APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 305-324
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

TwiVisual with Link and Timeline Analysis: A Forensics Visualization Tool for Twitter

Ng Jun Xiu[1], Nurul Hidayah Ab Rahman[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: hidayahar@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.017

Article Info

Received: 24 July 2024
Accepted: 18 june 2025
Available online: 30 June 2025

Keywords

Twitter Forensics, Forensic Visualization, Link Analysis. Timeline Analysis

Abstract

The substantial volume of Twitter data generation poses challenges for digital forensics. Due to the huge amount of Twitter data, Twitter platform occurs the increasing cybercrime rate has led investigators to perform extensive manual analysis on Twitter. Therefore, this study proposed TwiVisual, a forensic visualization tool which integrates link and timeline analysis to enhance the experience of analysing massive volume of Twitter data then provide comprehensive insights for digital forensic investigations. The modules of TwiVisual involves create case module, visualization analysis module and reporting module. Visualization analysis module involves 4 submodules which are file importation submodule, data tabulation submodule, link analysis submodule and timeline analysis submodule. Functional testing results confirm TwiVisual ability to create case, title and description display, adjust size of graph, save graph image and generate report. Forensic analysis testing was conducted to identify the functions of display network graph for link analysis and display line graph for timeline analysis. All actual outputs are similar with expected outputs. The user acceptance testing shows positive feedback to confirm the TwiVisual functions and performance. In conclusion, TwiVisual with link and timeline analysis has been successfully developed to offer more in-depth Twitter data analysis, enable investigators to better understand complex information, uncover relationships and timelines of events and reduce their workload.

1. Introduction

Social media's increasing usage and development caused many digital devices under criminal investigation. This makes forensic investigations of social media challenging and leads to significant issues of fast and accurate forensic analysis [1]. As an example, Twitter is the most widespread social network with over 313 million active users and more than 500 million daily posts on Twitter [2]. Therefore, various forensic artifacts are found on Twitter, such as hashtags, tweets, timelines, and like counts. The high-volume and high-velocity surge of Twitter artifacts generated at each second can be used for significant analytical and interpretation purposes. However, two critical issues the digital forensics domain faces are data complexity and volume. For data complexity, the relative difficulty of investigations is increasing due to increased networking, data storage, and complex consumers. For data volume, the information overload practitioners face results from endless growth in the storage capabilities of modern digital consumer devices [3].

The massive volume of Twitter data increases the difficulties of relevant artifacts investigation process and analysis. This will cause the process to be delayed or incomplete. For example, unstructured data could be time-consuming and increase the investigation cost for forensic examination tasks. As a result, there has been significant interest in investigating suitable methods to speed up digital forensic analysis tasks. Forensic visualization is one of the potential strategies. One of the ideas that explains how, what, and why organizations need to organize awkward, unordered data in a well-structured way is data visualization. Visualization can be focused, filtered, and provides high-level overviews of data that might help achieve the goals of reducing the relative size of the evidence and providing high level overviews [4].

Apart from that, Silva and Catarci [5] stated that time can be thought of as merely an ordinal dimension in a 2D or 3D visualization from a basic perspective. Therefore, link and timeline analysis are used as the approach in this project. Link analysis is the process of finding a connection or relationship between network nodes or users while timeline analysis is performed to obtain the processed information at a particular period. Both timeline analysis as well as link analysis are considered powerful instruments in an investigation. According to Henseler [6], combination of link and timeline analysis are proposed because it is more powerful to enable investigators to analyze the links in the relation graph in a chronological order which provide greater context and significance than merely limiting the relation graph or timeline to a specific time or set of entities.

Therefore, a forensic visualization tool for Twitter with link and timeline analysis which is called TwiVisual is proposed. As stated by Vikas [7], visualization is applied to go through data that is in enormous amounts to reduce the amount of time invested in analyzing it. The tool in this project was also used to streamline the analysis process and enable more efficient and effective analysis of Twitter data by using forensic visualization which integrates link and timeline analysis techniques. The tool provides comprehensive insights into Twitter data, offering graphical representations of tweets, user connections, and hashtags, enhancing data comprehension and aiding in effective communication of findings. With features like link and timeline analysis, users gain enhanced investigative depth, allowing them to track information flows and the chronological order of tweets, crucial for forensic investigations. In this project, the objectives are as follows:

i. To design a tool for forensics visualization of Twitter, called TwiVisual with link and timeline analysis,
ii. To develop TwiVisual with link and timeline analysis and
iii. To test TwiVisual in terms of tool's functionalities and user acceptance test.

2. Related Work

Section 2 discusses Twitter forensics, type of data visualization, forensic visualization, forensic analysis, and comparison with previous tools.

2.1 Twitter Forensics

With over a thousand tweets being made every second, it is impossible to exclude the likelihood of private, illegal text sharing [8]. Social media Twitters one area for disclosing accounting information in tweets, retweets, and other posts [9]. Twitter data comes in several forms, including user profile information and tweet messages. The latter is seen as dynamic, whereas the former is static. Tweets may contain links, photos, videos, text, or spam. The studies do not consider spam tweets and automatic tweets engines into account as they can because they will affect the accuracy and add bias to analysis results [10].

Boididou [11] said that the issues surrounding false news and misinformation online are becoming more and more serious. Popular social media sites like Facebook and Twitter are frequently used to post and extensively distribute false material. Therefore, the web-based application that supports visualizing and communicating the classification Tweet data is proposed to analyze a large dataset of tweets that shared debunked fake and confirmed real images and videos.

Meyer [12] found that breaking news by searching through the content shared on Twitter. There are several incidents which include fires, earthquakes, and attacks. Twitter connected to a forest fire that happened in France in 2009 is analyzed by Longueville [13]. Through the work of Longueville [13], the analysis of the text shared on Twitter was precise and in line with the fire incident. Bollen [14] uses GoogleProfile and OpinionFinder to analyze information from Twitter. The experiment's outcomes indicate that including public emotions can greatly enhance the precision of DIJA forecasts.

2.2 Type of Data Visualization

Although statistical graphics and data visualization are sometimes thought of as relatively recent discoveries in statistics, data visualization are essential components of business and data analytics. Data visualization is the process of presenting vast amounts of data in an approachable, clear, and uncomplicated manner. Data visualization is an effective technique for telling visually oriented people data by transforming data from multiple sources into tangible visuals (such as length, position, shape, color, and so forth) [15].

The use of computer-supported visual data representation is known as data visualization. Computer graphics are used in data visualization to display relationships, patterns, and trends among data items. Using easy pull-down menus and mouse clicks, it can create scatter plots, pie charts, bar charts, and other kinds of data visualizations [16].

There are several types of data visualization. Through the research of Shadare [16], the relationship between the components is displayed in a line graph. It can be applied to compare variations across time. It can be applied to compare variations across time. Line graph is one of the basic techniques to make the data more appealing and visualized. This is because it shows the relationship between two patterns. It is also very effective to compare several values at the same time interval [17].

