**CHAPTER 1**

**INTRODUCTION**

Mobile phones have become an integral part of our lives, and they often contain critical evidence in criminal cases. Think of text messages, call logs, GPS data, and much more. But the problem is that extracting this data from mobile devices can be really tough. It's like trying to unlock a puzzle, and the methods used in the past aren't always up to the task, especially as mobile technology keeps evolving. Over the years, investigators have had to adapt to the changing world of technology. We've come a long way from the early days of personal computers to the smartphones we use today. Along this journey, digital forensics has played an increasingly vital role in solving crimes. Think of cases where text messages or GPS data on a suspect's phone helped crack the case; that's the power of digital evidence.

The "Digital Forensic Tool" is significant because it can make the process of collecting and analyzing data from mobile devices faster and more reliable. This can be a game-changer for law enforcement agencies. It means investigations can move more swiftly, evidence can be gathered more effectively, and it can ultimately lead to more successful prosecutions, which helps keep our communities safe. So, why do we need this tool? Well, it's because mobile devices are getting more complex. They come in all shapes and sizes, with various operating systems and security measures. This complexity can slow down investigations and even result in vital data being lost. This project is here to fill that need for a powerful and up-to-date tool that can handle these challenges head-on. Imagine a specialized computer program designed to help detectives and investigators solve crimes more effectively. Therefore, this project, known as "A Digital Forensic Tool for Extracting User Activity from Mobile Devices in Crime Scene Investigation Applications," is aimed at making the job of law enforcement agencies easier. It does this by helping them retrieve crucial information from mobile devices, which can be incredibly valuable in solving crimes.

**Problem Definition**

To put it simply, the problem this project is tackling is the difficulty investigators face when trying to extract user activity data from mobile devices. This task involves dealing with a wide range of devices, data formats, and encryption methods. The goal of this project is to create a user-friendly and comprehensive digital forensic tool that can efficiently handle these challenges. It's like giving detectives a high-tech magnifying glass to examine the digital fingerprints left behind on mobile devices.

**SOURCE CODE**

from tkinter import messagebox

from tkinter import \*

from tkinter.filedialog import askopenfilename

from tkinter import simpledialog

import tkinter

import numpy as np

from tkinter import filedialog

from bs4 import BeautifulSoup

import datetime

import pathlib

main = tkinter.Tk()

main.title("A Digital Forensic Tool for Extracting User Activity from Mobile Devices")

main.geometry("1300x1200")

global filename

global testData

global content

def upload():

global filename

filename = filedialog.askopenfilename(initialdir = "MobileData")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,'Selected file loaded\n')

def extractData():

global content

global testData

text.delete('1.0', END)

with open(filename, 'rb') as f:

content = f.read().decode("utf-16")

f.close()

soup = BeautifulSoup(str(content), "html.parser")

testData = soup.text

text.insert(END,content)

def forensicsActivity():

global testData

text.delete('1.0', END)

arr = testData.split("\n")

text.insert(END,"Total lines found in file : "+str(len(arr))+"\n")

fname = pathlib.Path(filename)

modify\_time = datetime.datetime.fromtimestamp(fname.stat().st\_mtime)

create\_time = datetime.datetime.fromtimestamp(fname.stat().st\_ctime)

size = fname.stat().st\_size / 1000

text.insert(END,"File Created Date : "+str(create\_time)+"\n")

text.insert(END,"File Last Modified Date : "+str(modify\_time)+"\n")

text.insert(END,"File size in KB : "+str(size))

def filterData():

global testData

text.delete('1.0', END)

arr = testData.split("\n")

values = ''

for i in range(len(arr)):

if 'PM)' in arr[i] or 'AM)' in arr[i]:

values+=arr[i]+"\n"

text.insert(END,values)

def close():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='A Digital Forensic Tool for Extracting User Activity from Mobile Devices')

title.config(bg='dark goldenrod', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

upload = Button(main, text="Upload Mobile Data", command=upload)

upload.place(x=700,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='DarkOrange1', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=700,y=150)

featureextractionButton = Button(main, text="Extract Data", command=extractData)

featureextractionButton.place(x=700,y=200)

featureextractionButton.config(font=font1)

featureselectionButton = Button(main, text="Forensic Activity", command=forensicsActivity)

featureselectionButton.place(x=700,y=250)

featureselectionButton.config(font=font1)

proposeButton = Button(main, text="Filter Data", command=filterData)

proposeButton.place(x=700,y=300)

proposeButton.config(font=font1)

existingButton = Button(main, text="Exit", command=close)

existingButton.place(x=700,y=350)

existingButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=80)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=100)

text.config(font=font1)

main.config(bg='Lavender')

main.mainloop()

**RESULTS AND DISCUSSION**

This project is a graphical user interface (GUI) application built using the Tkinter library. The application serves as a digital forensic tool for extracting user activity from mobile devices. The application's main functionality includes uploading a mobile data file, extracting, and displaying its content, providing forensic information about the file, filtering specific data, and allowing the user to exit the application. It's a basic digital forensic tool's GUI frontend. Below are the step-by-step details:

* Importing Required Libraries:

The code imports several libraries, including Tkinter for creating the GUI, numpy for numerical operations, BeautifulSoup for parsing HTML content, datetime for working with dates and times, and pathlib for handling file paths.

