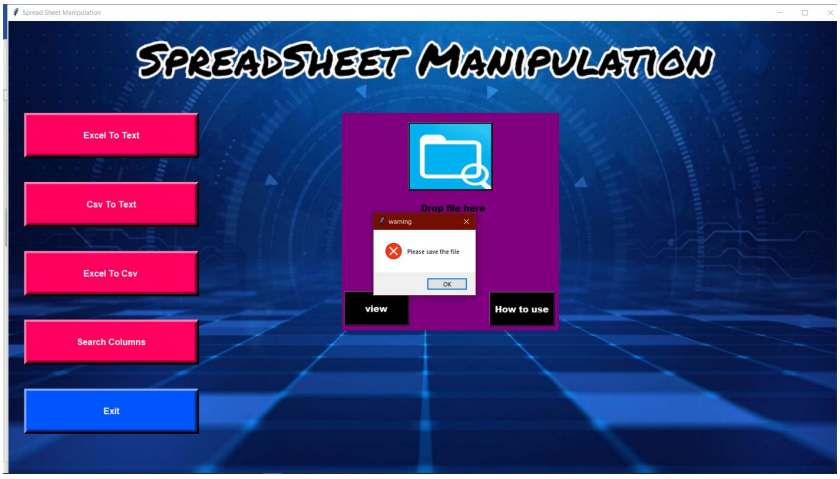
Invalid Selection:

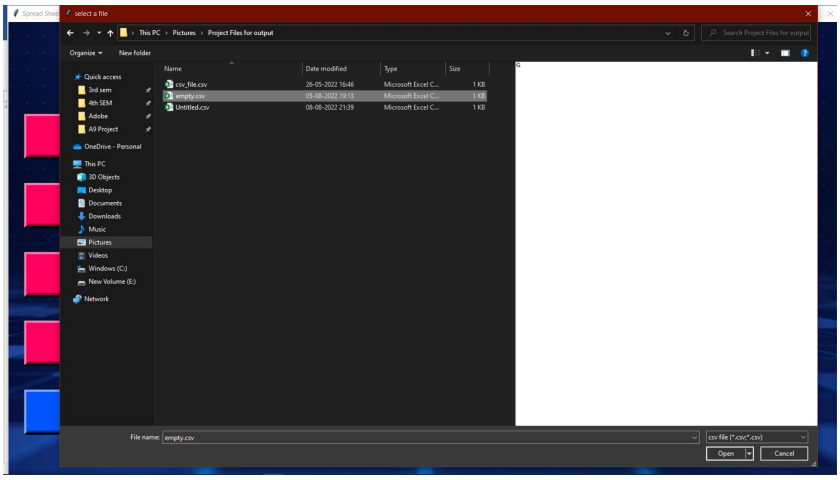


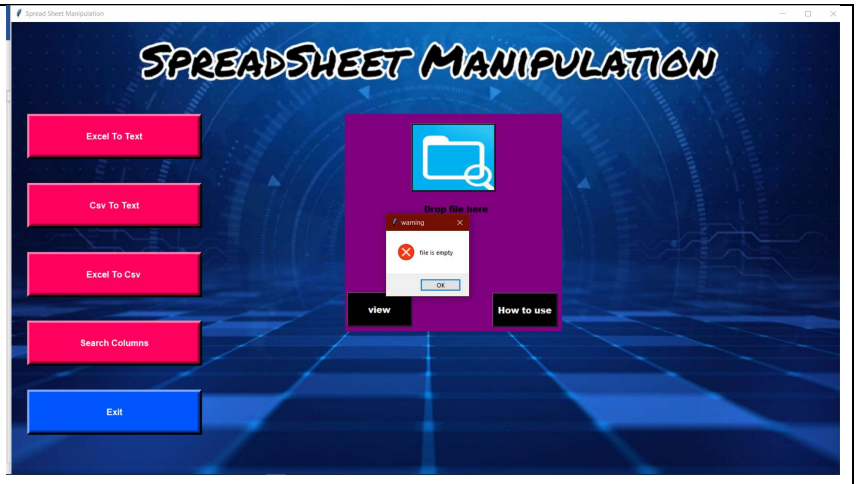


Invalid Saving File:

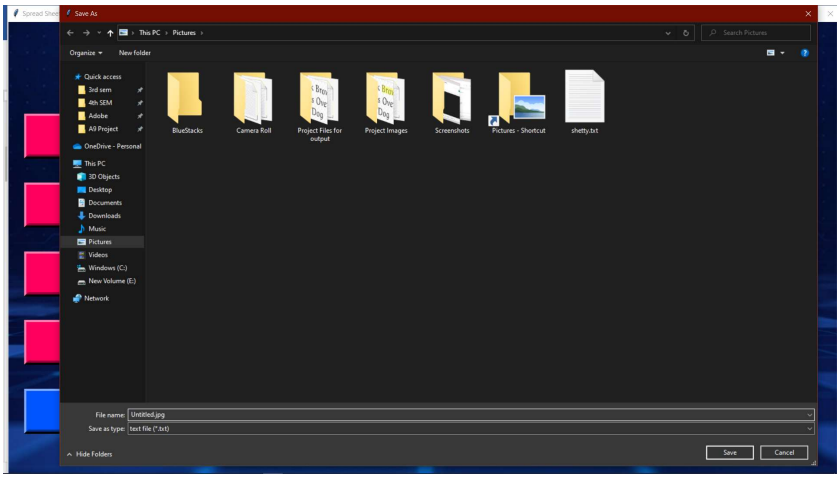


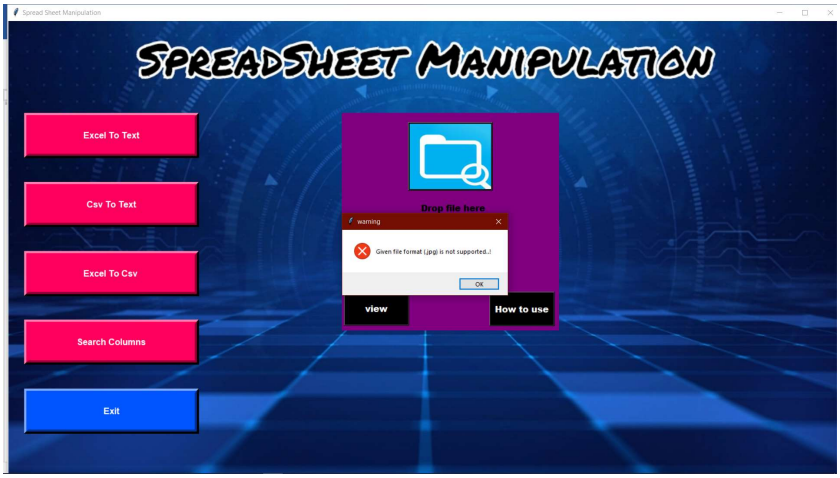


Test case-ID	05
Test case Title	CSV to Text
Purpose of testing	Testing for csv file contains data or not
Test Data	Image
Steps	Step 1: if csv file contains data then Step 2: convert to text file and display location of file Step 3:ELSE Display Error message Invalid input: 

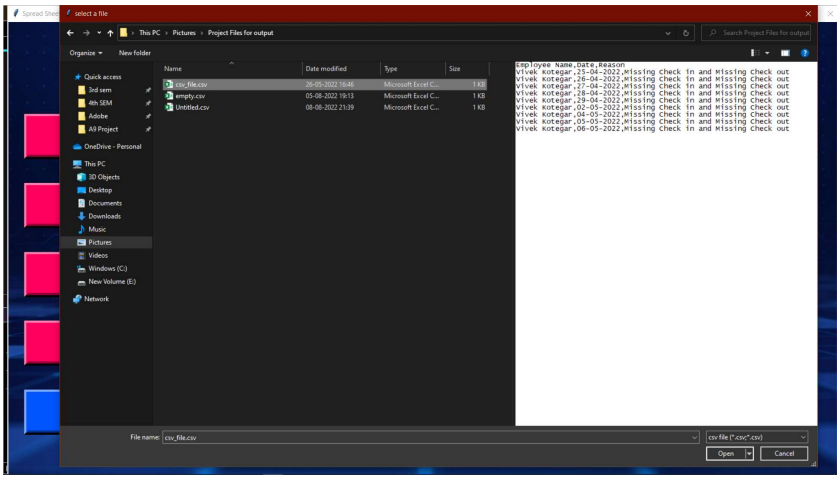


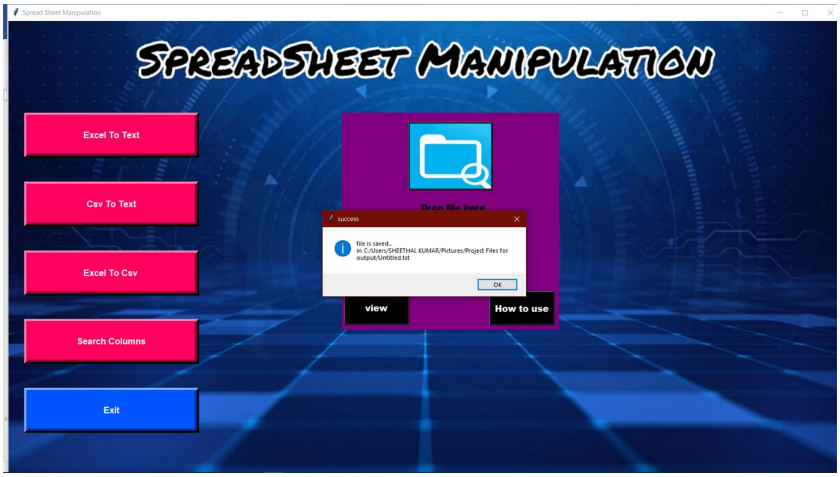
Invalid Extension for saving File:





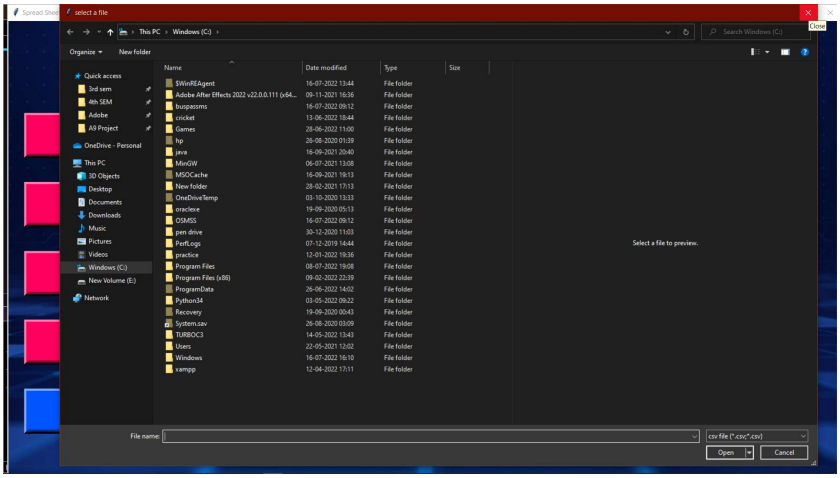
Valid input:

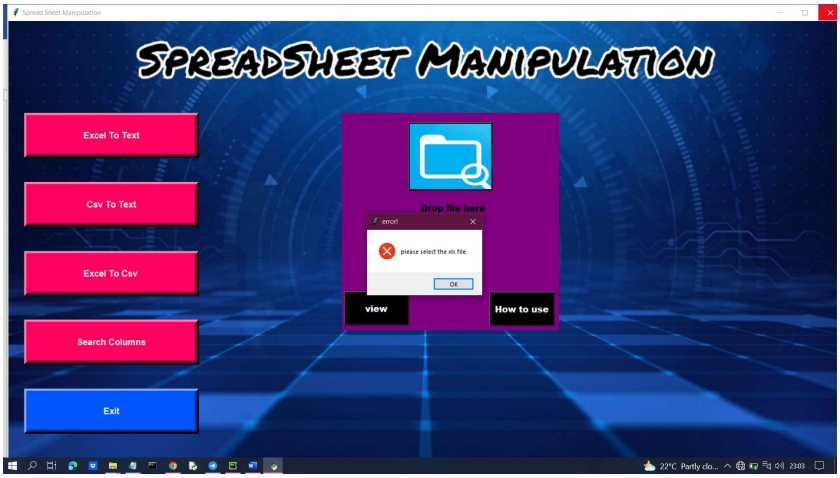




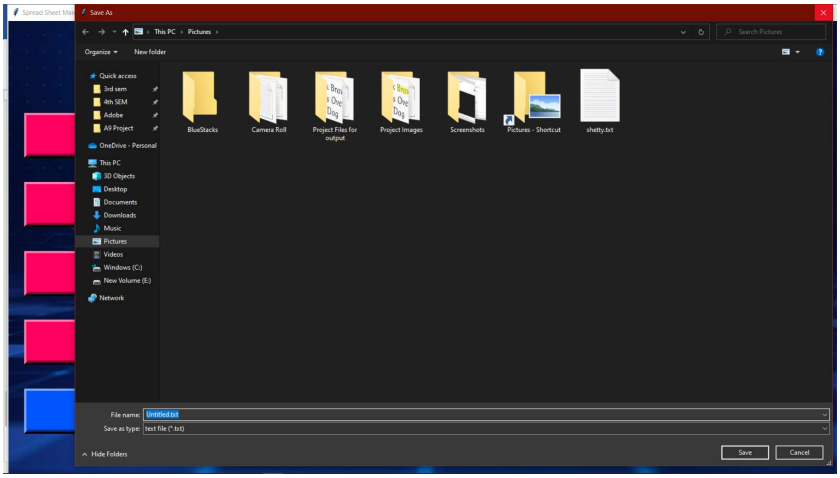
Invalid Selection:

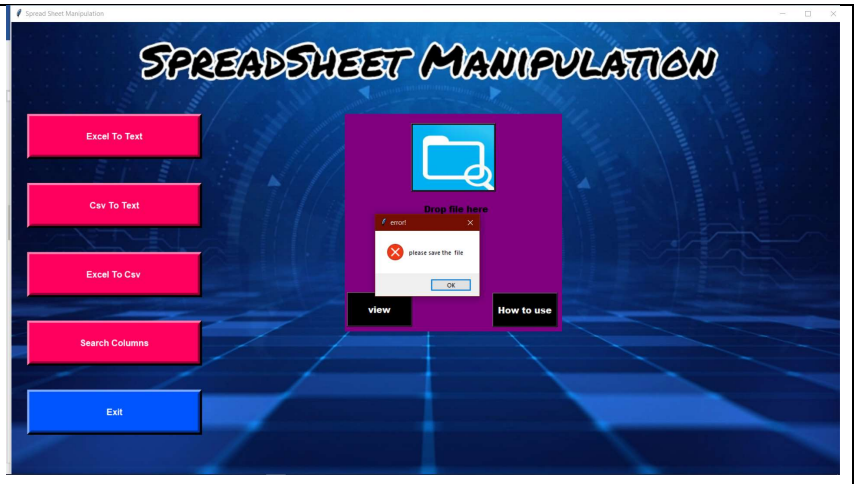
`

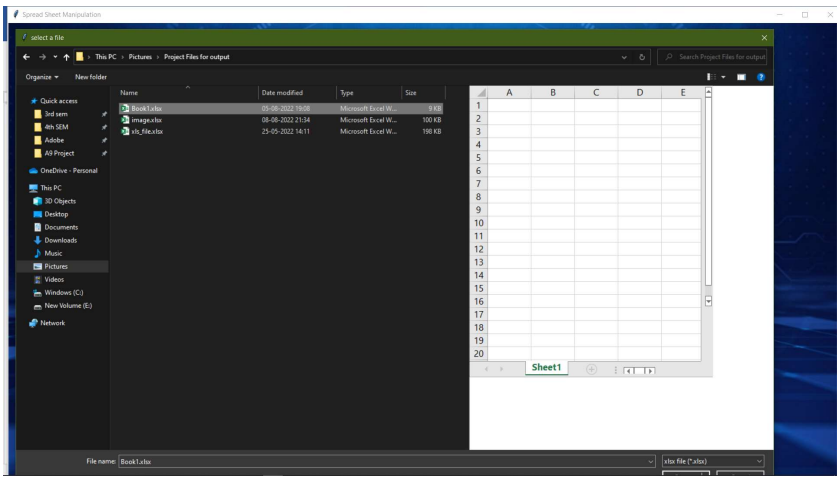


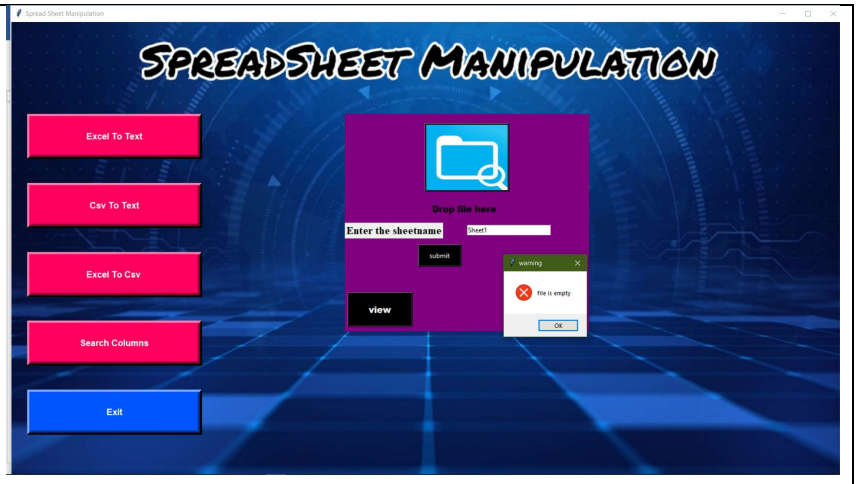


Invalid saving:

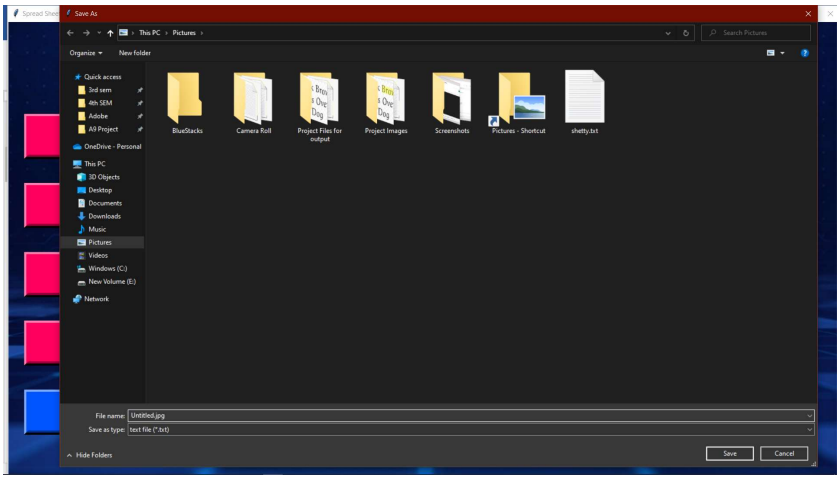


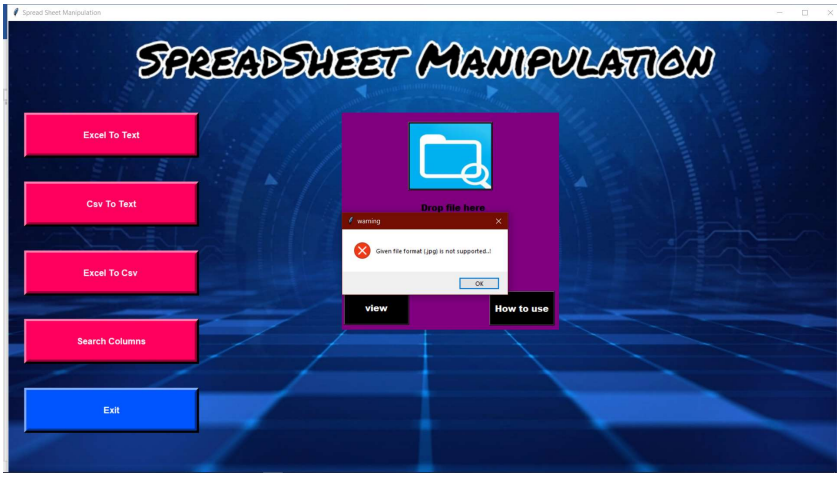
	 Spread Sheet Manipulation SPREADSHEET MANIPULATION Excel To Text Csv To Text Excel To Csv Search Columns Exit Drop file here error please save the file OK view How to use

Test case-ID	06
Test case Title	Excel to csv
Purpose of testing	Testing for Excel file contains data or not using sheetname
Test Data	Excel File
Steps	Step 1: if Excel file contains data then Step 2: convert to csv file using sheetname and display location of file Step 3:ELSE Display Error message Invalid input: 

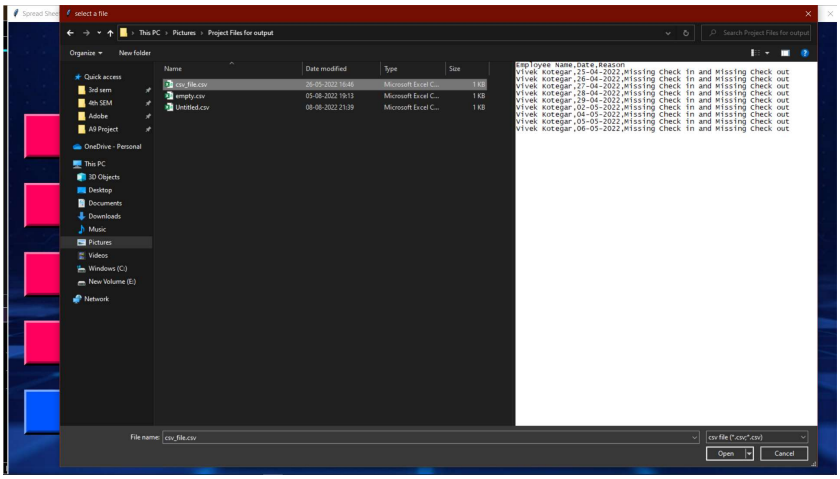


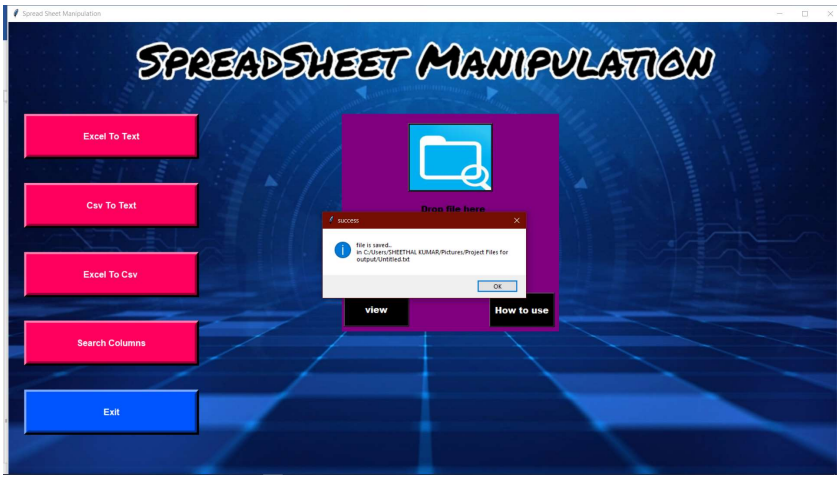
Invalid Extension for saving File:



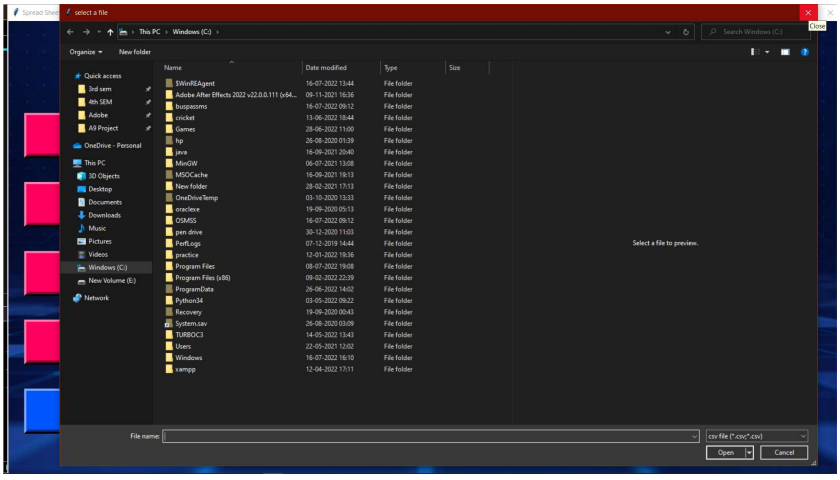


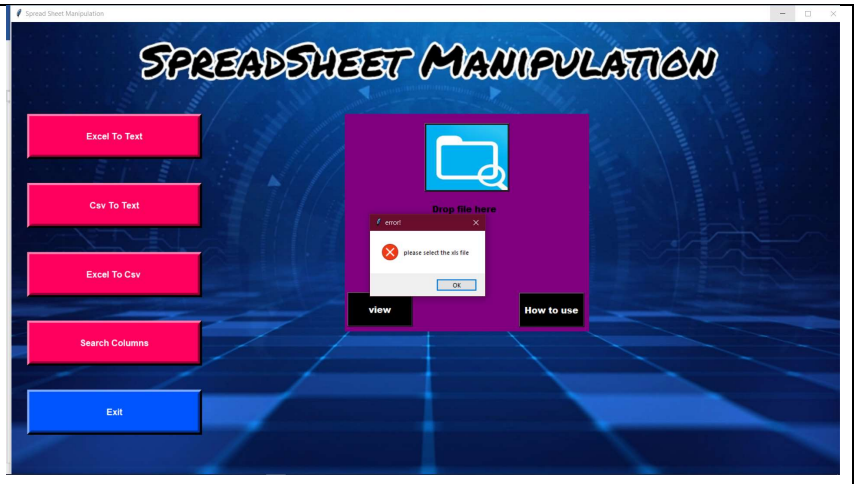
Valid input:



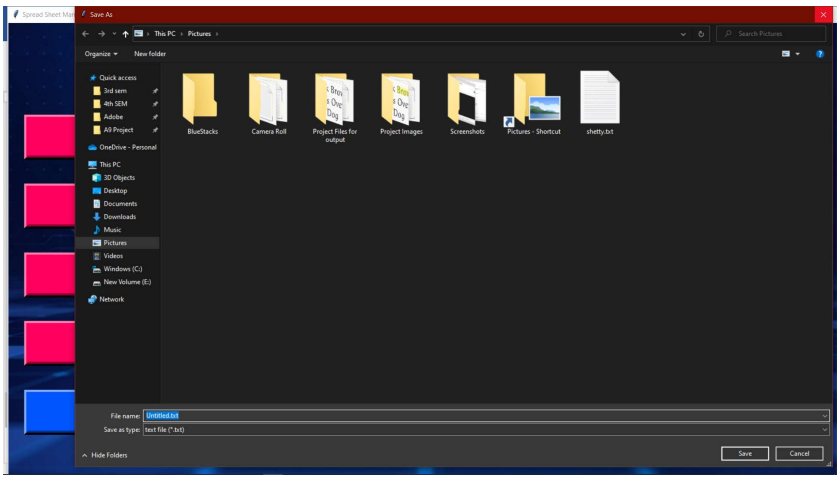


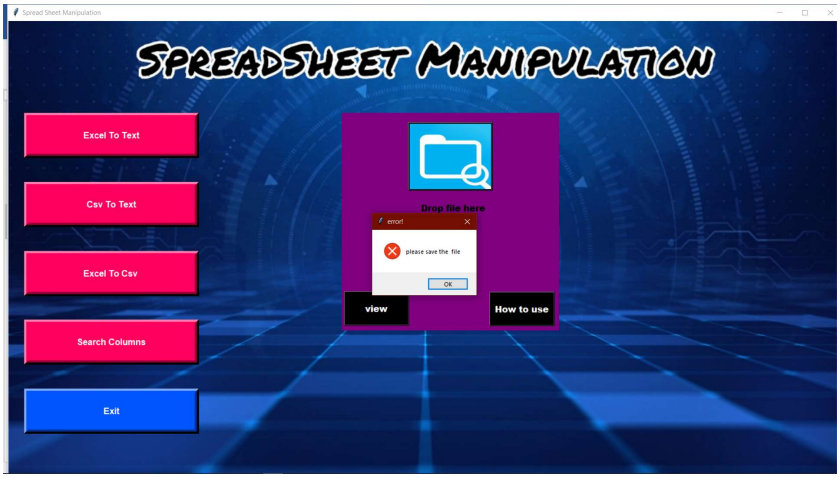
Invalid selection:

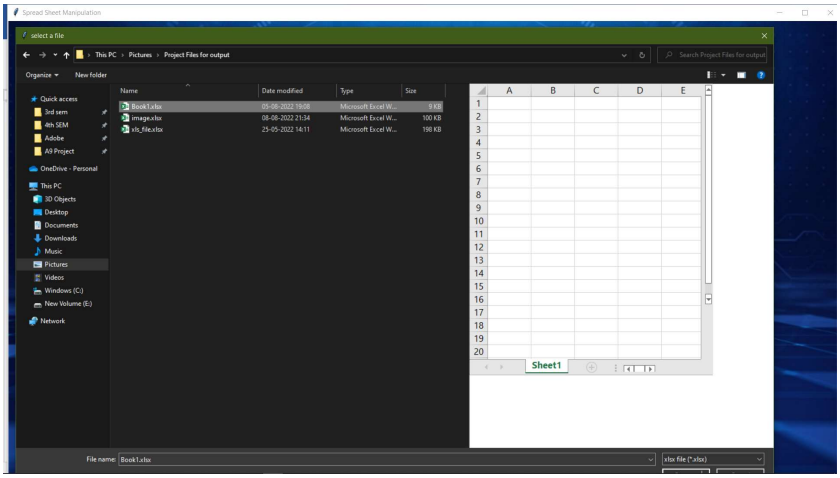


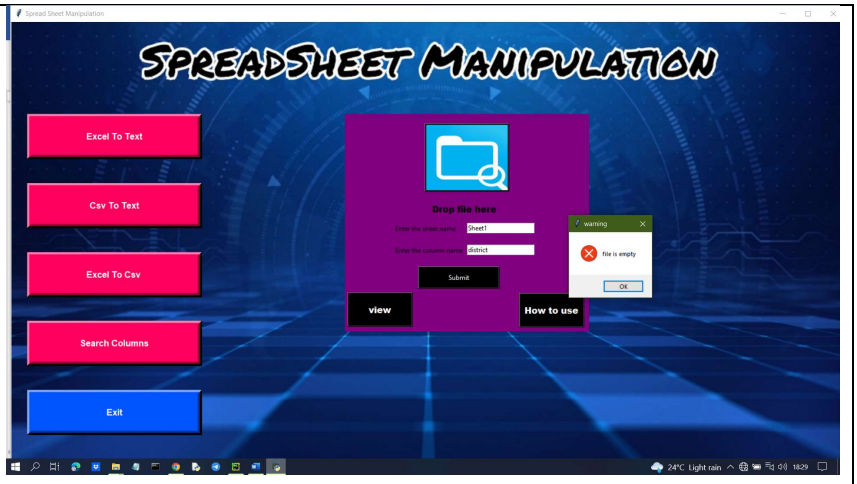


Invalid saving:

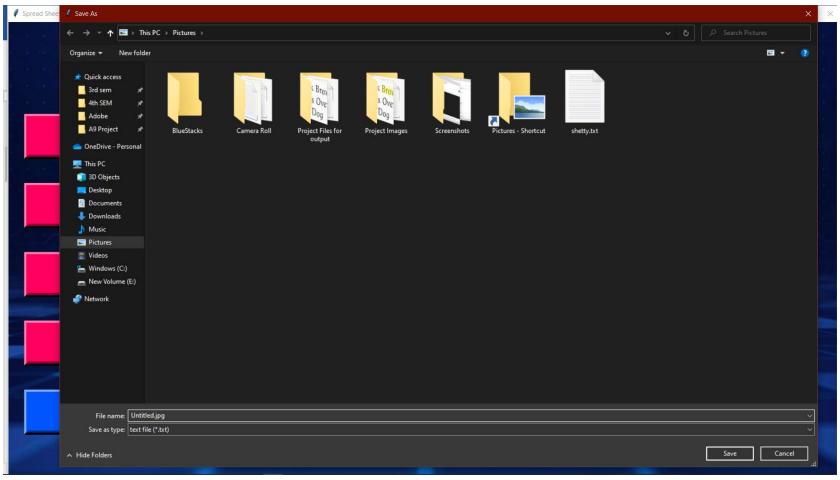


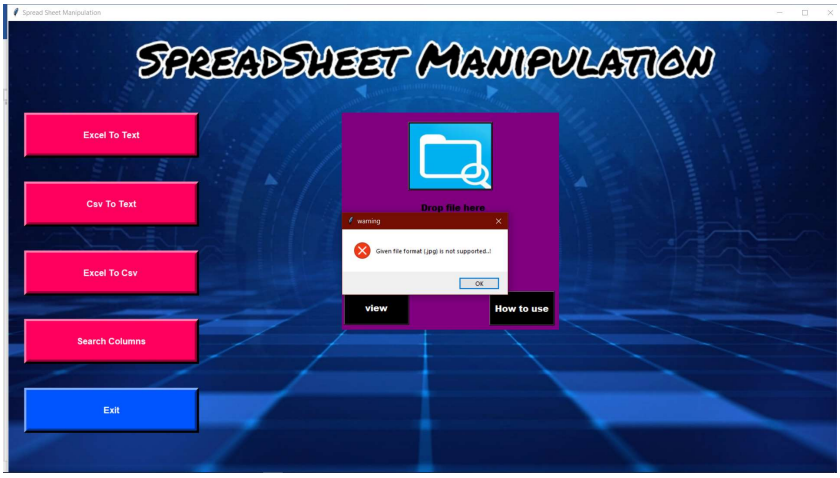


Test case-ID	07
Test case Title	Searching sheet name and column name
Purpose of testing	Testing for Excel file contains data or not using sheet name and column name
Test Data	Excel File
Steps	Step 1: if Excel file contains data then Step 2: convert to csv file using sheet name and column name and display location of file Step 3:ELSE Display Error message Invalid input: 