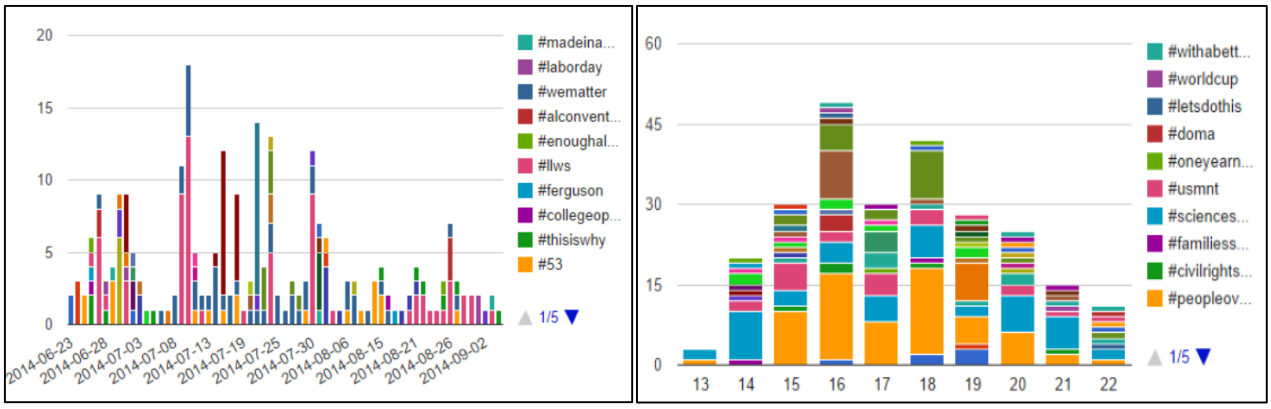
The network graph is used to illustrate the relationship between triggers and actions. It also demonstrates how closely they are connected. Analysis greatly aids in the visualization of massive data. For a variety of big data applications, achieving the best results requires a close combination of analytics and visualization [17].

Essential components of the data are represented by spatial variables including position, size, and shape. The original dataset should be projected onto a screen, transformed, and undergo data reduction via a visualization system. A discursive tool for disseminating statistical and frequent numerical data is data visualizations. The graphical representations of data that are mainly, but not exclusively, numerical are referred to as data visualizations.

2.3 Forensic Visualization

Visualizations may be used in digital forensics to support forensic analysis and interpretation and to conduct rapid and cost-effective investigations. Sern [18] proposed that visualization is a well-developed field that has been included in the field of digital forensics. The studies on the forensic visualization of evidentiary artefacts from instant messaging apps are still lacking, nevertheless. Adopting visualization is urgently needed to support forensic investigation tasks for instant messaging apps, as the use of developing technologies grows.

Forensic visualization provides vital answers in pressing concerns and conveying results. Based on measurable data and scientific analysis, forensic visualization is used to clearly and succinctly depict how an occurrence occurred [18]. Through the research of Pirzada [4], it has been highlighted that forensic visualization is a practice for digital forensic investigators to perform cross-analysis using various forensic software, but there is a lack of advanced visualization approaches to facilitate evidence analysis. Fig.1 shows the samples for forensic visualization. Fig.1 (a) uses hashtags and timelines to analyze changes in user activity while Fig.1 (b) uses different hashtags the user tweeted about over some time interval to provide a nice visual way of seeing what the user is interested in and even detect what sort of topics he tends to combine.



(a) **(b)**

Fig.1 *Samples for the forensic visualization (a) User-hashtag distribution; (b) User-hashtag distribution in different time of day*

2.4 Forensic Analysis

Forensic analysis is one of the phases in digital forensics, which is undertaken after evidence collection and examination [4]. The forensic analysis process included collecting information from the available evidence, formulating and testing theories to explain the evidence, and finally gaining a greater knowledge of a specific piece of evidence or the crime [19]. Link analysis and timeline analysis are the two analysis approaches that can be applied in performing digital forensics analysis.

2.4.1 Link Analysis

Through link analysis, researchers compile pertinent data from unprocessed data. After processing, the compiled data is shown in an organized manner. Many link analysis studies have been undertaken in the last few years. Link analysis entails examining and combining data from disparate sources, scrutinizing police records, and refining findings using specialized expertise. A few links analysis tools further offer interactive features for manually visualizing criminal network topologies. The relationships between the items in these systems are frequently entered by hand by the users [20].

Besides, link analysis is based on digital forensic artifacts relies on a much richer set of data. Modern digital forensic tools have a feature that performs link analysis to assist forensic examiners with their investigations. Link analysis concept to files and operating system artifacts, helping a forensic examiner to visualize relationships within artifacts and across evidence sources, such as, computers, mobile devices, and even cloud-based accounts [6].

2.4.2 Timeline Analysis

The timeline analysis technique is a chronological analysis of incidents to display all occurrences in a chronological sequence [4]. Forensic procedures are needed to successfully retrieve these data, including the text messages' substance and the sender's location. Subsequently, the data is visualized to offer information on the participants, time details, activity tracking, and timeline alignment. A timeline is made by first capturing a series of events and then using that visualization to make a graph where the time axis is placed at the start or end of each event. This technique is known as timeline visualization [1].

The prevalent method for visualizing timelines involves employing a horizontal axis where time unfolds from left to right. Events are then positioned along this axis based on their respective timestamps [21]. The analysis method of timelines aids in organizing a sequence of events within a chronological dataset, making it easier to identify temporal relationships. This approach alleviates the burden on analysts to memorize events. Typically, timeline visualizations utilize rectangular graphs or charts to depict event times, incorporating horizontal lines to represent intervals [1].

2.5 Tools Comparison

There are three existing tools shown in this research. The tools include Autospy, TweetViz and Chatvisualizer. These forensic visualization tools are useful for forensic investigators and analysts to easier handle the data and improve the efficiency of their job. The features extracted in the studies stated in Table 1. There are 7 points that are compared which involve data source, field that focus on, link analysis creation, timeline analysis creation, operating system, advantages, and disadvantages. For Autospy, the tool developed for examination and extraction of digital evidence for investigative purposes, Disk images and logical files as its data sources, has timeline analysis only, operating system on Window, Linux and MacOS and user-friendly interface.

However, Autospy is too complex because its operation may require a high level of expertise. Next, TweetViz is developed for analysis of Tweet word pairs and hashtags, Twitter data as its data source, has timeline analysis only, operating system on platform and effective for analyzing Twitter data. However, TweetViz can only choose a smaller number of topics because the number of topics must be predefined. Furthermore, TweetViz has not been developed to address forensic analysis needs. After that, Chatvisualize is developed for generating data visualizations of WhatsApp chats, Whatsapp data as the data source, has both link and timeline analysis, operating system on Window, Linux, MacOS and simple analysis for WhatsApp chat data. However, Chatvisualize is unable to handle real-time data effectively.

Therefore, TwiVisual is proposed in this study. Twitter data as its data source, has both link and timeline analysis, operating system on platform and provide easy and simple analysis of Twitter data through visualization. However, TweetViz is unable capable of handling real-time data effectively.

Table 1 *Comparison between 3 previous tools and the proposed tool*

Features	Autospy [22]	TweetViz [23]	Chatvisualizer [24]	TwiVisual
Data Source	Disk images and logical files	Twitter data	WhatsApp chats	Twitter data
Main Function	Examination and extraction of digital evidence for investigative purposes.	Analysis of Tweet word pairs and hashtags.	Generating data visualizations of WhatsApp chats.	Visualization and analysis of Twitter data.

Table 1 *(cont.)*

Features	Autospy [22]	TweetViz [23]	Chatvisualizer [24]	TwiVisual
Link analysis creation	No	No	Yes	Yes
Timeline analysis creation	Yes	Yes	Yes	Yes
Operating System	Window, Linux, MacOS	Window	Window, Linux, MacOS	Window
Advantages	User-friendly interface.	Effective for analyzing Twitter data.	Simple analysis for WhatsApp chat data.	Provide easy and simple analysis of Twitter data through visualization.
Disadvantages	Too complex because its operation may require a high level of expertise.	Smaller number of topics because the number of topics must be predefined.	The number of messages that can be sent at one time.	Unable to handle real-time data effectively.