* Creating the Main GUI Window:

It creates the main GUI window using tkinter.Tk().

Sets the window title and geometry (size).

* Global Variables:

The code defines global variables filename, testData, and content to store file-related data and extracted content.

* Functions:

upload(): This function is called when the "Upload Mobile Data" button is clicked. It opens a file dialog for the user to select a mobile data file. It updates the filename variable, displays the selected file's path, and clears the text box.

extractData(): This function is invoked when the "Extract Data" button is clicked. It reads the content of the selected file, decodes it as UTF-16, and stores it in the testData variable. It then inserts the content into the text box.

forensicsActivity(): This function is called when the "Forensic Activity" button is clicked. It provides some information about the selected file, including the number of lines, creation date, last modification date, and file size (in kilobytes). It uses the pathlib and datetime libraries for this purpose.

filterData(): This function is triggered when the "Filter Data" button is clicked. It filters the data in the testData variable to extract lines containing "PM)" or "AM)" and displays them in the text box.

close(): This function is executed when the "Exit" button is clicked. It destroys the main GUI window, effectively closing the application.

* GUI Elements:

The code creates various GUI elements, including labels, buttons, and a text box, and configures their appearance and behavior. These elements are placed at specific coordinates within the GUI window.

* Main Loop:

Finally, the main GUI loop is started using main.mainloop(). This loop keeps the GUI application running and responsive to user interactions.

To implement this project, we have designed following modules:

* Upload Mobile Data: This module is used to upload chat log HTML messages files to application.
* Extract Data: This module will extract HTML data from uploaded file and then display content of that file.
* Forensics Activity: This module will extract file size, file creation and modification date and number of lines in that file.
* Filter Data: This module will apply HTML parsers to remove HTML tags from chat logs and then display clean chat messages between users.

Figure 1 shows a visual representation of the Graphical User Interface (GUI) of the proposed application. It includes elements such as buttons, labels, text input fields, and other graphical components that make up the user interface. It provides an overview of what the application looks like before any data is uploaded or processed. Figure 2 is a visual representation of the chat log data in HTML format that is considered as the mobile device data to be uploaded into the GUI application. It shows the relevant information from the mobile device data. Figure 3 represents the GUI application interface after the user has successfully uploaded the chat log HTML data. It shows changes in the interface compared to Figure 1, indicating that data has been loaded into the application.

A screenshot of a computer

Description automatically generated

Figure 1: Illustration of proposed GUI application for extracting user activity from mobile devices data.

A screenshot of a computer

Description automatically generated

Figure 2: Displaying various chat log HTML messages as the data of mobile devices to upload it to the proposed GUI application.

Figure 4 illustrates the result of a user action, such as clicking a button labeled "extract data." It shows the content of the uploaded HTML file displayed within the application's interface. This content includes the HTML structure of the messages. Figure 5 displays the results of a forensic analysis conducted on the extracted data. It includes details such as the total number of lines found in the data, the creation date and last modified date of the uploaded file, and the size of the file in a graphical format. From Figure 5, the first line shows that the uploaded file contains total 113 lines and the file created and modified date and file size is 39.272 KB.

A screenshot of a computer

Description automatically generated

Figure 3: Illustration of GUI application after uploading the sample mobile data from chat log HTML messages.