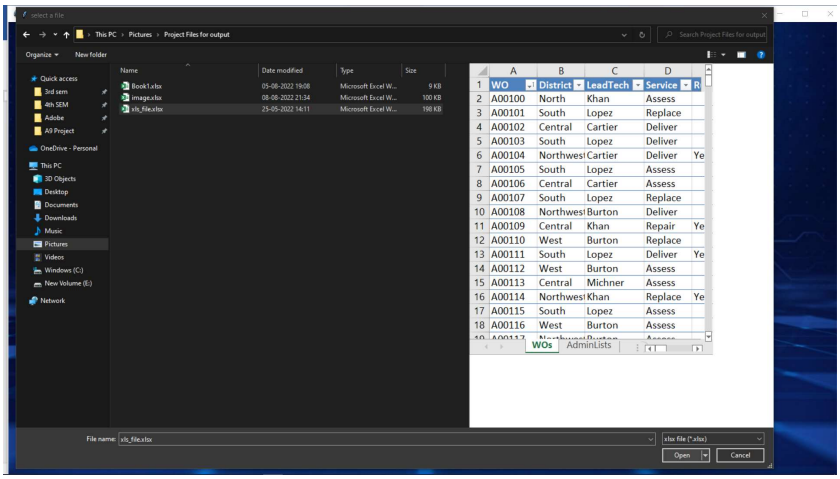


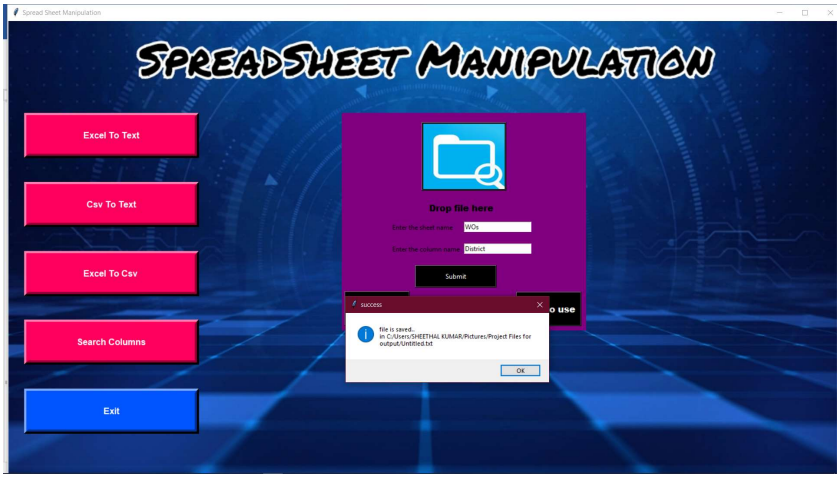
Invalid Extension for saving File:



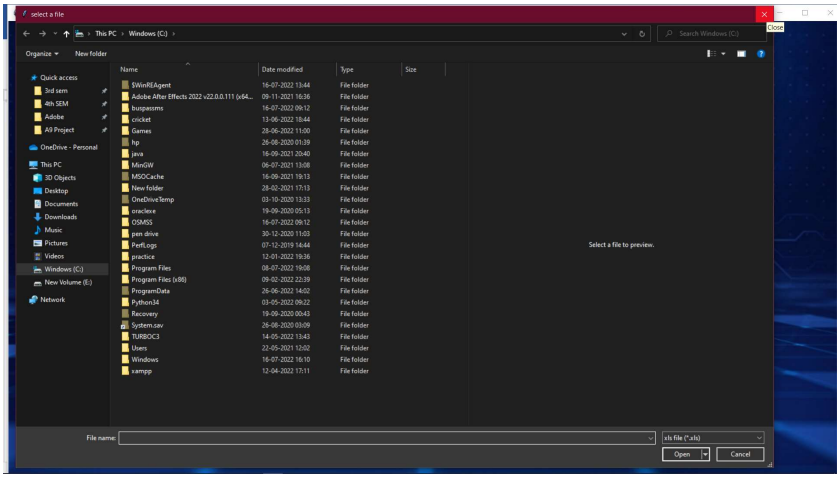


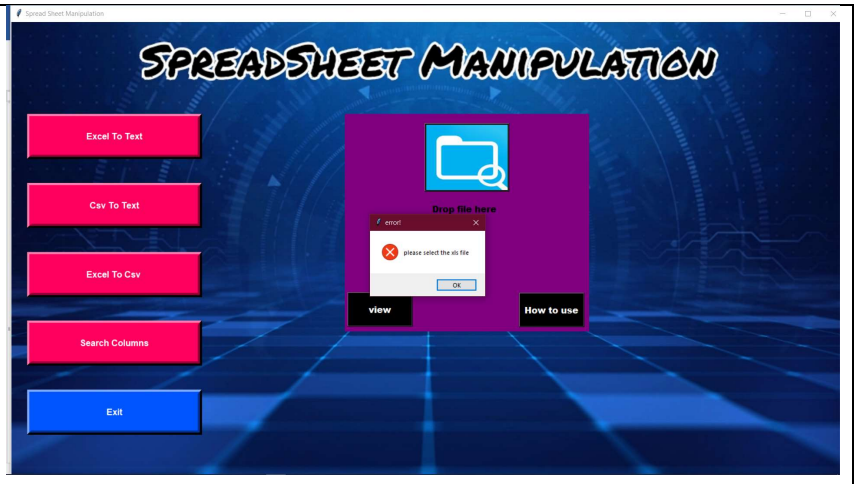
Valid input:



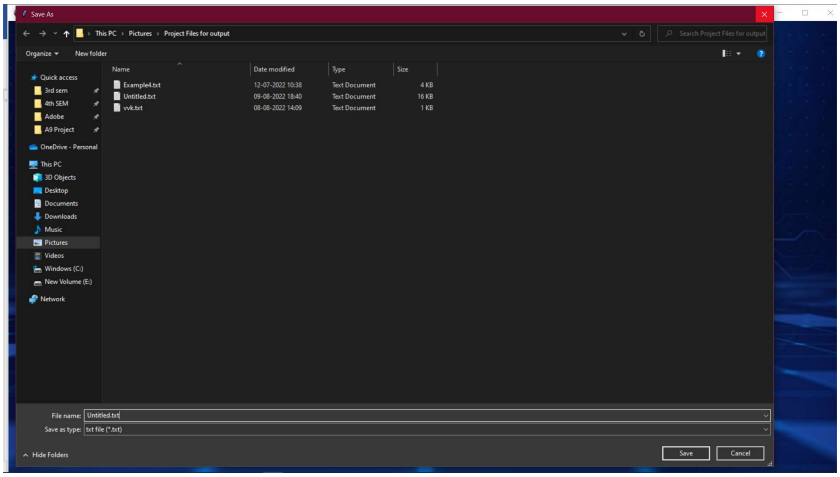


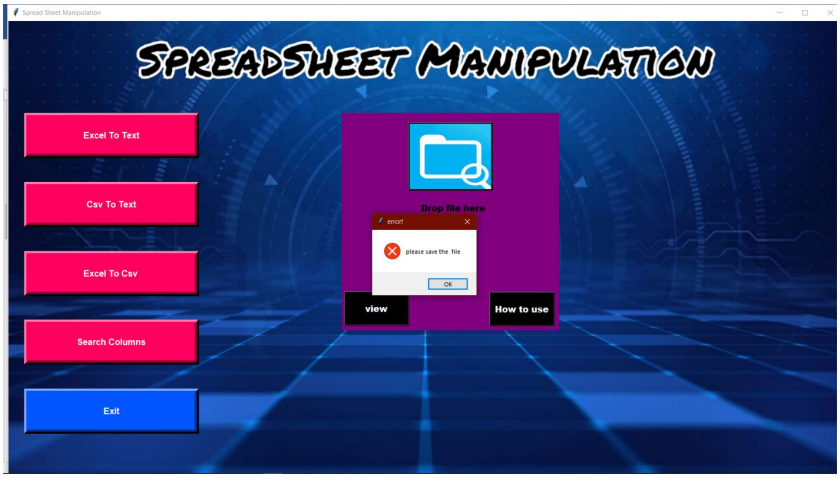
Invalid Selection:

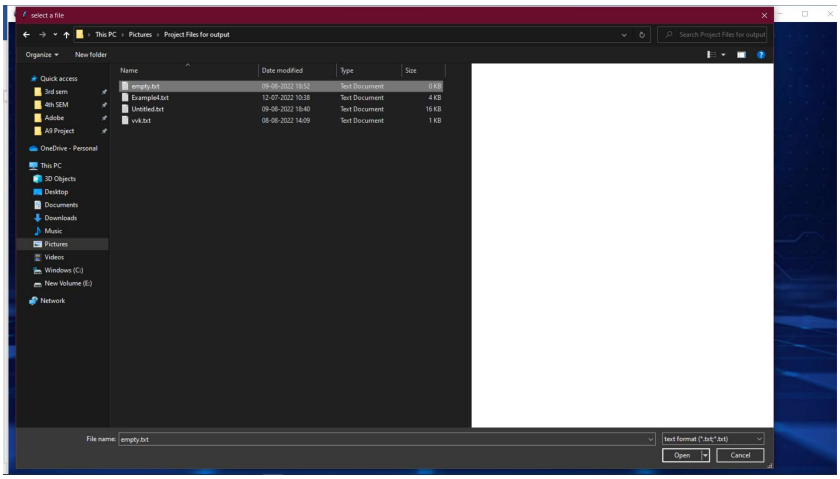
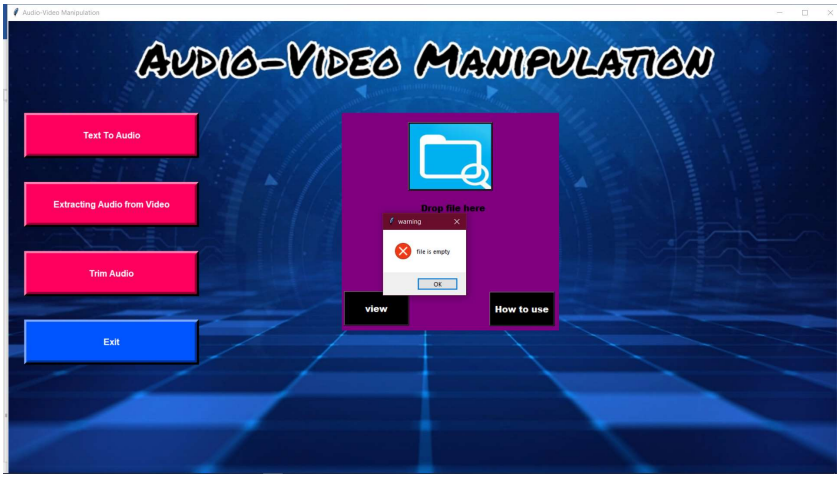




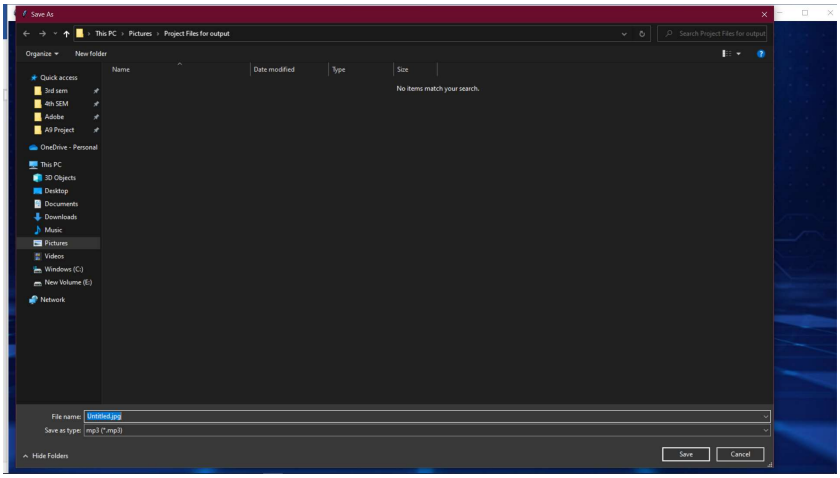
Invalid saving:

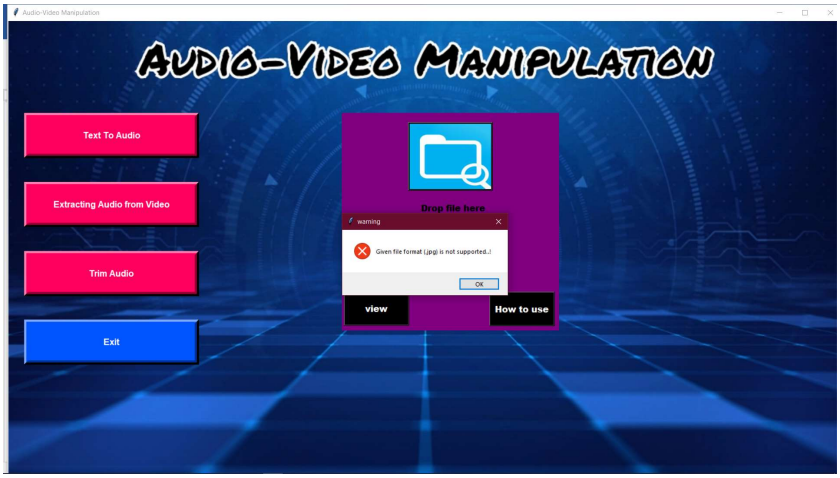




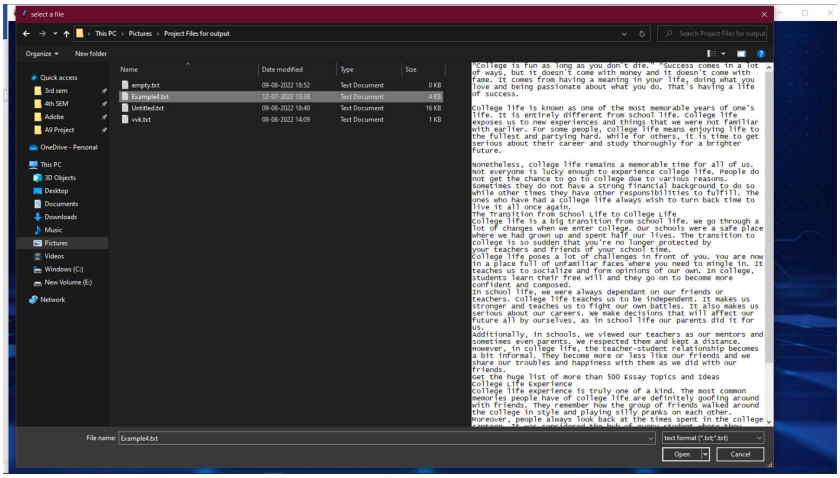
Test case-ID	08
Test case Title	Text file Empty or not
Purpose of testing	Testing for Excel file contains data or not using sheet name and column name
Test Data	Text File
Steps	Step 1: if text file contains data then Step 2: convert to audio and display location of file Step 3:ELSE Display Error message Invalid input:  

Invalid Extension for saving File:

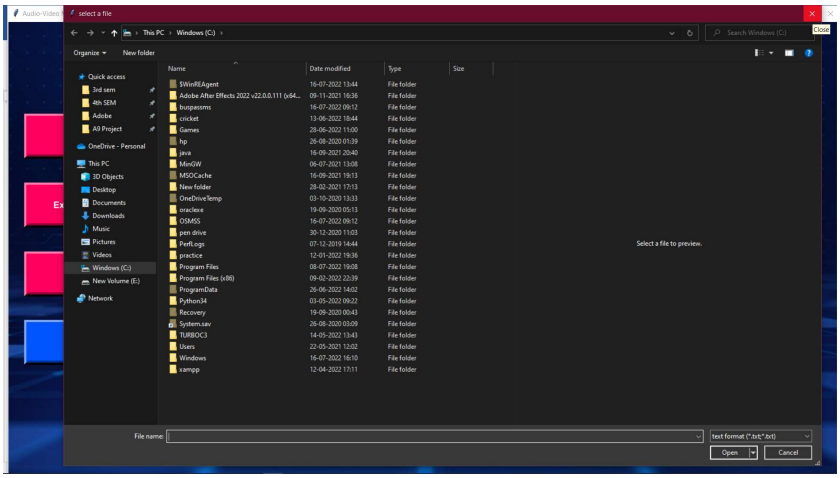




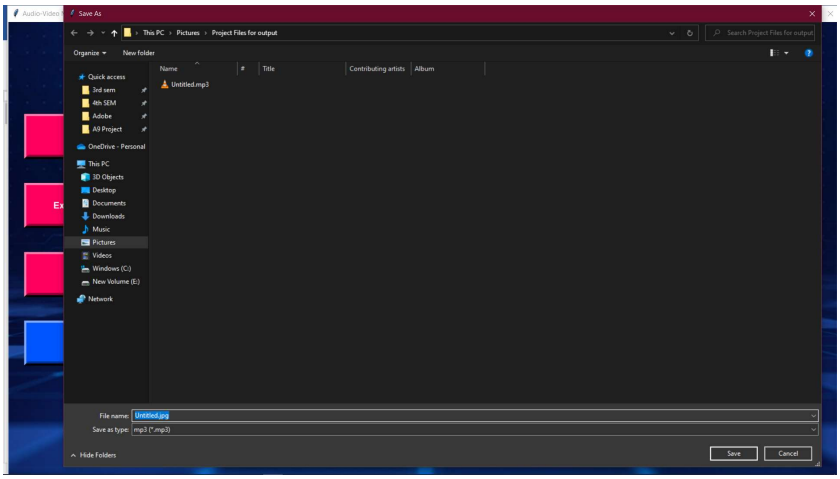
Valid input:

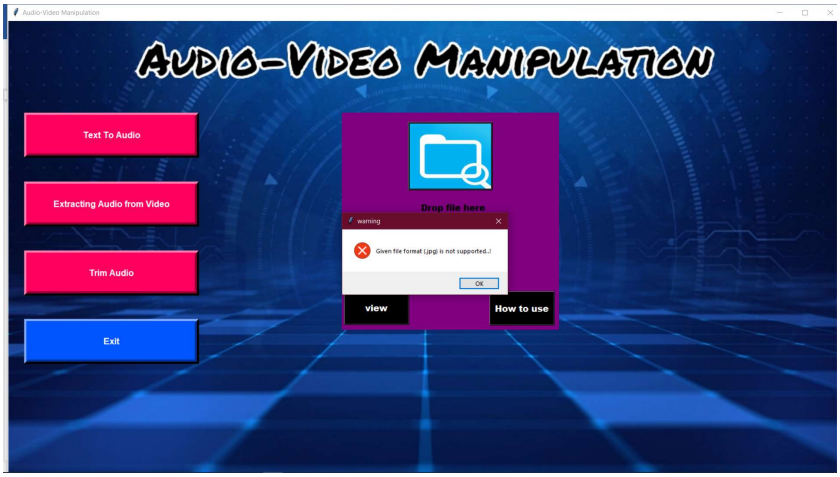


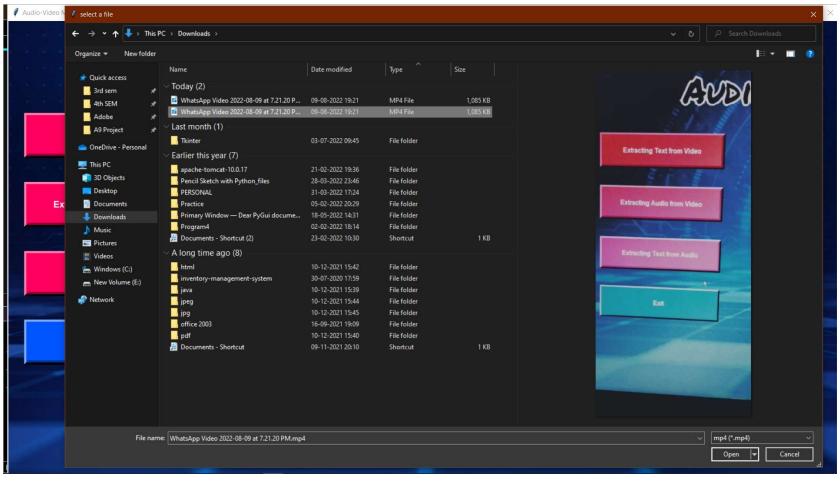
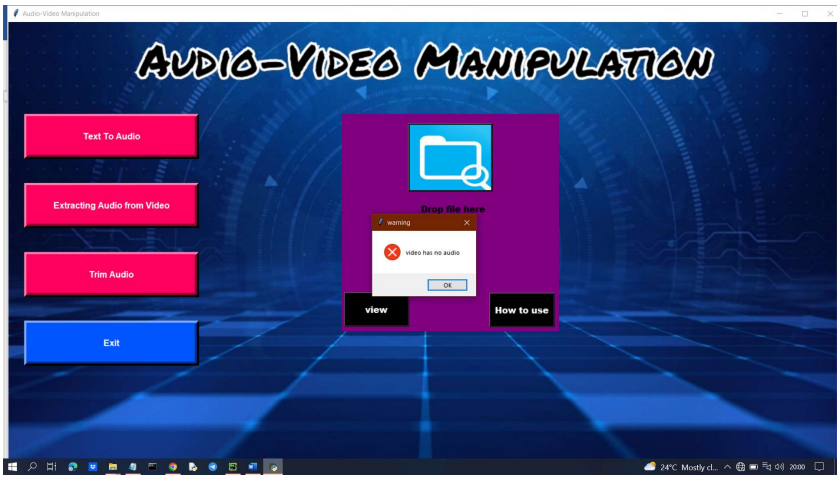
Invalid Selection:



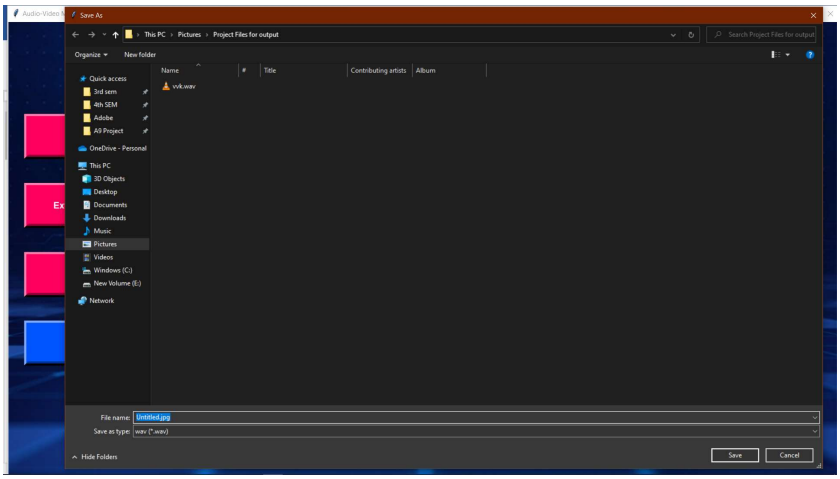
Invalid saving:

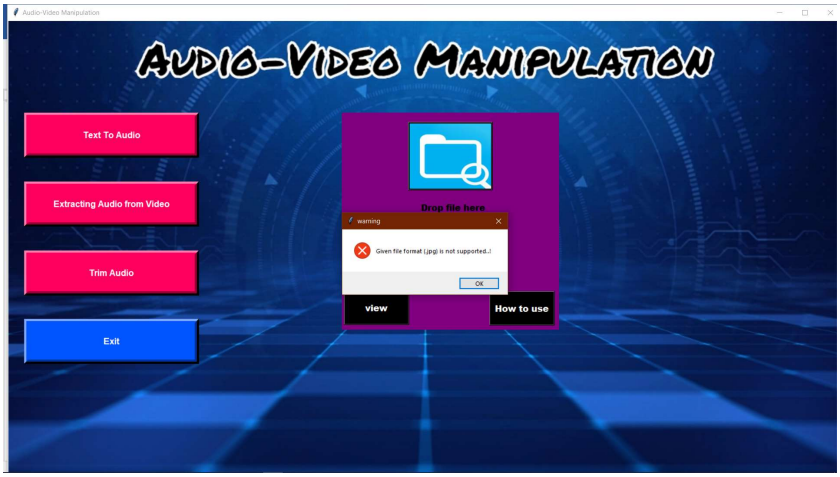




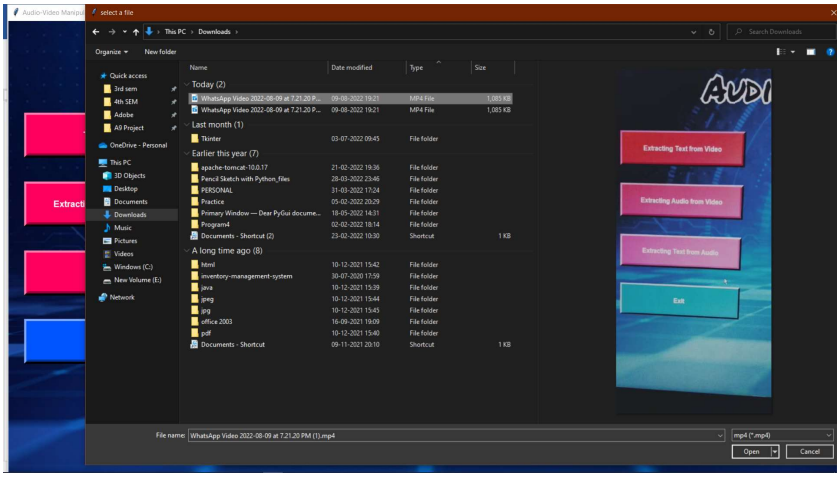
Test case-ID	09
Test case Title	Video to audio
Purpose of testing	Testing for Wheather video contains audio or not
Test Data	Video file
Steps	Step 1: if video file contains audio then Step 2: save the audio and display location of file Step 3:ELSE Display Error message Invalid input:  

Invalid Extension for saving File:

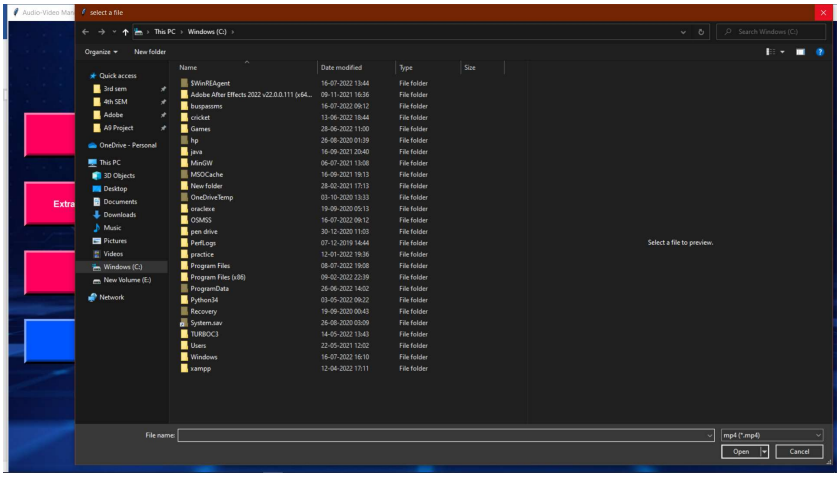


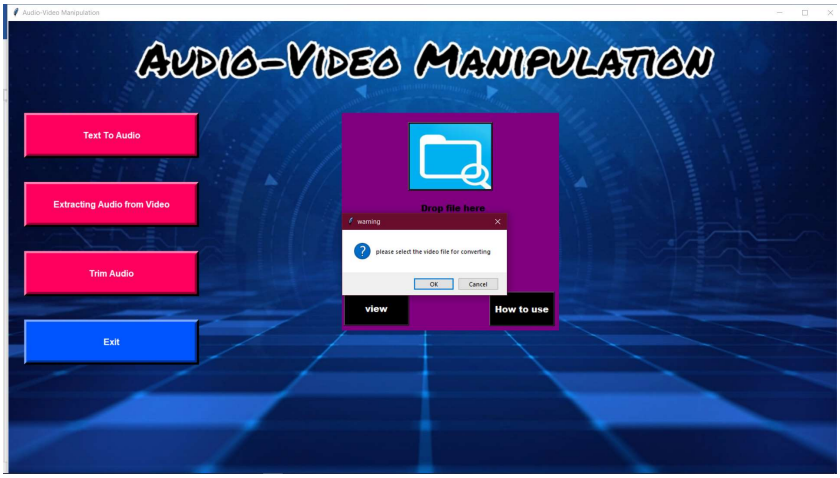


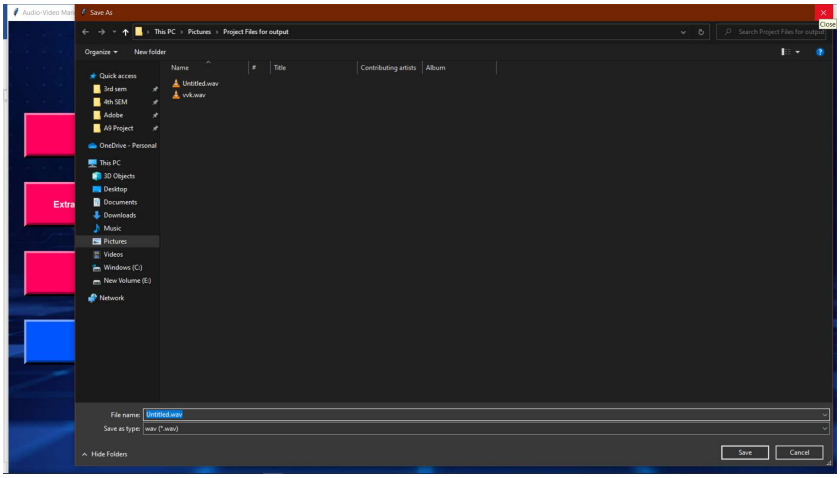
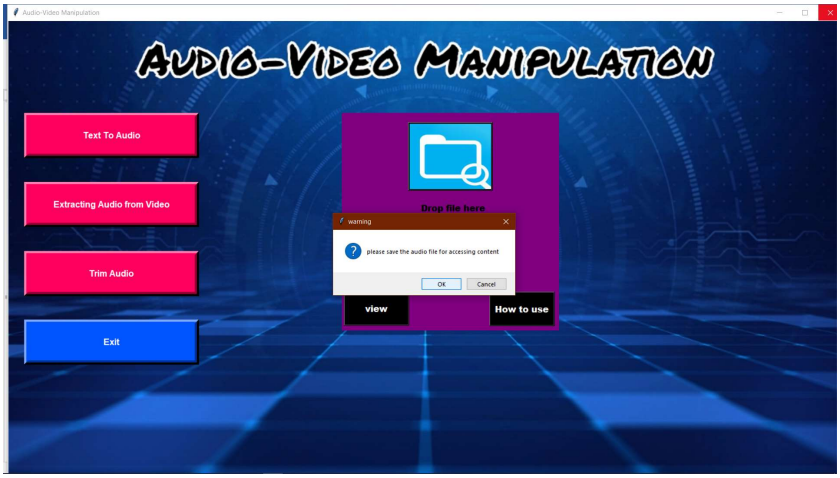
Valid input:

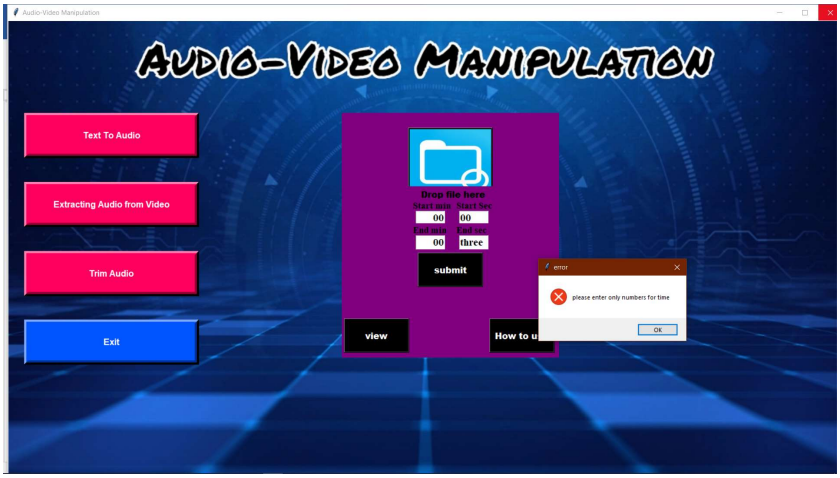


Invalid Selection:

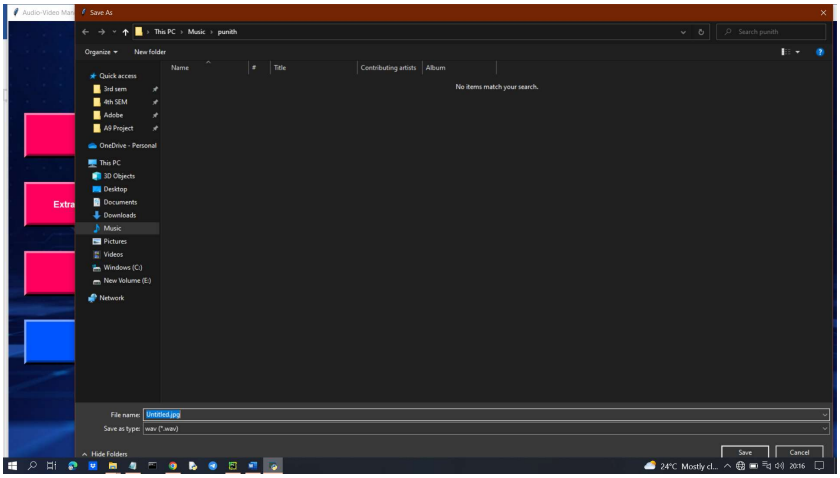




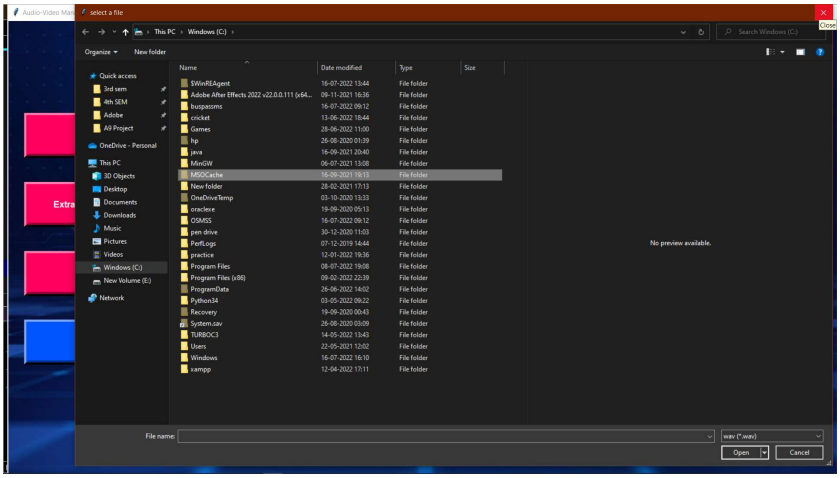
	Invalid Saving:  