3. Methodology

TwiVisual tool was developed using object-oriented software development model to visualize, organize, document, and understand complex relationships to produce well-designed tool [25]. In this study, there are 6 phases applied in object-oriented software development model which are requirement elicitation phase, analysis phase, design phase, implementation phase and requirement testing phase. The activities and expected outcomes of each phase of object-oriented software development model are presented in Table 2.

Table 2 *Object-oriented software development model activity and outcome*

Phase	Activity	Outcome
Requirement Elicitation	Enable the examination of existing applications and documentation to identify reusable concepts and components, comprehend domain knowledge, and acquire early requirements.	• Functional requirements • Non-functional requirements • Hardware requirements • Software requirements
Analysis	Discuss requirements based on use case diagrams and employ UML for specification and visualization.	• Class diagram • Activity diagram • Use-case diagram • Sequence diagram
Design	Focus on interface and algorithm design and emphasis the usability and functionality.	• Interface design • Algorithms design • System architecture
Implementation	Implementing an object-oriented design generally involves using a standard object-oriented programming language (OOPL) such as Python.	• Interface Outcomes • Code segments
Requirement Testing	Perform functional, forensic analysis, and user acceptance testing to ensure the program serves its purpose.	• Functional testing • Forensic analysis testing • User acceptance testing

3.1 Object-oriented Requirement Elicitation

In the Requirement Elicitation phase, there are 4 outcomes displayed which are functional requirements, non-functional requirements, hardware requirements and software requirements.

Table 3 *Functional requirements of TwiVisual*

Functional Requirements	Description
Import data	The tool shall be allowed to import the CSV files.
Analyze and visualize data	The tool shall be allowed to select the data which want to visualize.
	The tool shall be allowed to choose the patterns of data visualization.
Generate report	The tool shall be allo wed to generate the visualization report in PDF document.

Table 4 *Non-functional requirements of TwiVisual*

Functional Requirements	Description
Operational	The tool should be able to run on the windows operating system.
Performance	The tool should be able to perform efficiently and responsively even with large datasets.
	The tool should be supporting multiple users at the same time and perform without failure for a specific amount of time.
Security	Any personal data of the Twitter dataset should be critically considered and prevent data leakage.
	The tool shall perform error handling if the result is invalid or incorrect.
Cultural and Political	The contents should be presented in English which is the language most used on the internet.
	The tool should be used by digital forensics experts who have 3 years' studying experience.
	The tool should be used by the user who has already granted access from the owner
Usability	The tool should provide a user-friendly interface as users can use the tool easily and with minimal training.

Table 5 *Hardware requirements of TwiVisual*

Hardware Requirement	Description
Hardware Device	Computer
Processor	Multi-core processor
RAM	8 GB RAM (Minimum)
Storage	Sufficient storage space for data storage and analysis
Graphic Card	Nvidia RTX series (Recommended)
Network	High-speed internet connection for data access

Table 6 *Software requirements of TwiVisual*

Software Requirement	Description
Operating System	Windows 10 (Minimum)
Platform	Visual Studio Code

3.2 Object-oriented Analysis

There are four outcomes displayed in analysis phase which are class diagram, activity diagram, use-case diagram and sequence diagram.

Applied Information Technology and Computer Science Vol. 6 No.1 (2025) p. 305-324

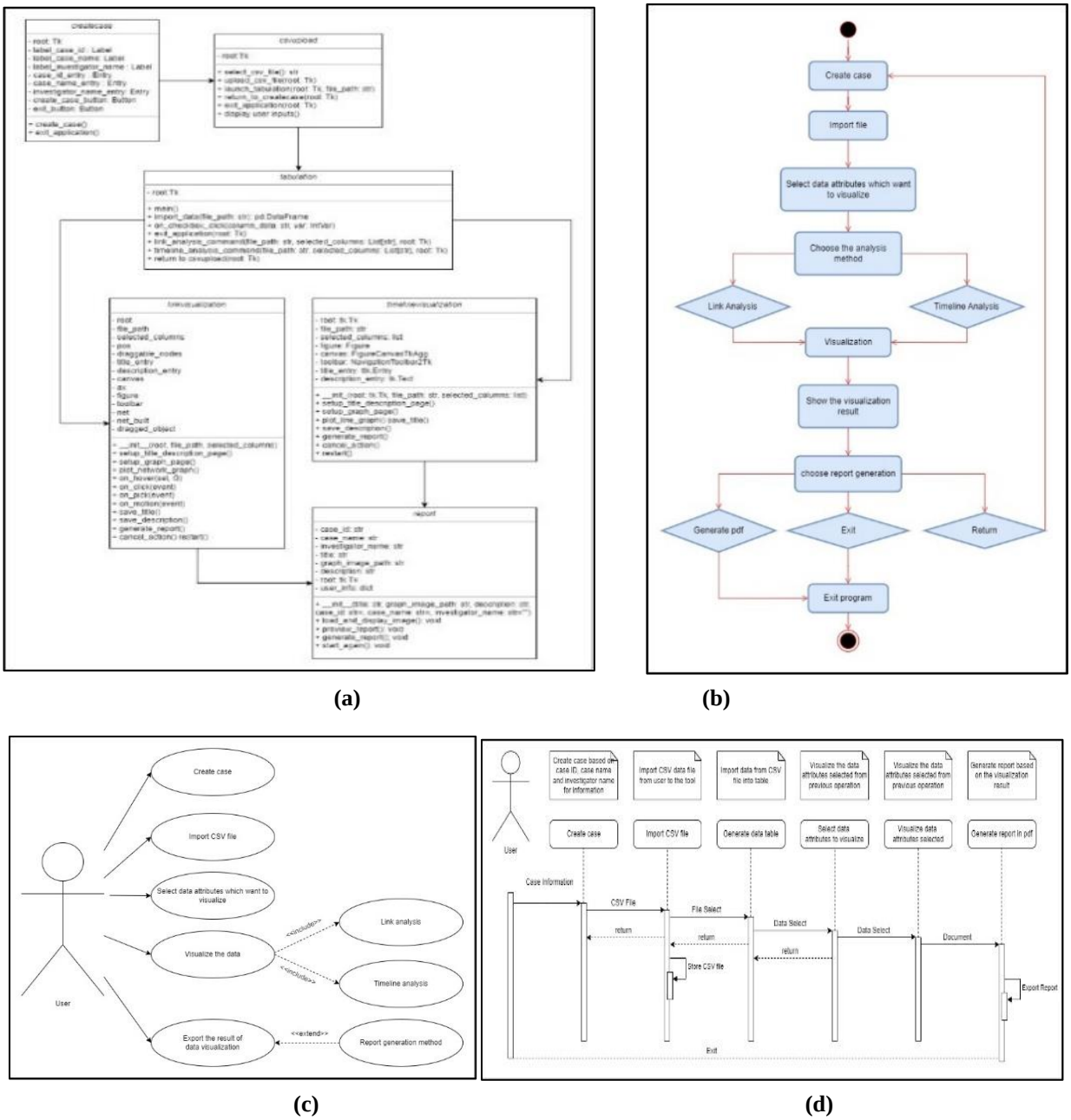


Fig.2 *UML Diagrams of TwiVisual (a) Class Diagram; (b) Activity Diagram; (c) Use-case Diagram; (d) Sequence Diagram*

Fig.2 (a) represents 6 classes used in TwiVisual. The class of createcase is used to create a case with case ID, case name and investigator name. Csvupload class is used to upload the selected CSV file and display the case information which involves case ID, case name and investigator name. Besides, the class of tabulation is used to tabulate the selected attribute columns to undergo the visualization analysis. In addition, linkvisualization and timelienvisualization class are used to input and save the title and description, plotting graphs, manage the events such as hover, click, pick and motion and generate report respectively. Finally, report class is used to generate forensic analysis reports based on the provided details such as case information, title, description and the graph image.