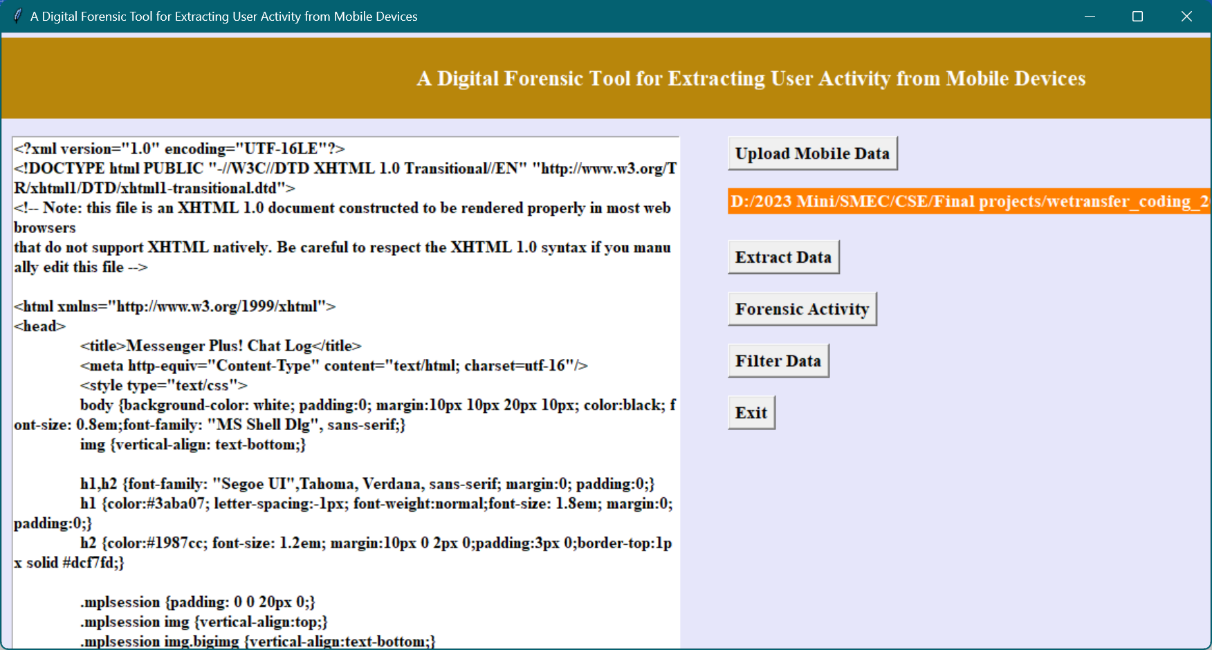


Figure 4: Displaying the uploaded file content in HTML format by clicking on extract data module.

A screenshot of a computer

Description automatically generated

Figure 5: Illustrating the forensic activity on extracted data module with the total lines found, file creation and last modified date and the size of the file.

Figure 6 represents the output within the GUI application after a specific filtering process has been applied to the chat logs extracted data. Here's a detailed explanation of Figure 6:

* Chat Logs Display: This part of the figure displays the chat logs or chat messages from the extracted data, but with some cleaning or filtering applied to make the messages more understandable to the user.
* Filtered Content: The content displayed consists of chat messages that have been processed or filtered to remove irrelevant or noisy information. Filtering involves removing system messages, duplicates, or any other content that doesn't contribute to the user's understanding of the chat.
* Clean Formatting: The chat messages are presented in a clean and structured format, making it easy for the user to read and comprehend. This includes timestamps, user IDs, and the actual message text.
* User-Friendly Presentation: The primary aim of this figure is to present the chat data in a way that is user-friendly and facilitates a better understanding of the conversation. It involves arranging messages in chronological order, highlighting important elements, or using different colors or fonts for improved readability.

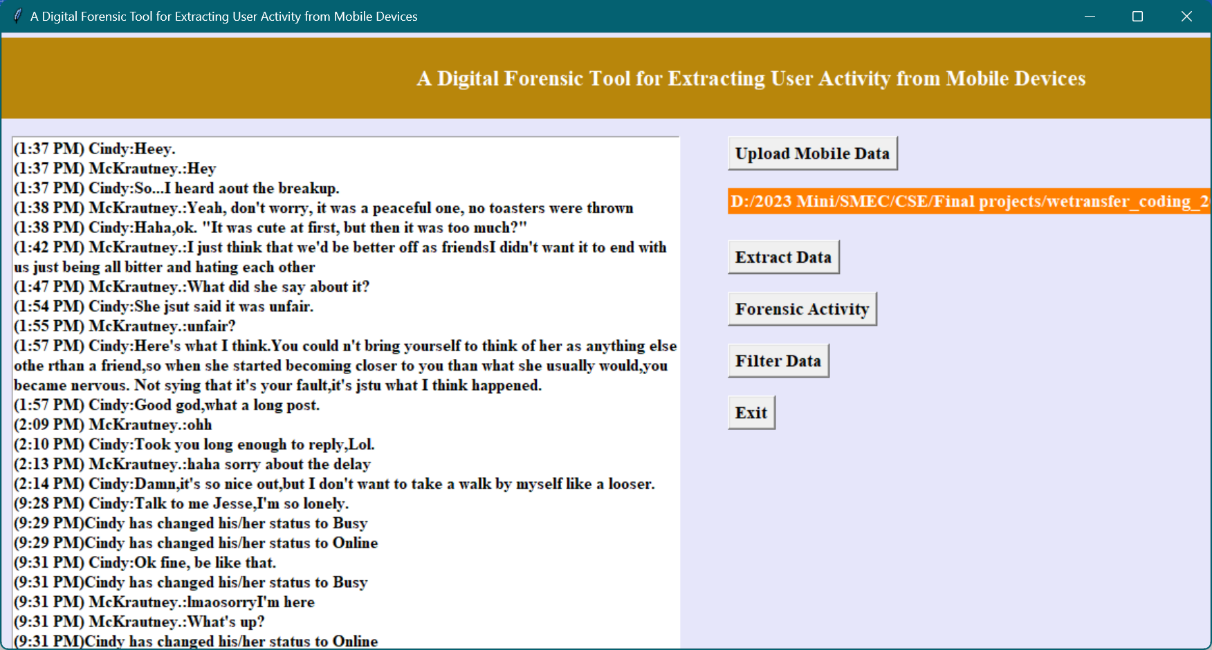


Figure 6: Chat logs extracted data after applying the filtering to clean chat messages for user understating.

From Figure 6, the chat messages are extracted from the HTML content and user can read above messages clearly. So, this proposed system has a clean chat message from HTML tags by applying crime scene investigation logger. Similarly, any file can be uploaded, and the messages can be extracted as discussed in above outcome.