Test case-ID	10
Test case Title	Trim audio
Purpose of testing	Testing for correct data entered for trimming audio
Test Data	Audio file
Steps	Step 1: if selected audio file then Step 2: trim the audio with respect to minute and second and display location of file Step 3:ELSE Display Error message Invalid input:  Valid input: 

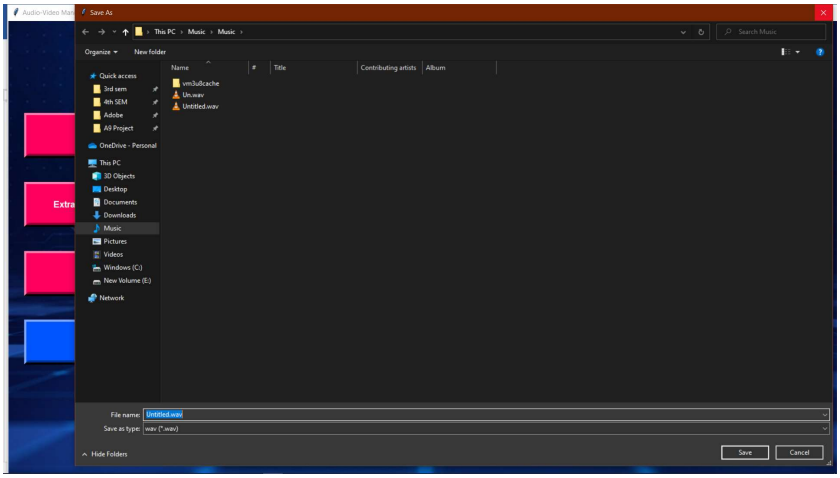
Invalid Extension for saving File:

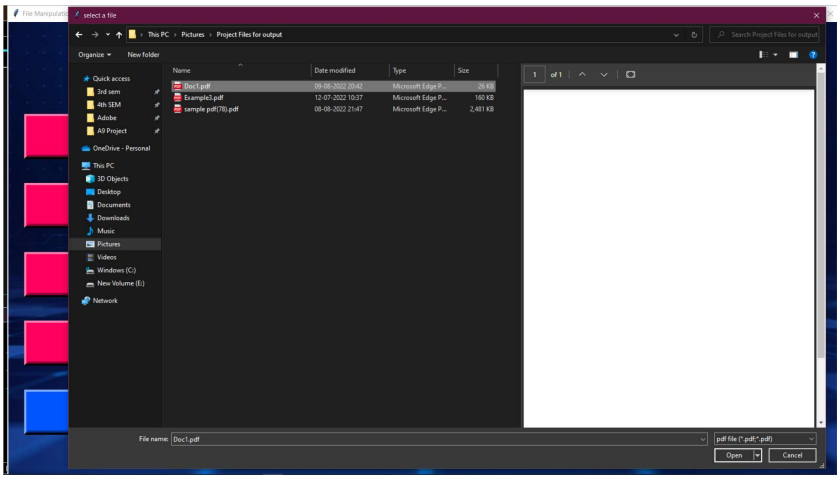
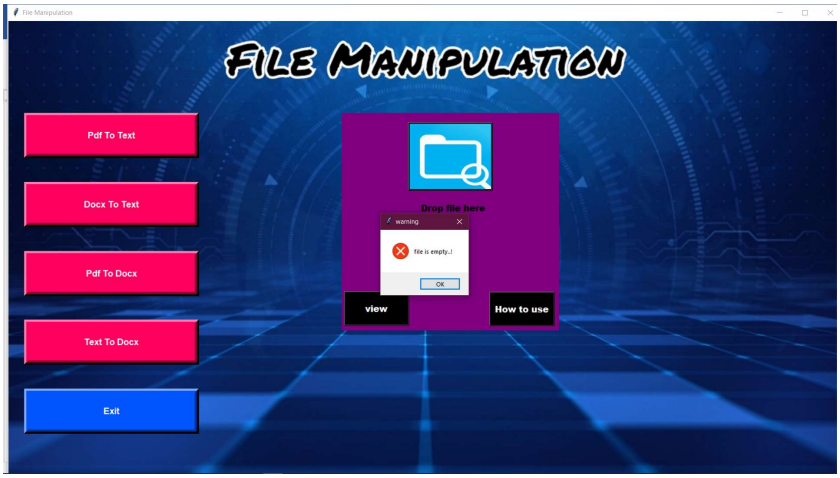


Invalid Selection:

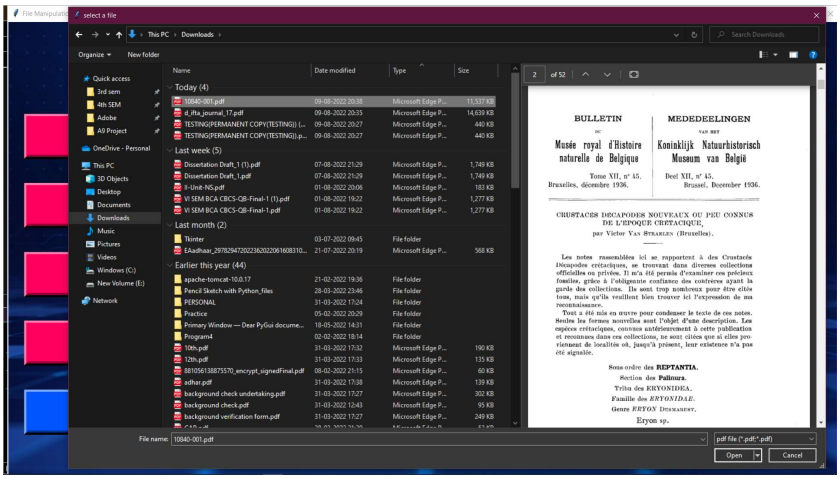


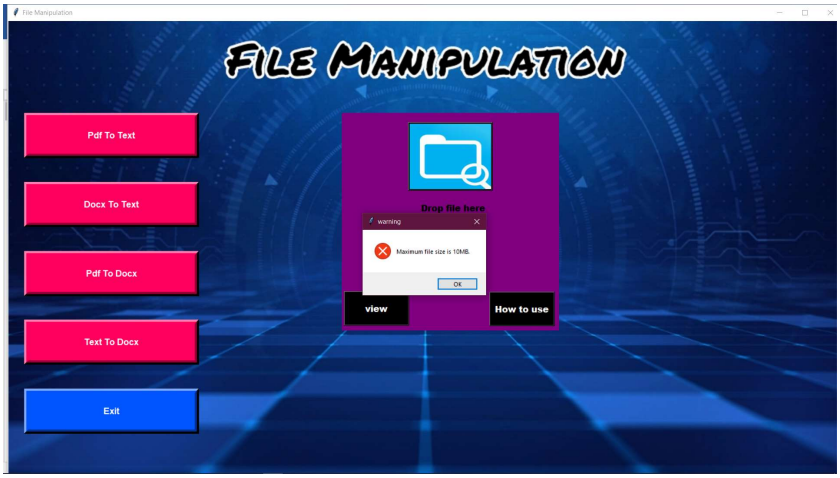
Invalid saving:



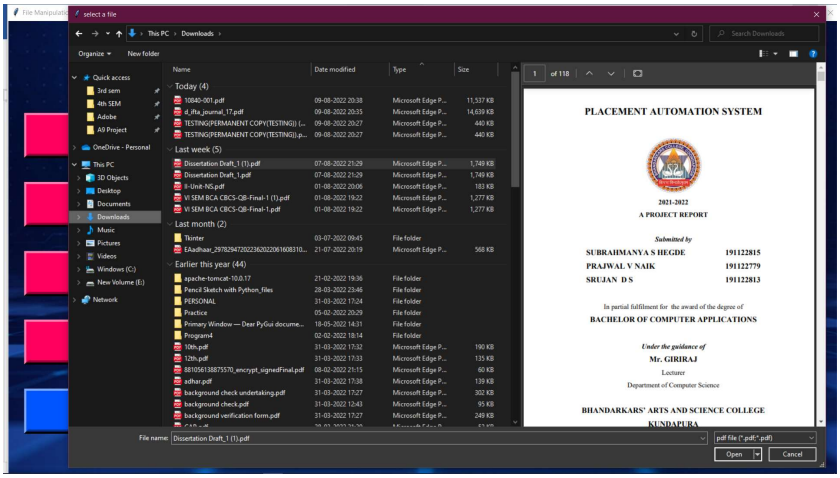
Test case-ID	11
Test case Title	Pdf file
Purpose of testing	Testing for Proper size document
Test Data	Pdf file
Steps	Step 1: if selected pdf file then Step 2: convert to text and display location of file Step 3:ELSE Display Error message Invalid input:  

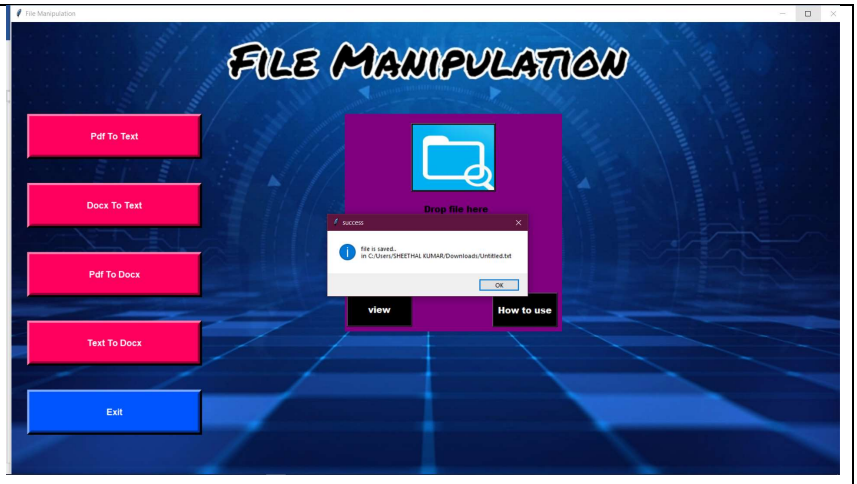
Valid maximum file size:



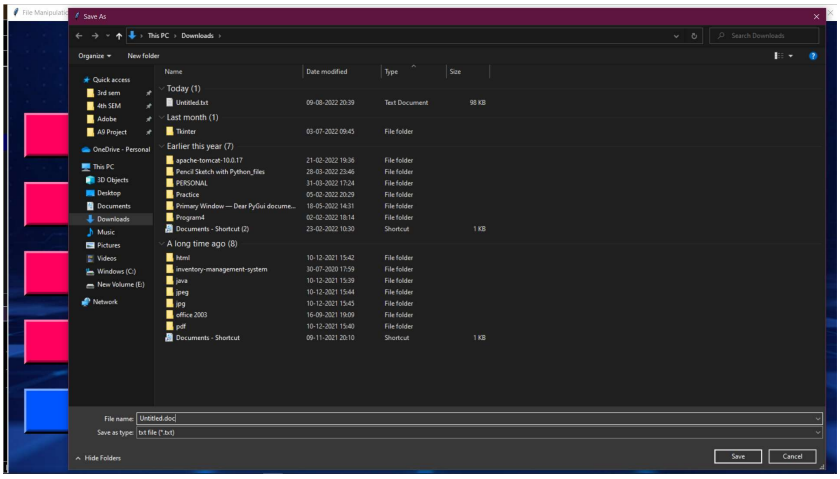


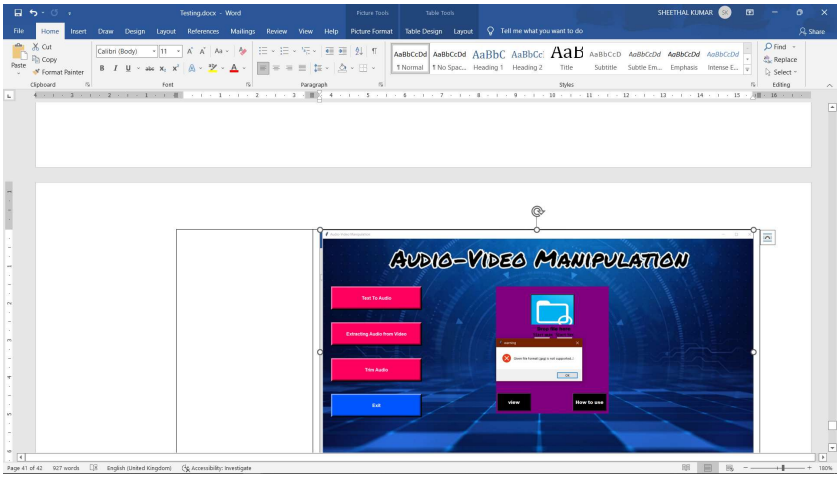
Valid input:



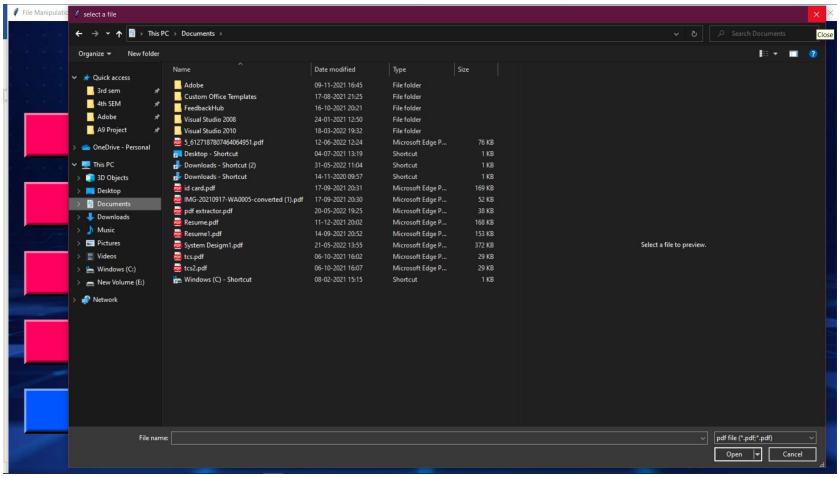


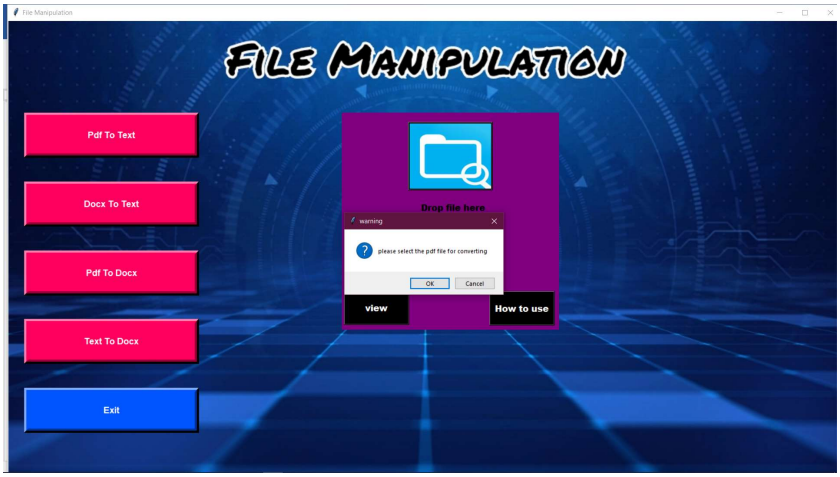
Invalid Extension for saving File:



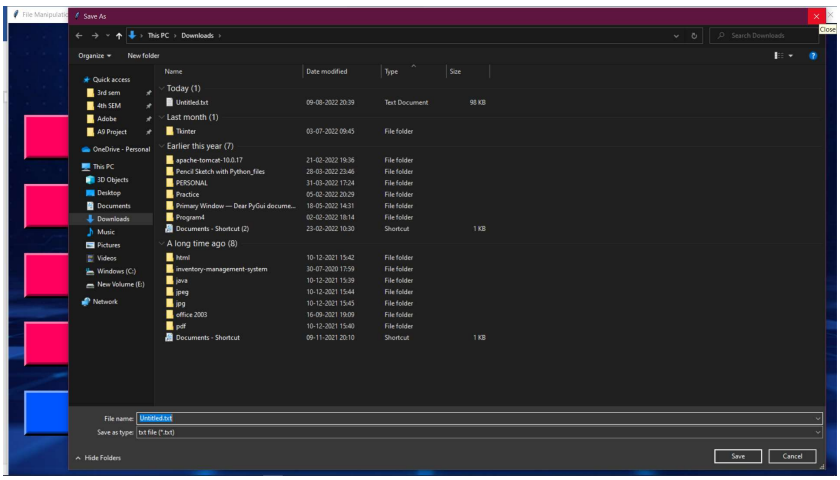


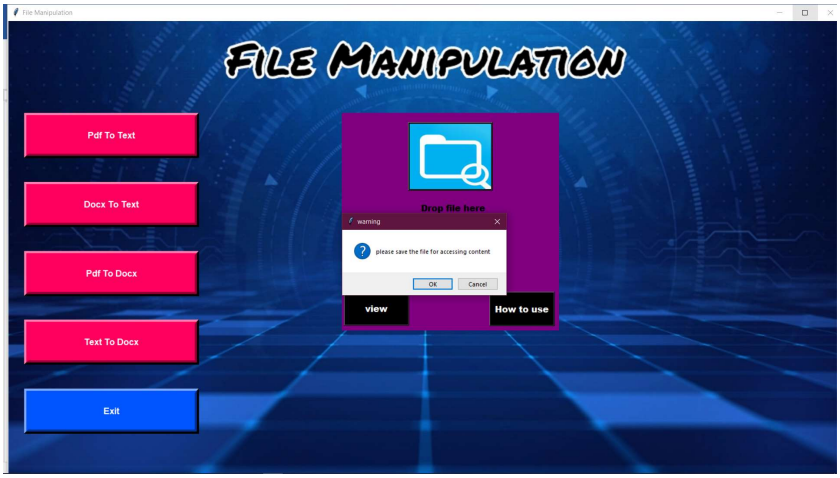
Invalid Selection:

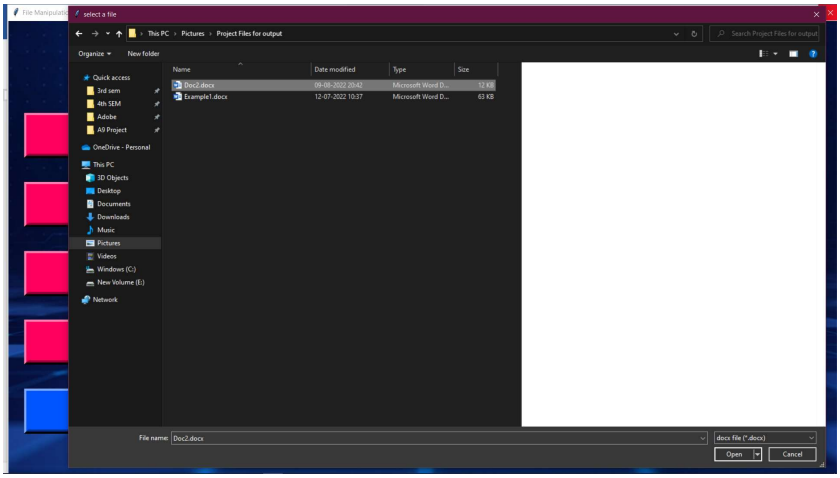


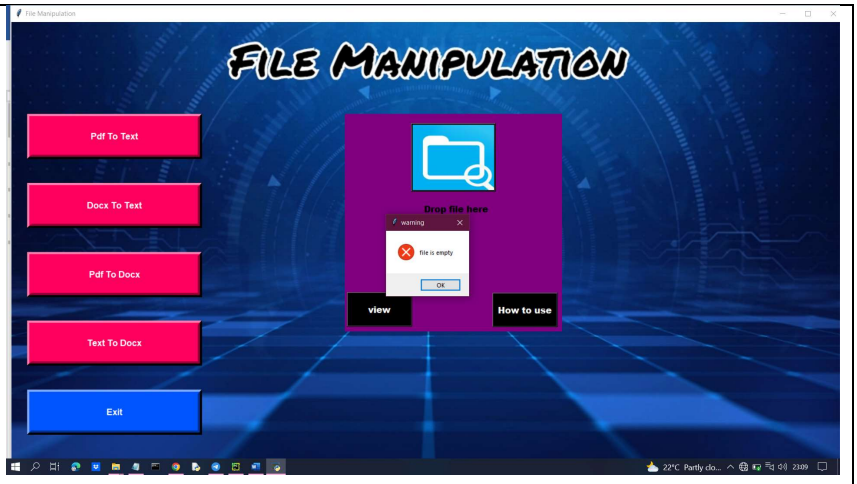


Invalid saving:

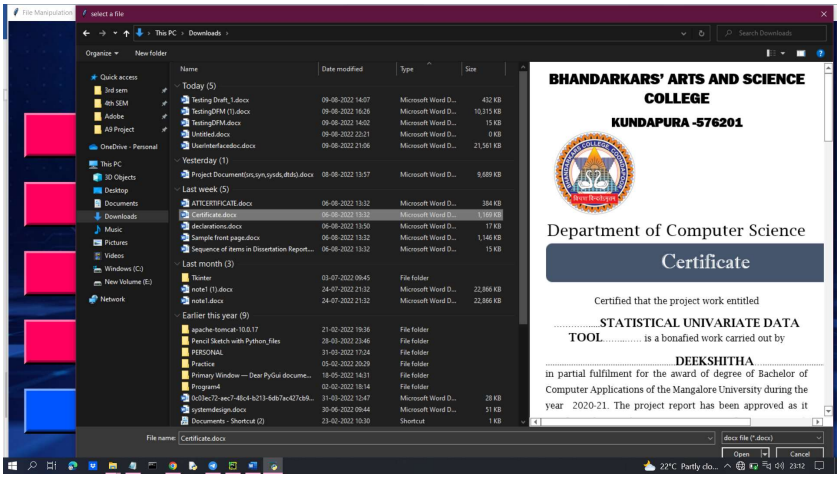


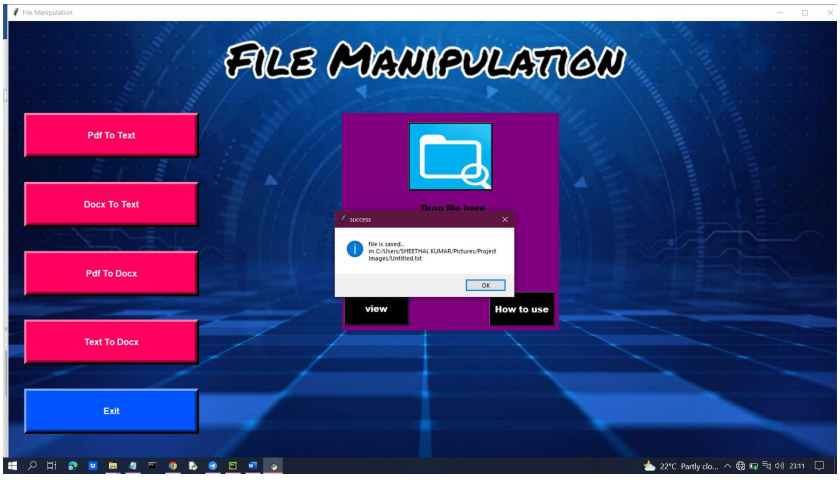

File Manipulation
FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
Drop file here
warning
please save the file for accessing content
OK
Cancel
view
How to use

Test case-ID	12
Test case Title	Docx file
Purpose of testing	Testing for docx operation
Test Data	docx file
Steps	Step 1: if selected pdf file then Step 2: convert to text and display location of file Step 3:ELSE Display Error message Invalid input: 

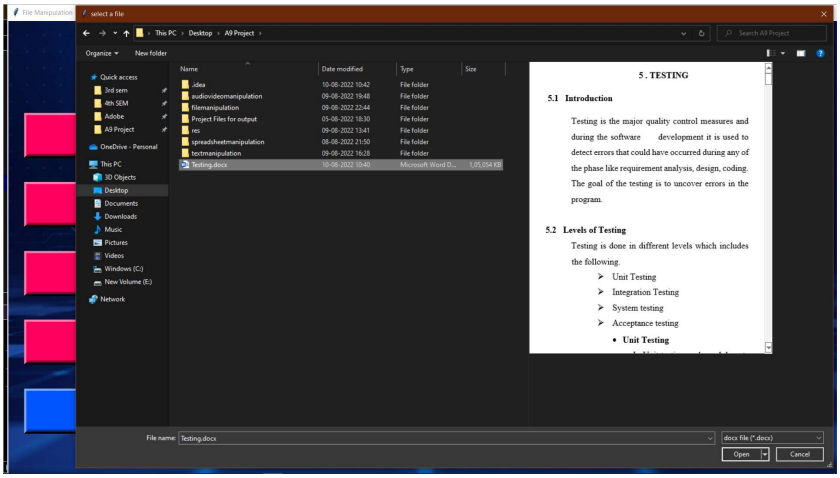

FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
Drop file here
warning
file is empty
OK
view
How to use

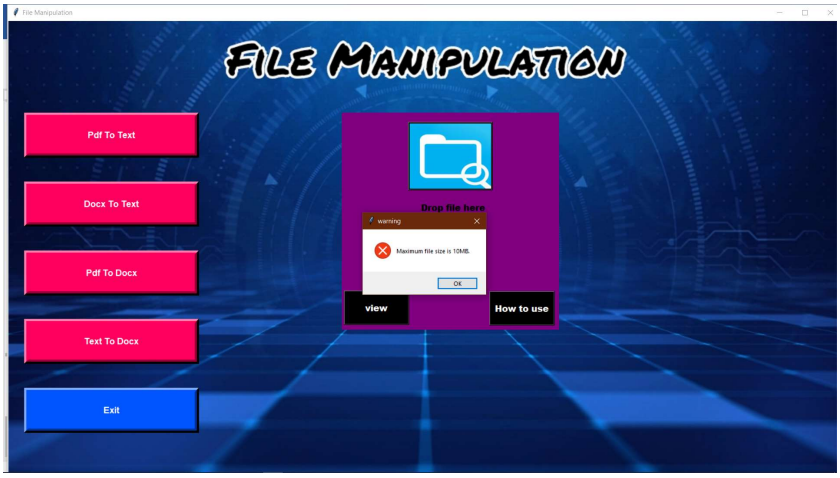
Valid input:


select a file
BHANDARKARS' ARTS AND SCIENCE COLLEGE
KUNDAPURA -576201
Department of Computer Science
Certificate
Certified that the project work entitled
STATISTICAL UNIVARIATE DATA TOOL is a bonafied work carried out by
DEEKSHITHA
in partial fulfilment for the award of degree of Bachelor of Computer Applications of the Mangalore University during the year 2020-21. The project report has been approved as it

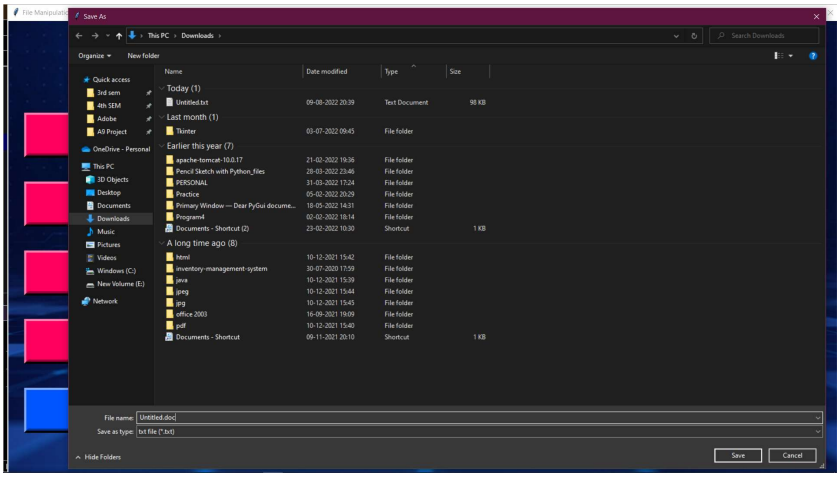

FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
Drop file here
success
file is saved..
in C:/Users/SHEETHAL KUMAR/Pictures/Project Images/Untitled.txt
OK
view
How to use

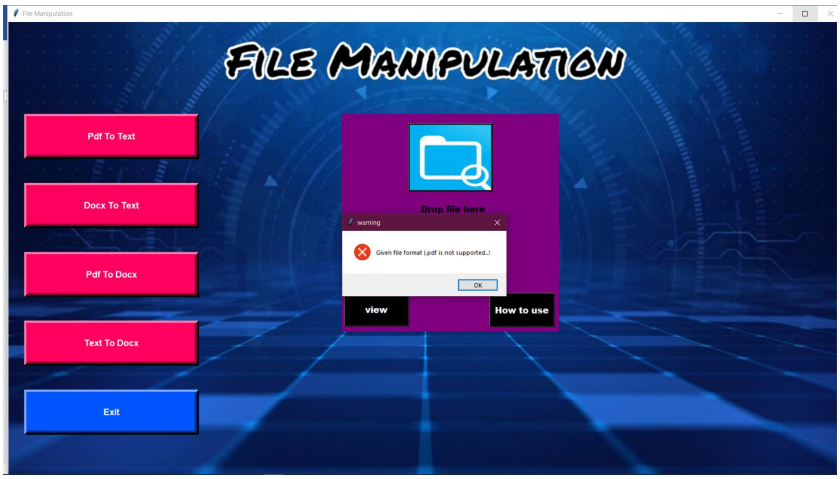
Invalid Maximum file size:



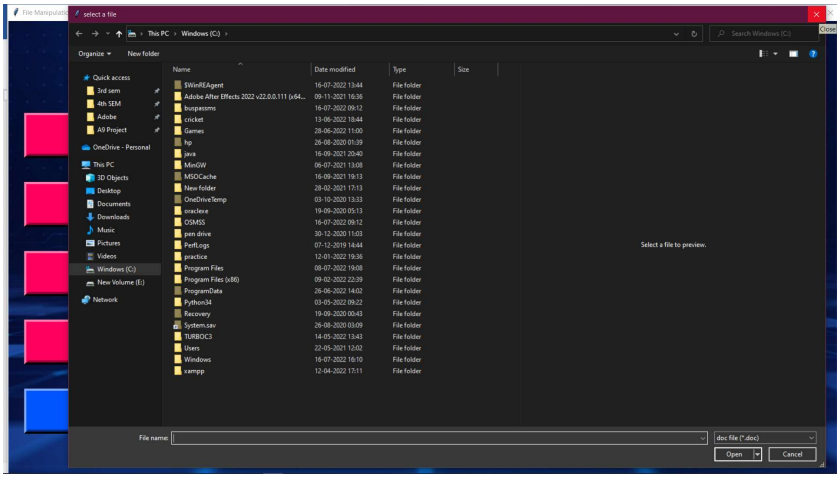


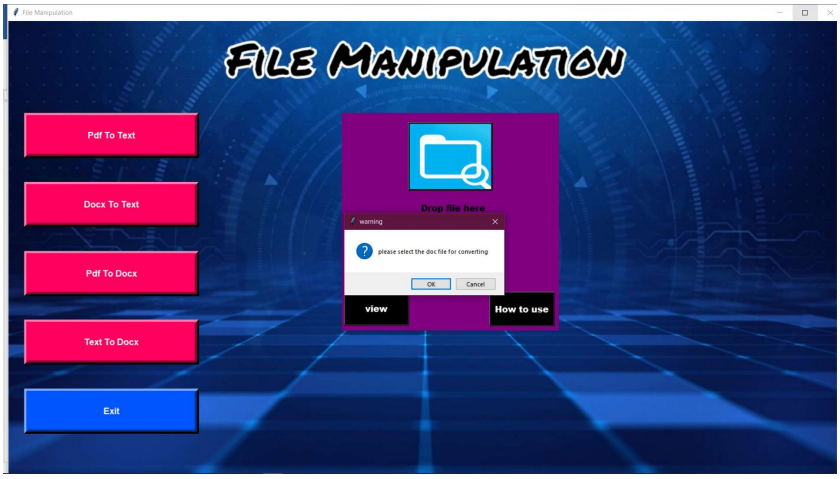
Invalid Extension for saving File:



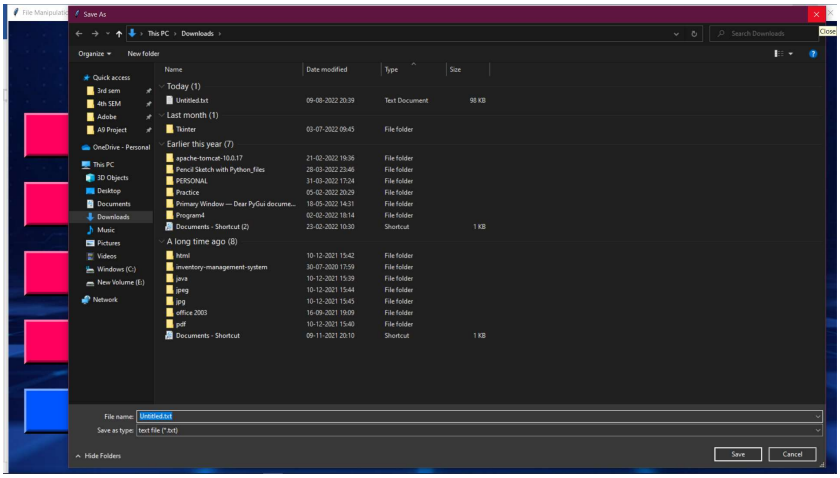


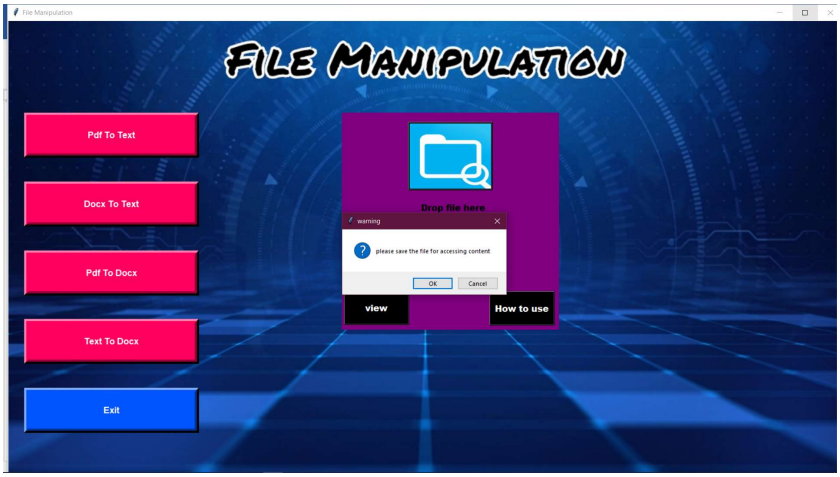
Invalid Selection:

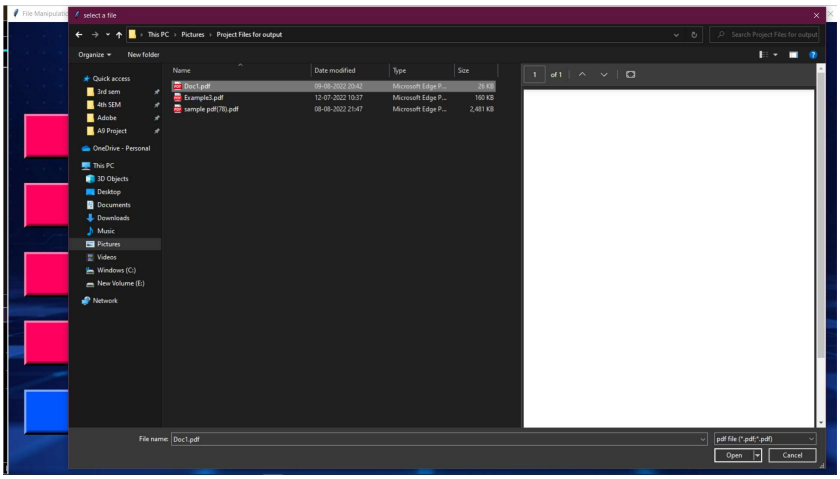


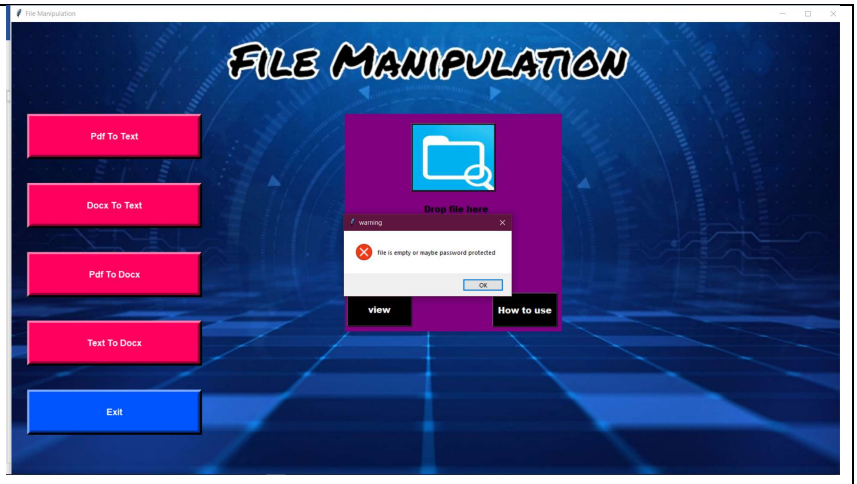


Invalid saving:

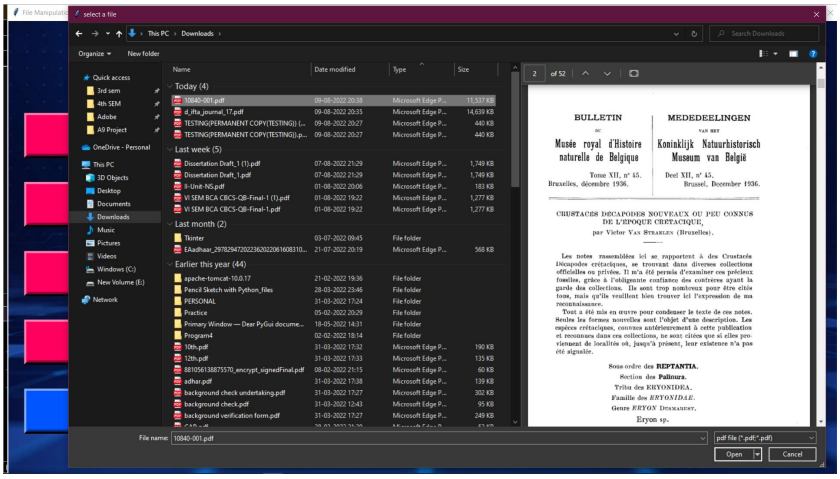


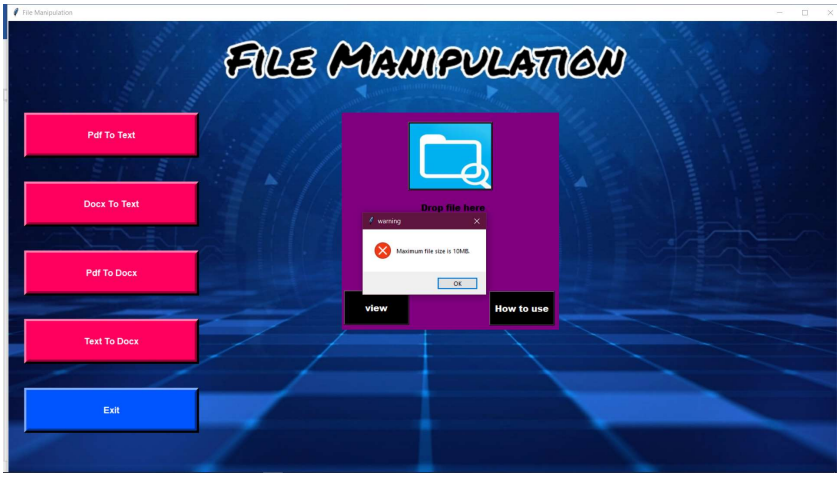


Test case-ID	13
Test case Title	Pdf file
Purpose of testing	Testing for pdf to docx operation
Test Data	Pdf file
	Step 1: if selected pdf file then Step 2: convert to text and display location of file Step 3:ELSE Display Error message Invalid input: 

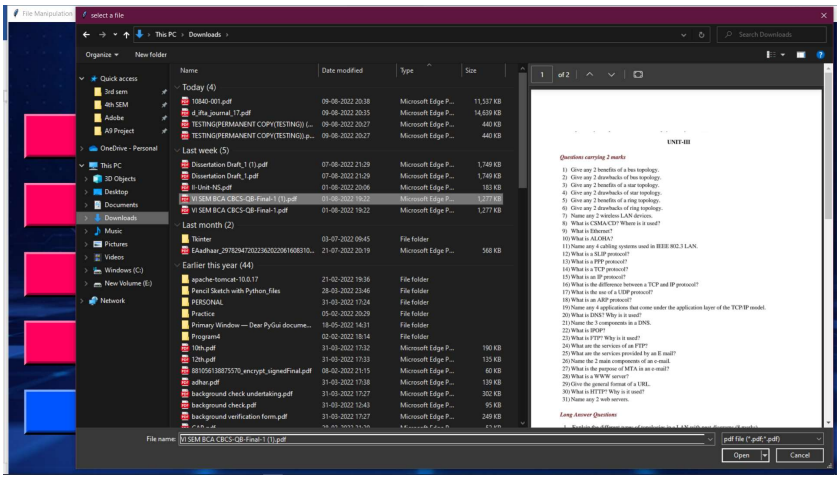


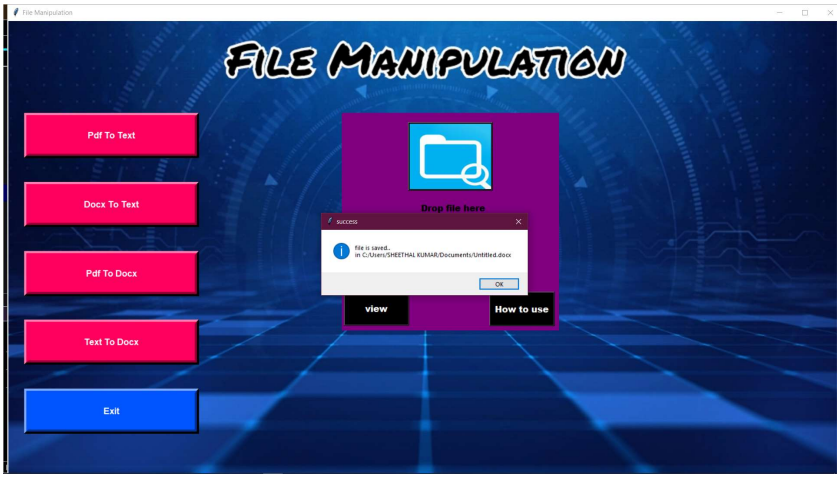
Invalid Maximum Size:



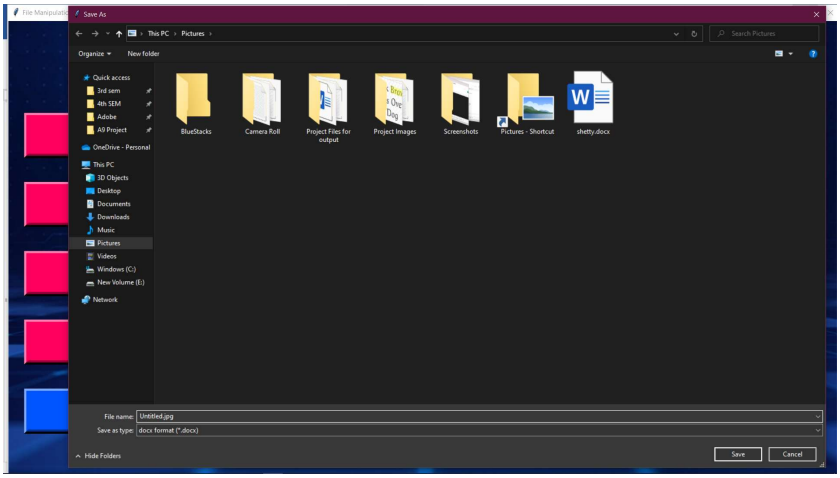


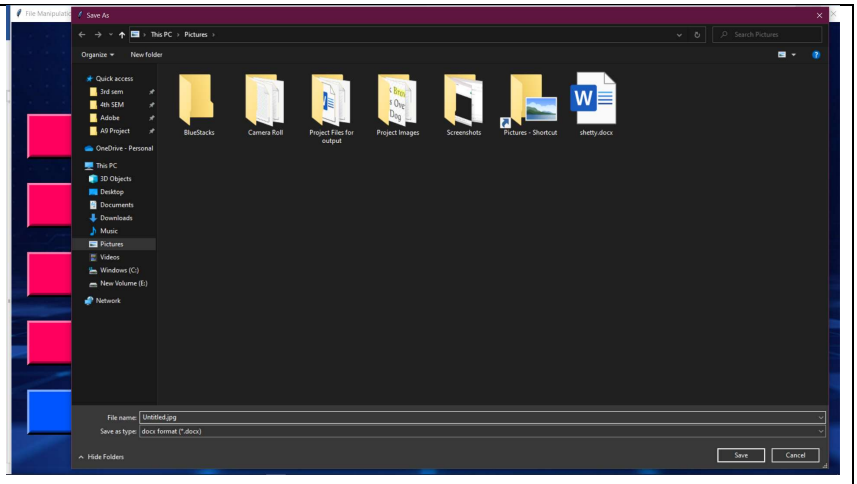
Valid input:



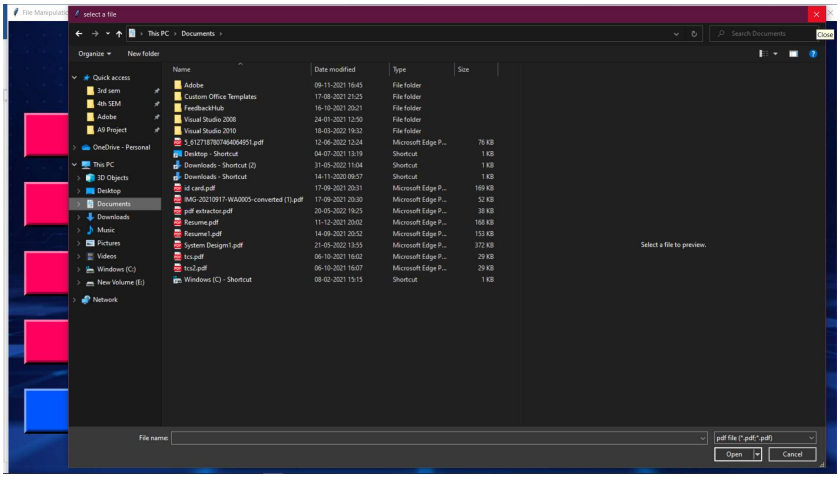


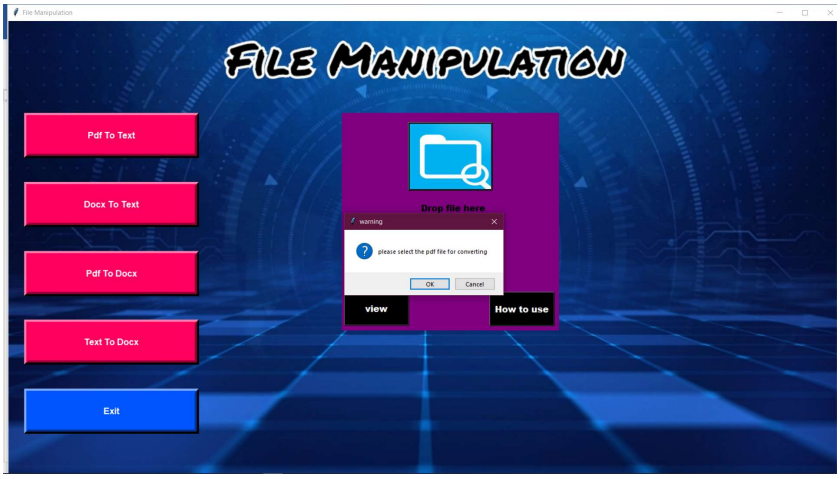
Invalid Extension for saving File:



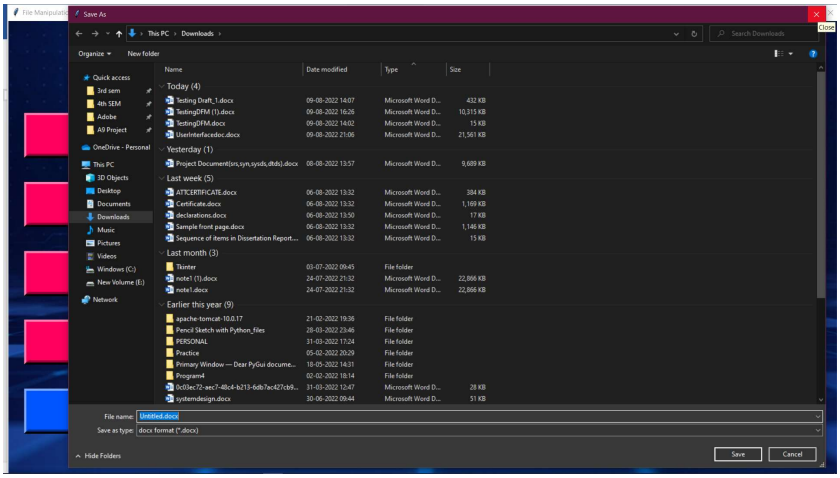


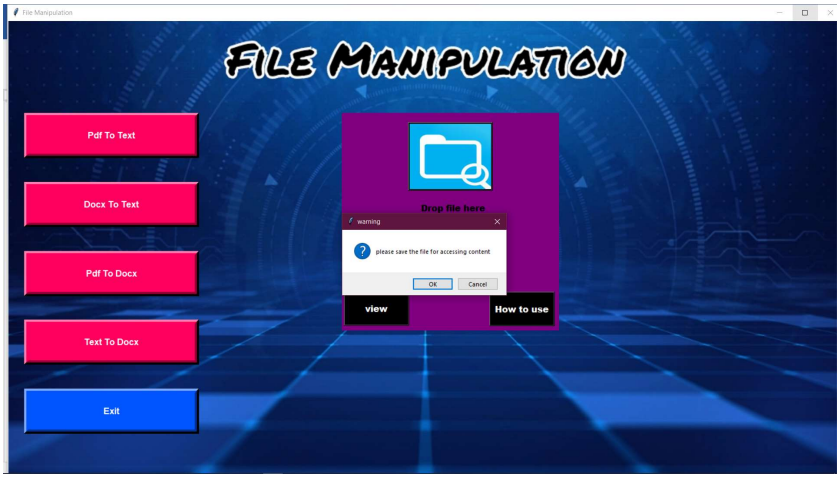
Invalid Selection:

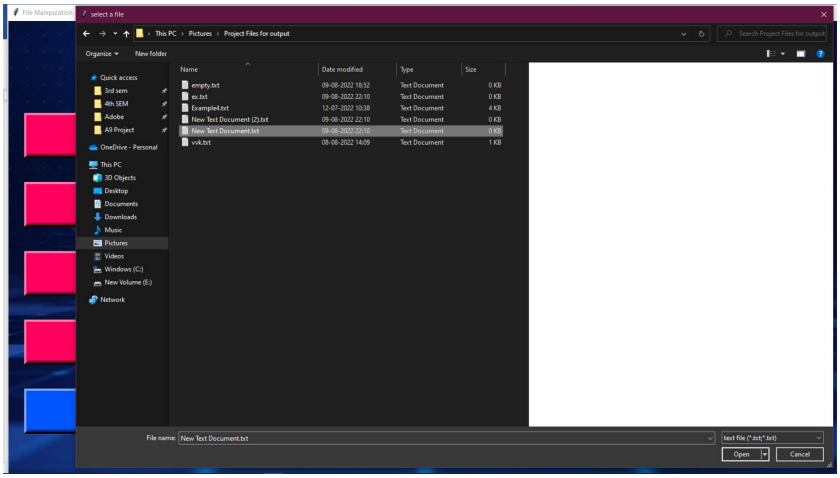


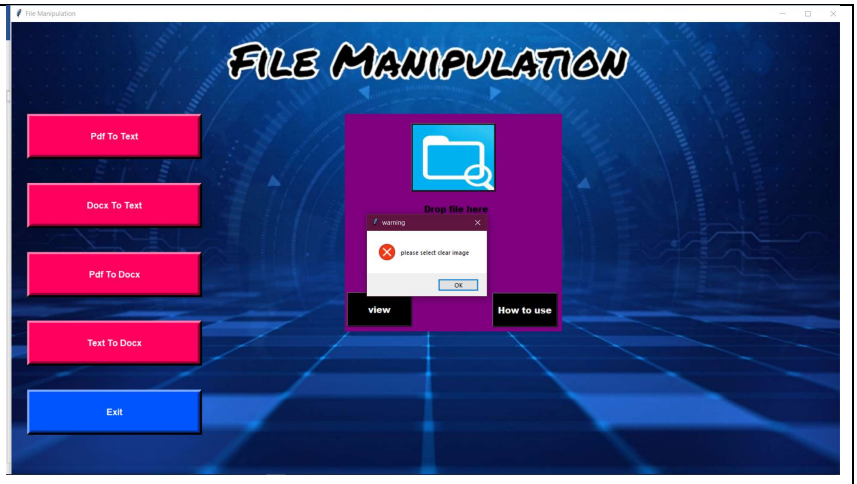


Invalid Saving:

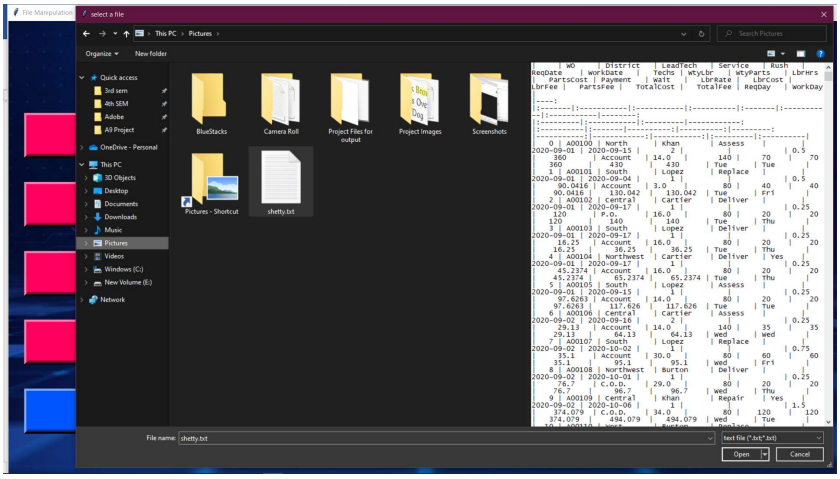


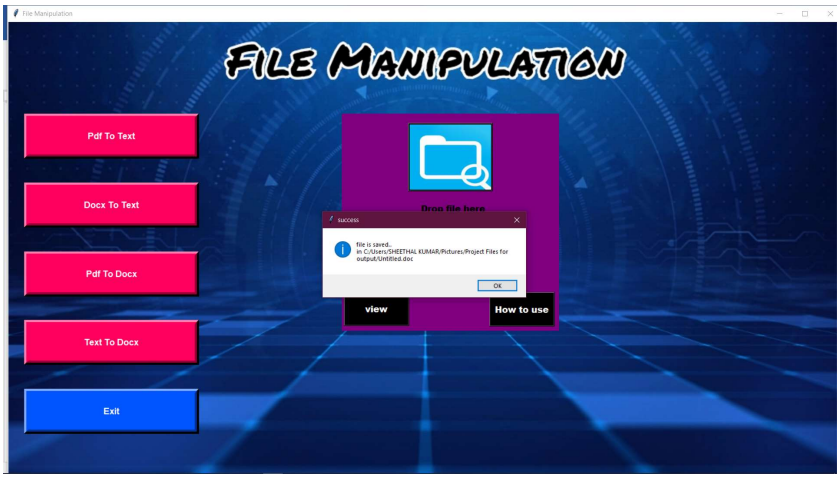


Test case-ID	14
Test case Title	Text file
Purpose of testing	Testing for text to docx operation
Test Data	Text file
Steps	Step 1: if selected text file then Step 2: convert to docx and display location of file Step 3:ELSE Display Error message Invalid input: 

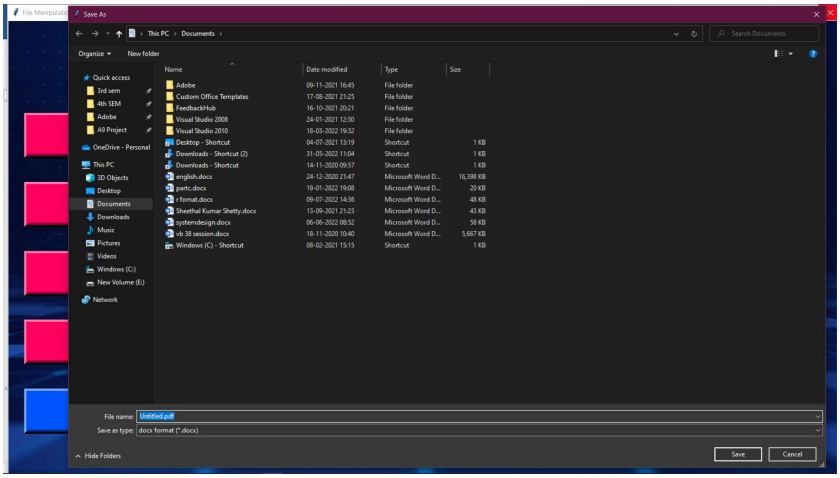

FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
Drop file here
warning
please select clear image
OK
view
How to use

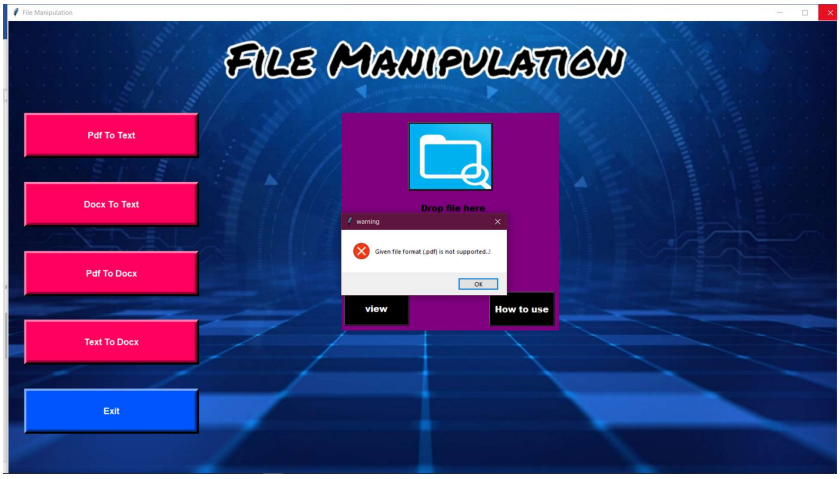
Valid input:


select a file
This PC
Pictures
Organize
New folder
BlueStacks
Camera Roll
Project Files for output
Project Images
Screenshots
Pictures - Shortcut
shetty.txt
File name:
Open
Cancel

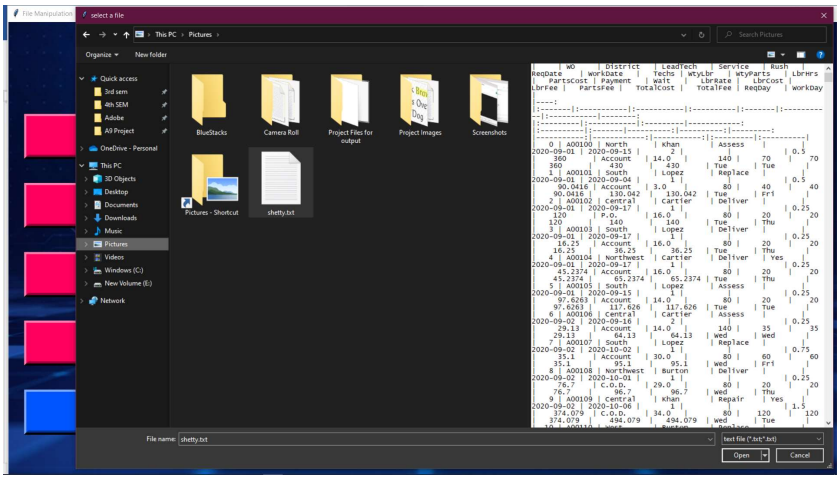

FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
success
file is saved.
OK
view
How to use

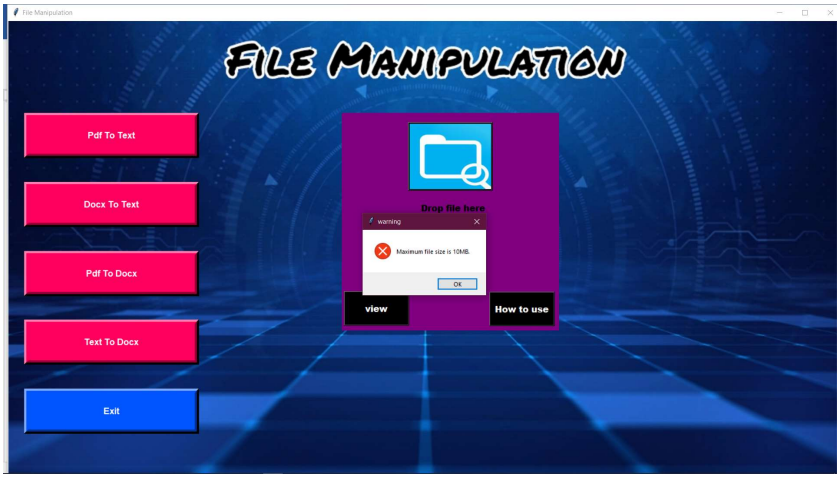
Invalid Extension for saving File:



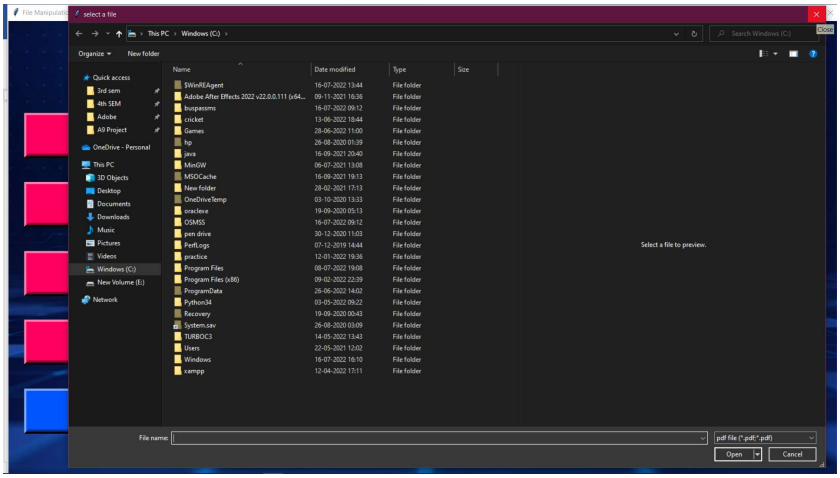


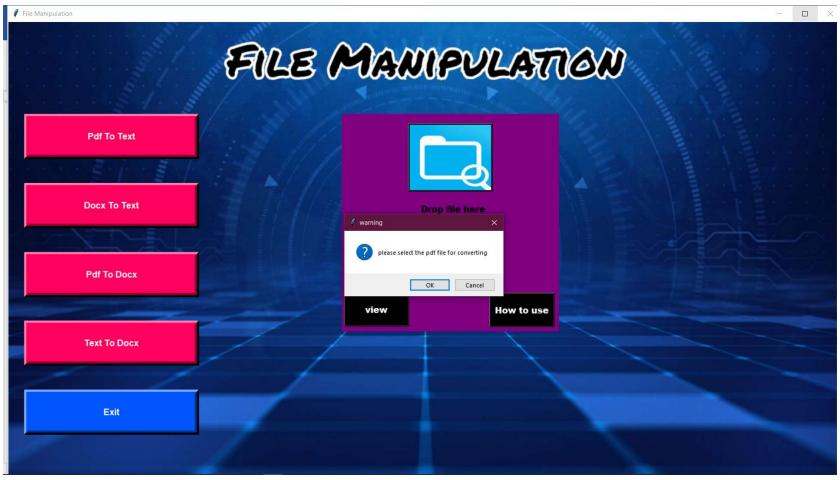
Invalid maximum size:



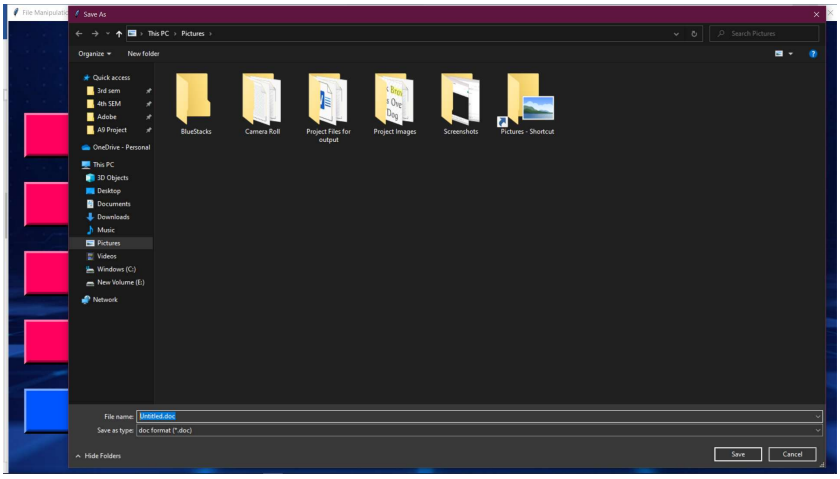


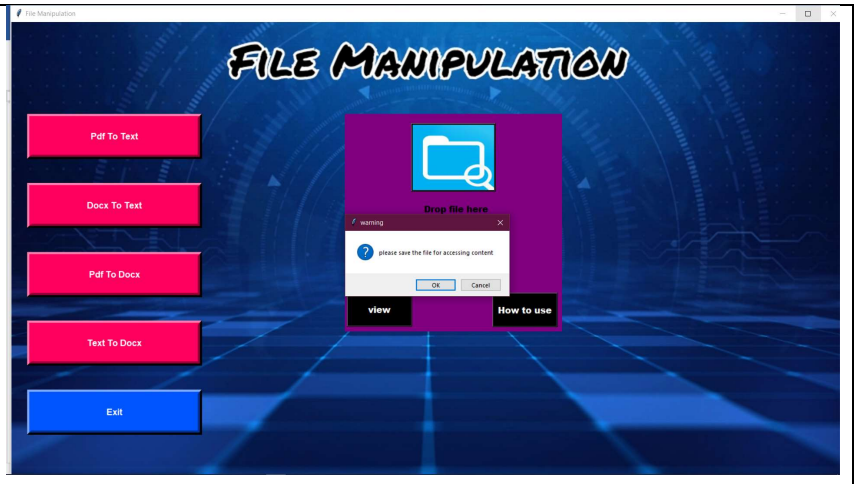
Invalid Selection:





Invalid saving:




File Manipulation
FILE MANIPULATION
Pdf To Text
Docx To Text
Pdf To Docx
Text To Docx
Exit
Drop file here
warning
please save the file for accessing content
OK
Cancel
view
How to use