Fig.2 (b) presents the tool starts to bring users to create a case then prompt them to import their CSV file. Users can import a CSV file that they want to visualize. After that, the user can select the dataset attributes which want to visualize. The tool will provide two analysis method options which are link analysis and timeline analysis. After that, the tool will undergo the visualization process and comes out with data visualization results. Next, the user can choose whether to generate the visualization result in PDF format. After the user generates the report, he/she can choose to exit the program.

Fig.2 (c) shows the use-case diagram of TwiVisual. There are 5 operations provided to the users of the proposed tool which are create case, import CSV files, select data attributes which want to visualize, visualize the

data, and export the result of data visualization. Firstly, the tool should provide an input method to enable user to import the CSV file. Moreover, users can select the data that they want to visualize then visualize the data selected in link and timeline analysis. Lastly, the result of data visualization should be able to generate successfully in PDF format.

Fig.2 (d) discusses the sequence diagram of TwiVisual. Firstly, the tool starts with creating a case from user to the tool. Next, the tool importing a CSV file, the user needs to browse the file. In the analysis process, the data attributes will be selected and made the analysis with the operation of link or timeline analysis. If the data is supported with visualization, the user can analyse the data visualization in the visualization process. In the visualization process, the data selected needs to be received and visualized. Lastly, the result of visualization can be exported to generate the report. Then, the user can exit the tool or continue to import a new CSV file.

3.3 Object-oriented Design

There are three outcomes displayed in design phases which are interface design, algorithms design and system architecture.

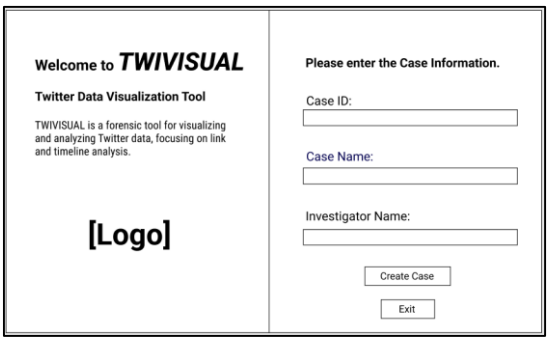
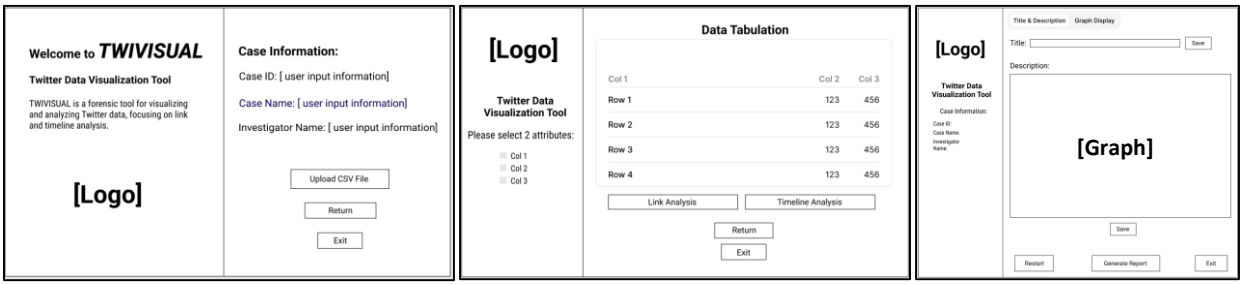


Fig.3 *Interface design of TwiVisual's create case module*



(a) **(b)** **(c)**

Fig.4 *Interface design of TwiVisual's visualization analysis modules (a) Data Tabulation Submodule; (b) Link Analysis Submodule; (c) Timeline Analysis Submodule*

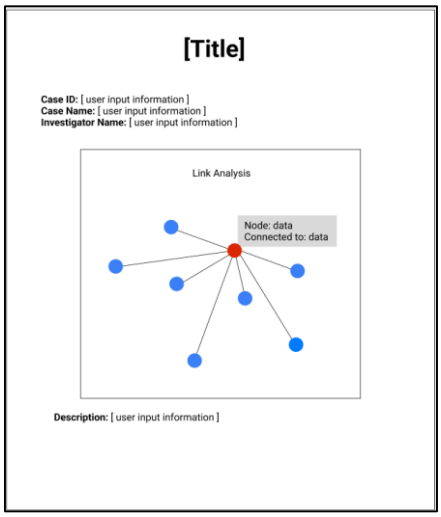


Fig.5 *Interface design of TwiVisual's reporting module*

For create case module which shows as Fig.3, users can insert the case ID, case name and investigator name. The case ID is unique for each investigation case. The case name is responsible for recording what is the case against while investigator name is record who analyze the case. The interface displays case information for user analysis, allows CSV file uploads of Twitter data, includes a return button for correcting case information, and provides an exit button to terminate the tool.

The visualization analysis module (Fig.4), data tabulation submodule represents that left panel displays its attributes and prompts the user to choose two for link or timeline analysis while the right panel sets up a corresponding table. After selecting the analysis method, a confirmation message is displayed to allow users to click yes to proceed or no to cancel. Moreover, link analysis submodule shows that title and description tab allow users to save titles and descriptions with confirmation messages while graph tab allow users to view, navigate, and save the network graph with the provided toolbar. Furthermore, timeline analysis submodule shows that title and description tab allow users to save titles and descriptions with confirmation messages while graph tab allow users to view, navigate, and save the line graph with the provided toolbar. In the reporting module (Fig.5), users select a save folder, ensuring a unique filename with a timestamp, and the reports for link and timeline analysis include the title, case information, graph, and description after clicking the generate report button.

Table 7 *Algorithms design of TwiVisual*

Algorithms Design	Description
Link analysis: 1. BEGIN 2. Load user information from a JSON file. 3. Create labels for case information (case ID, case name, investigator name) using loaded user information. 4. Define a class LinkVisualizationApp: 5. Initialize the application with root window, file path, and selected columns. 6. Set up the main window with tabs for Title & Description and Graph Display. 7. Setup title and description page. 8. Save the title and description. 9. Setup graph page. 10. Plot the network graph: 11. Try to read the selected CSV file with different encodings. 12. If data is loaded successfully: 13. Add edges to the graph based on selected columns 14. Exit the loop 15. Handle column not found. 16. Show network graph. 17. END	The algorithm design for link analysis, which begins by loading user information from a JSON file to create labels for case details like case ID, case name, and investigator name. Within the LinkVisualizationApp class, the application is initialized with key parameters such as the root window, file path, and selected columns. The main window features tabs for inputting a title and description and for displaying the graph. After setting up and saving the title and description, the module configures the graph page by attempting to plot a network graph from the selected CSV file, trying different encodings until the data is successfully loaded. Upon retrieving the data, edges are added to the graph based on the selected columns, with error handling for missing columns, culminating in the display of the network graph.
Timeline analysis: 1. BEGIN 2. Load user information from a JSON file. 3. Create labels for case information (case ID, case name, investigator name) using loaded user information. 4. Create the main application class (TimelineVisualizationApp): 5. Initialize the class with root window, file path, and selected columns. 6. Create a notebook with two tabs: Title & Description and Graph Display. 7. Setup title and description page. 8. Save the title and description. 9. Setup graph page. 10. Plot the line graph: 11. Try to read the selected CSV file with different encodings. 12. If data is loaded successfully:	The algorithm designed for timeline analysis, starting with loading user information from a JSON file to create labels for case ID, case name, and investigator name. The core component, the TimelineVisualizationApp class, initializes with a root window, file path, and selected columns. The application includes a notebook with two tabs: one for entering a title and description, and another for displaying the graph. After setting up and saving the title and description, the graph page attempts to read a selected CSV file using various encodings. Upon successful data loading, it clears previous plots, sets labels and title, and plots the data with hover effects. The plot can be saved as an image. The module includes error handling for issues like missing columns and other potential problems, ensuring a reliable and

Table 7 *(cont.)*

Algorithms Design	Description
13. Clear the previous plot and create a new one. 14. Set labels and title for the plot. 15. Plot the data for selected columns and enable hover effects. 16. Save the plot as an image. 17. Handle column not found. 18. Show line graph. END	robust visualization experience.

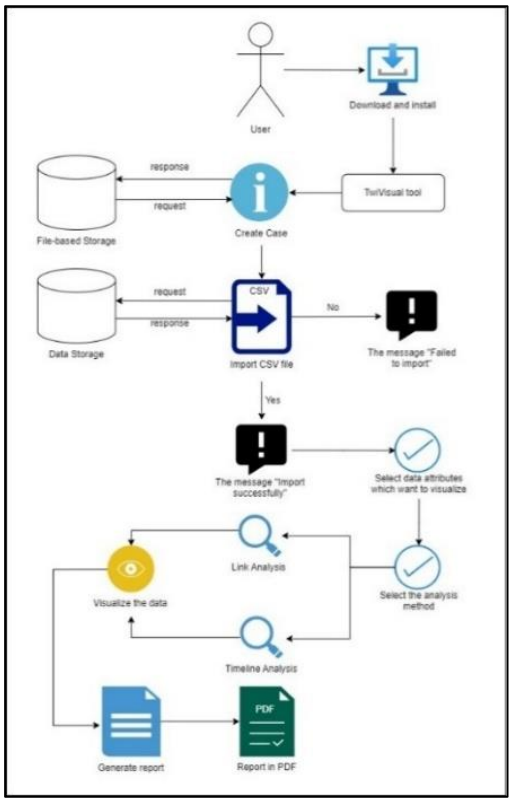


Fig.6 *System architecture of TwiVisual*

From Fig. 6, we can see that users can a create case such as case ID, case name and investigator name and save the case information into the file-based storage temporarily after the download and installation of TwiVisual tool. Next, they can import the CSV file that they choose from directory. After importing the file, users can select the data attributes which want to visualize then undergo the data visualization. Lastly, the users can generate the visualization report in pdf format.

3.4 Object-oriented Implementation

TwiVisual is developed by using Python programming language. The tool will include creating the input and output processes and giving a button a function.

3.5 Object-oriented Testing

There are 3 testing plans used in this phase which are system functional testing, forensic analysis testing and user acceptance testing that will be carried out. System functional testing and forensic analysis testing will test whether all modules integrate seamlessly, user-friendly interface, measure response time of the tool and the performance of the tool under condition of the tool handles large dataset load. For the user acceptance testing, the tool will be evaluated by the 20 potential user of TwiVisual to obtain suggestions and feedback for improving the tool.

4. Result and Discussion

This section discusses the implementation and testing of TwiVisual. Firstly, TwiVisual will be going perform visualization through link analysis and timeline analysis. The implementation and testing of the tool will be discussed.

4.1 Implementation of TwiVisual

This section discusses and explains how TwiVisual's modules are coded and implemented. TwiVisual contains several modules which are create case modules, visualization analysis module and reporting module. Visualization analysis module involves 5 submodules which are file importation submodule, data tabulation submodule, title & description submodule, link analysis submodule and timeline analysis submodule.

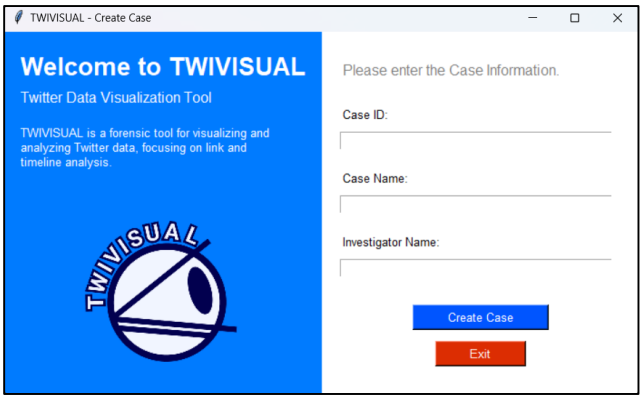


Fig. 7 *Create case interfaces design of TwiVisual*

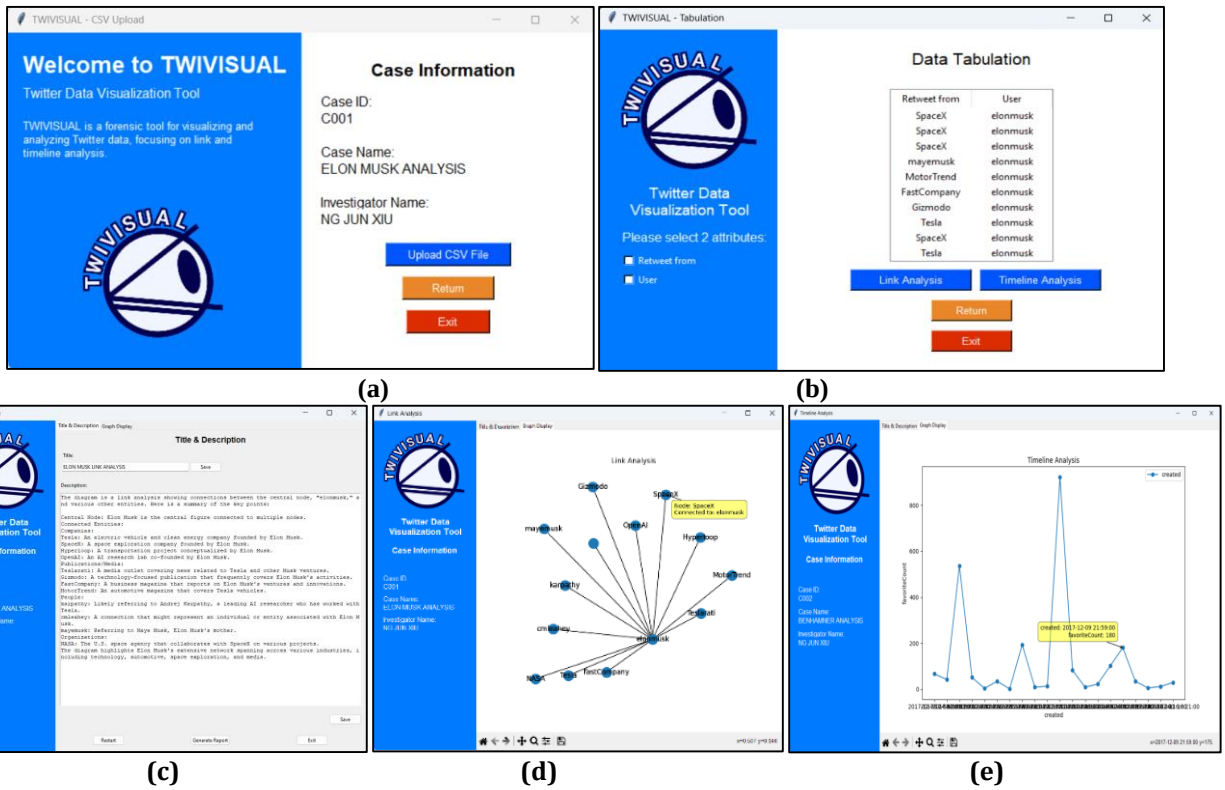
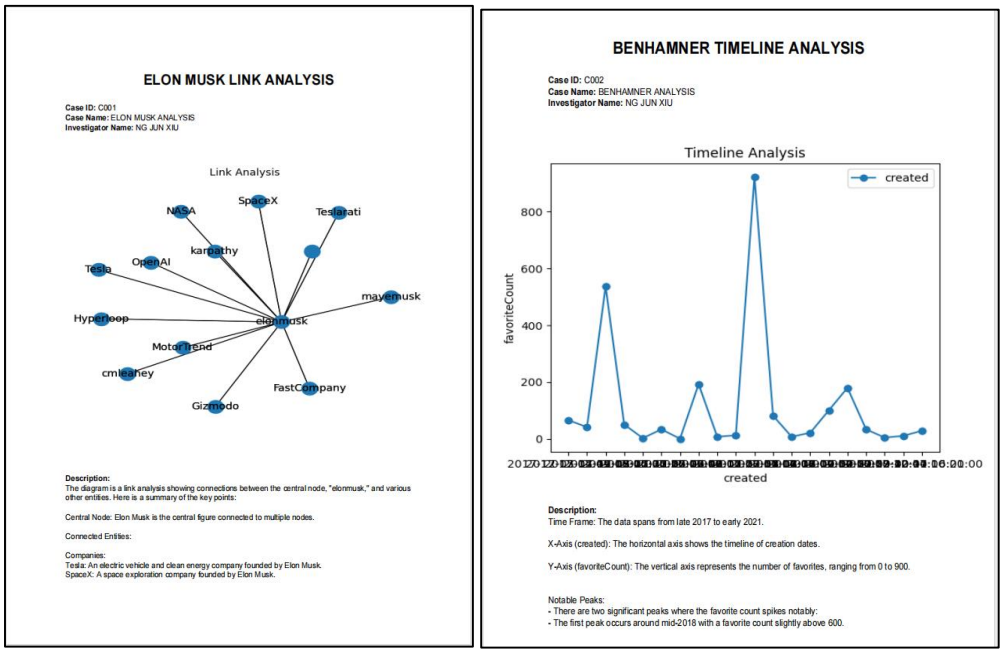


Fig. 8 *Visualization Analysis interface design of TwiVisual (a) file importation interface; (b) Data tabulation interface; (c) Title & description interface (d) Link visualization analysis interface; (e) Timeline visualization analysis interface.*



(a) **(b)**

Fig. 9 *Reporting interfaces design of TwiVisual (a) Link analysis report; (b) Timeline analysis report*

Fig. 7 shows the create case interfaces. In this interface, the case ID, case name and investigator name are required for the user. The case information will be printed out in the report to record the case in PDF format. Before the information printed out, the case information will be displayed at this interface to let users know the case information they want to analyze.

Fig. 8(a) shows the upload CSV file button is provided to enable user to upload the Twitter data file in CSV format. For return button, user can return to the previous page to refill again if fill any wrong case information. This is to prevent mistakes in the case information and make the tool more interactive. Exit button is provided to user to terminate the tool.

The data tabulation interfaces are shown as Fig. 8(b). The left panel provides the selected file's attributes, then it will prompt user to choose any two attributes under the link analysis or timeline analysis. The table will be set up at the right panel and it based on the selected file. After that, link analysis and timeline analysis are provided to enable user to choose the analysis method.

After that, Fig. 8(c) represents the title and description tab and display tab of the link and timeline analysis module are designed for recording the title and description of the graph. This tab includes several buttons: save, restart, generate report, and exit. It also features a left panel displaying the current case information for user reference. The graph displays tab of the link and timeline analysis module, which is used to visualize the network graph and timeline graph. Users are allowed to see detailed information when hovering over each point. Additionally, a navigator toolbar enables users to zoom in, zoom out, and move the graph. The toolbar also provides an option to save the graph as an image.

Fig. 9(a) and Fig. 9(b) shows the reports of link analysis and timeline analysis respectively. The report contains title, case information, graph, and description for investigator to review. These interfaces are enabled more convenient for investigators to further print out their comments and discoveries.

Table 8 *Coding segments for link and timeline analysis*

Analysis	Coding Segments

Link Analysis	class LinkVisualizationApp:
	def __init__(self, root, file_path, selected_columns):
	self.root = root
	self.file_path = file_path
	self.selected_columns = selected_columns
	self.pos = None # Initialize pos attribute
	self.draggable_nodes = []

Table 8 *(cont.)*

Analysis	Coding Segments

```
    def setup_graph_page(self):
self.net = Network(notebook=True)
        self.net_built = False
        self.figure, self.ax = plt.subplots(figsize=(7, 6))
        self.canvas = FigureCanvasTkAgg(self.figure, master=self.graph_frame)
        self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        # Add NavigationToolbar2Tk for zooming and panning
        self.toolbar = NavigationToolbar2Tk(self.canvas, self.graph_frame)
        self.toolbar.update()
        self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
        self.plot_network_graph()

    def plot_network_graph(self):
        G = nx.DiGraph()

        encodings = ['utf-8', 'latin1', 'ISO-8859-1', 'cp1252', 'utf-16']

        for encoding in encodings:
            try:
                with open(self.file_path, newline='', encoding=encoding) as csvfile:
                    reader = csv.DictReader(csvfile)
                    for row in reader:
                        source = row[self.selected_columns[0]]
                        target = row[self.selected_columns[1]]
                        G.add_edge(source, target)
                # If successful, exit the loop
                break
            except FileNotFoundError:
                print("CSV file not found.")
                return
            except UnicodeDecodeError as e:
                print(f"Unicode decode error with encoding {encoding}: {e}")
                continue  # Try the next encoding
            except KeyError as e:
                print(f"Column not found: {e}")
                return
            except Exception as e:
                print(f"An error occurred: {e}")
                return
        else:
            # If none of the encodings work, print an error message
            print("Unable to decode the CSV file with any of the specified encodings.")

        self.ax.clear()  # Clear previous plot
        self.pos = nx.spring_layout(G)
        title = self.title_entry.get()  # Get the title from the entry field
        self.ax.set_title(title if title else "Link Analysis")
        nx.draw(G, self.pos, with_labels=True, arrows=False, ax=self.ax)

        # Enable hover effects for each node
        nodes = nx.draw_networkx_nodes(G, self.pos, ax=self.ax)
        edges = nx.draw_networkx_edges(G, self.pos, ax=self.ax, arrows=False)
        labels = nx.draw_networkx_labels(G, self.pos, ax=self.ax)
        cursor = mplcursors.cursor(nodes, hover=True)
        cursor.connect("add", lambda sel: self.on_hover(sel, G))
```

Table 8 *(cont.)*

Analysis	Coding Segments

```
        self.canvas.draw()  # Ensure the canvas is drawn
        # Save the graph as an image
        self.figure.savefig("link_graph.png")

    def on_hover(self, sel, G):
        node = list(G.nodes)[sel.index]
        connected_nodes = list(G.neighbors(node)) + [n for n in G.nodes if node in G[n]]
        connected_nodes_str = ', '.join(connected_nodes)
        sel.annotation.set_text(
            f'Node: {node}\nConnected to: {connected_nodes_str}' )

    def on_click(self, event):
        # Get the coordinates of the click event
        x, y = event.xdata, event.ydata

        if event.button == 1 and x is not None and y is not None:
            # Check if any node is clicked
            for node, pos in self.pos.items():
                if x - 0.05 < pos[0] < x + 0.05 and y - 0.05 < pos[1] < y + 0.05:
                    if self.dragged_node == node:
                        # If the clicked node is already selected, fix it in place
                        self.dragged_node = None
                        return
                    else:
                        # If a different node is clicked, update the dragged_node
                        self.dragged_node = node
                        return

        # If no node is clicked, deselect any selected node
        self.dragged_node = None

    def on_pick(self, event):
        # Store the picked object (node)
        self.dragged_object = event.artist

        # Toggle draggable state of the node
        if self.dragged_object in self.draggable_nodes:
            # If the node is draggable, make it non-draggable
            self.draggable_nodes.remove(self.dragged_object)
            self.dragged_object.set_picker(False)
        else:
            # If the node is non-draggable, make it draggable
            self.draggable_nodes.append(self.dragged_object)
            self.dragged_object.set_picker(True)

    def on_motion(self, event):
        if hasattr(self, 'dragged_object') and self.pos is not None:
            x, y = event.xdata, event.ydata
            if x is not None and y is not None:
                self.dragged_object.set_offsets([[x, y]])
                self.canvas.draw_idle()

                # Update the position of the node in the graph
                for node, pos in self.pos.items():
                    if (pos == self.dragged_object.get_offsets()[0]).all():
                        self.pos[node] = (x, y)
                self.canvas.draw_idle()
```

Table 8 *(Cont.)*

Analysis	Coding Segments

Timeline Analysis	class TimelineVisualizationApp:
	def __init__(self, root, file_path, selected_columns):
	self.root = root
	self.file_path = file_path
	self.selected_columns = selected_columns
	
	def setup_graph_page(self):
	self.figure = Figure(figsize=(6, 5))
	self.canvas = FigureCanvasTkAgg(self.figure, master=self.graph_frame)
	self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
	
	# Add NavigationToolbar2Tk for zooming and panning
	self.toolbar = NavigationToolbar2Tk(self.canvas, self.graph_frame)
	self.toolbar.update()
	self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
	self.plot_line_graph()
	
	def plot_line_graph(self):
	encodings = ['utf-8', 'iso-8859-1', 'windows-1252']
	data = None
	
	for encoding in encodings:
	try:
	data = pd.read_csv(self.file_path, encoding=encoding)
	print(f"Data loaded successfully with {encoding} encoding")
	break
	except UnicodeDecodeError:
	print(f"Failed to load data with {encoding} encoding")
	
	if data is None:
	print("Failed to load data with all attempted encodings")
	return
	
	def on_hover(col):
	def _on_hover(sel):
	# Update the hover text with the x and y values
	x, y = sel.target
	date_str = pd.to_datetime(data[col].iloc[int(x)]).strftime('%Y-%m-%d %H:%M:%S')
	
	sel.annotation.set_text(
	f'{col}: {date_str}\n{self.selected_columns[0]}: {int(y)}')
	return _on_hover
	
	try:
	# Clear previous plot
	self.figure.clear()
	ax = self.figure.add_subplot(111)
	ax.set_xlabel(self.selected_columns[1])
	ax.set_ylabel(self.selected_columns[0])
	title = self.title_entry.get()
	ax.set_title(title if title else "Timeline Analysis")
	
	# Plot the data for the selected columns
	y_col = self.selected_columns[0] # Assume the first selected column as the y-axis attribute

Table 8 *(cont.)*

Analysis	Coding Segments

```
                for col in self.selected_columns[1:]:
                    if col in data.columns:
                        line = ax.plot(data[col], data[y_col], marker='o', linestyle='-', label=col)
                        scatter = ax.scatter(data[col], data[y_col], color=line[0].get_color())  # Add
scatter for better hover

                        # Enable hover effects for each plot point
                        cursor = mplcursors.cursor(scatter, hover=True)
                        cursor.connect("add", on_hover(col))  # Pass col to on_hover function
                    else:
                        print(f"Column {col} not found in data")
                ax.legend()
                # Draw the plot
                self.canvas.draw()

                # Save the plot as an image file
                image_path = os.path.join(os.getcwd(), "timeline_graph.png")
                self.figure.savefig(image_path)
                print(f"Plot saved as image at {image_path}")

            except KeyError as e:
                print(f"Column not found: {e}")
            except Exception as e:
                print(f"An error occurred: {e}")
```

For the code segment of link analysis, the function of plot_network_graph attempts to read selected CSV file with each encoding in the list until successfully. csv.DictReader is used to read the rows from the selected CSV file. Several exceptions such as file not found, Unicode decoding errors and missing columns are handled. If an exception occurs, it prints an error message and continues or exits the loop. The interactive graph is generated to allow the user to hover over nodes to see their connections through mplcursors inside the on_hover function. On_click method is used to handle mouse clicks which are selecting or deselecting nodes based on click proximity. Apart from that, on_pick function is applied to toggles the draggable state of a node when it is picked. on_motion method is used to update the position of a node as it is dragged and reflect the changes in the graph's layout and redraw the canvas. Besides, users have the option to leave the program, resume it, create a comprehensive report with the graph, and save the title and description.

For the code segment of timeline analysis, the function of 'plot_line_graph' attempts to load selected CSV file with different encodings until successful. The code will check if the data is loaded. If data is loaded successfully, it breaks out of the loop. Otherwise, it prints an error message. The hover function uses the nested method to handle the hover event to display the date and corresponding y-value for the selected attribute columns. Mplcursors is used to enable hover effects. The 'plot_line_graph' function also adds scatter plots for each point on the line graph for better hover functionality. To increase accuracy of the line graph, any errors such as the column not found are caught during the plotting process.

4.2 TwiVisual Testing

Testing is the important phase to implement when the system is completed. There are three types of testing that TwiVisual undergoes which are system functionality testing, forensic analysis testing and user acceptance testing. Table 9 shows system functionality testing for creating case module of TwiVisual. For create case module, the test cases involve displaying information of TwiVisual, entering and displaying case information to create case, uploading CSV files and navigating with return and exit buttons. For analysis module, there are 12 test cases conducted. For example, attributes selection, input title and description, hover the graph points, adjust size of graph, save the graph and the function of buttons. The buttons in analysis module include link analysis button, timeline analysis button, save button, restart button and exit button. There are 4 test cases for reporting module which are generate the report, save the report, report name and report details. Generally, all the test cases are passed as expected.

Applied Information Technology and Computer Science Vol. 6 No.1 (2025) p. 305-324

Table 9 *Functionality testing of TwiVisual*

Modules	No	Test Cases	Expected Output	Actual Output
Create Case Module	1	Left Panel	Success in displaying the introduction of the TwiVisual.	Success in displaying the introduction of the TwiVisual.
	2	Enter Case ID, Case Name and Investigator Name	Enter successfully.	Enter successfully.
	3	Display Case ID, Case Name and Investigator Name	Display successful.	Display successful.
	4	Create Case	Create case successful.	Create case successful.
	5	Upload CSV file	Select and upload CSV file. If the file is uploaded, a success message will be displayed. Otherwise, an error message will be displayed.	Select and upload CSV file. If the file is uploaded, a success message will be displayed. Otherwise, an error message will be displayed.
	6	Return Button	Return to previous page successfully.	Return to previous page successfully.
	7	Exit Button	Determine whether exit TwiVisual. If yes, exit from TwiVisual. Otherwise, stay on the current page.	Determine whether exit TwiVisual. If yes, exit from TwiVisual. Otherwise, stay on the current page.
Analysis Module	1	Attributes Selection	Select or unselect the attributes without any errors.	Select or unselect the attributes without any errors.
	2	Link Analysis Button	Determine whether to perform link analysis. If yes, perform the link analysis process. Otherwise, stay at the current page.	Determine whether to perform link analysis. If yes, perform the link analysis process. Otherwise, stay at the current page.
	3	Timeline Analysis Button	Determine whether to perform timeline analysis. If yes, perform the timeline analysis process. Otherwise, stay on the current page.	Determine whether to perform timeline analysis. If yes, perform the timeline analysis process. Otherwise, stay on the current page.
	4	Enter Title and Description	Enter successfully.	Enter successfully.
	5	Hover Graph Points	The details of each point on the graph will be successfully displayed when hovering over each point of graph.	The details of each point on the graph will be successfully displayed when hovering over each point of graph.
	6	Adjust Size of Graph	Able to adjust the size of graph.	Able to adjust the size of graph.
	7	Save the Graph	Able to save the graph as an image.	Able to save the graph as an image.
	8	Save Button	Success in saving the title and description respectively.	Success in saving the title and description respectively.
	9	Restart Button	Restart the program successfully.	Restart the program successfully.
	10	Exit Button	Determine whether exit TwiVisual. If yes, exit from TwiVisual. Otherwise, stay on the current page.	Determine whether exit TwiVisual. If yes, exit from TwiVisual. Otherwise, stay on the current page.

Table 9 *(cont.)*

Modules	No	Test Cases	Expected Output	Actual Output
Reporting Module	1	Generate Report	Success in generating report as PDF file.	Success in generating report as PDF file.
	2	Save Report	Able to save the report as a PDF file to the selected directory.	Able to save the report as a PDF file to the selected directory.
	3	Report Name	Able to generate unique report filenames based on the current timestamp.	Able to generate unique report filenames based on the current timestamp.
	4	Report Details	Display the details correctly.	Display the details correctly.

Table 10 *Forensic analysis testing of TwiVisual*

No	Test Cases	Expected Output	Actual Output
1	Generate Attributes List	Able to automatically generate the attribute list from the selected CSV file.	Able to automatically generate the attribute list from the selected CSV file.
2	Display Table	The data from the selected file is displayed in a table without any errors.	The data from the selected file is displayed in a table without any errors.
3	Transfer Table	Able to transfer table into graph without any errors.	Able to transfer table into graph without any errors.
4	Analysis Method Selection	Able to trigger different types of graphs based on the analysis method that is selected.	Able to trigger different types of graphs based on the analysis method that is selected.
5	Graph Data Display	Able to display the data on the graphs correctly.	Able to display the data on the graphs correctly.
6	Data Details	Able to display the details of each data on the graph correctly.	Able to display the details of each data on the graph correctly.

Table 10 shows forensic analysis testing of TwiVisual. There are 6 test cases for forensic analysis testing which include generating attributes list, display table, transfer data, analysis method selection, graph data display and data details. The test case of attribute list generation is to test whether the attribute list at tabulation page of TwiVisual automatically generated from the selected CSV file. The Twitter data from the selected file displays as a table successfully while the table can transfer into graph without any errors. Furthermore, TwiVisual is also able to trigger different types of graphs based on the analysis method that is selected. For example, the network graph is generated for the link analysis method while line graph is generated for the timeline analysis method. TwiVisual tested whether the data and its details display on the graphs correctly. As a result, all actual output of the test cases is passed as expected.

Table 11 *User acceptance testing for user feedback*

No	Test Cases	1	2	3	4	5
1	Do you agree with the network graphs as the suitable visualization graph for link analysis of Twitter dataset?	0	0	0	4	16
2	Do you agree with the line graphs as the suitable visualization graph for timeline analysis of Twitter dataset?	0	0	0	2	18
3	Do you agree that it is easier to find the relationship between targeted user and its connections by using network graph?	0	0	1	2	17
4	Do you agree that it is easier to construct a timeline of events with a timestamp by using line graph?	0	0	0	3	17
5	Do you agree that TwiVisual generates meaningful results for forensic analysis?	0	0	0	3	17
6	Do you agree that TwiVisual helps users to save time and workloads in forensic analysis?	0	0	0	1	19

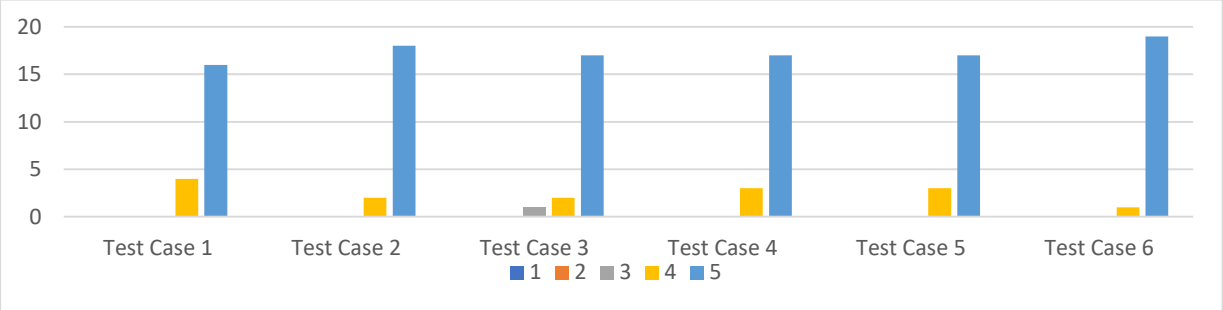


Fig. 10 *Result of test cases for user feedback*

Table 11 shows the user acceptance testing for user feedback to collect the opinion to improve TwiVisual tool while Fig.10 presents the result of test cases in user review. Six statements will be asked for overall review from the respondents. Number 1, 2, 3, 4 and 5 are used to represent the options which are disagree, somewhat disagree, neutral, somewhat agree and agree respectively. There are 4 respondents (20%) who somewhat agree with statements that do you agree with the network graphs as the suitable visualization graph for link analysis of Twitter dataset while 16 respondents (80%) agree with the statement. For the statement of do you agree with the line graphs as the suitable visualization graph for timeline analysis of Twitter dataset, 18 respondents (90%) agree, and 2 respondents (10%) somewhat agree with the statement. Besides, only one respondent (5%) stands neutral with the statement of it is easier to find the relationship between targeted user and its connections by using network graph. 17 respondents (10%) and 2 respondents (85%) agree and somewhat agree with the statement. The next statement is that it is easier to construct a timeline of events with a timestamp by using line graph and the result shows that 17 respondents (85%) and 3 respondents (15%) agree and somewhat agree with the statement respectively. Other than that, 17 respondents (85%) agree, and 3 respondents (15%) somewhat agree with the statement about TwiVisual generates meaningful results for forensic analysis. The most respondents (95%) agree TwiVisual helps users to save time and workloads in forensic analysis.

Overall, the results indicate that the users felt TwiVisual is user-friendly and easy to use. They are also satisfied with the effectiveness of using network graph as the visualization of link analysis and line graph as the visualization of timeline analysis. Most users agree that TwiVisual is easier to find the relationship between targeted user and its connections by using network graph and to construct a timeline of events with timestamp by using line graph. Most users can generate meaningful results, save time and workloads in forensic analysis.

5. Conclusion

In this study, the design and development of TwiVisual and its testing results are discussed. TwiVisual can import files, analyze Twitter data, visualize as graph and generate forensic analysis reports. Therefore, the objectives of this study were successfully achieved. Twivisual could facilitate digital forensic analysis. There are also limitations in this study. The first limitation of TwiVisual is that the graph in the system lacks interactive control. The second limitation is that the tool does not provide more options of graph. The third limitation is that the tool does not provide real-time data monitoring. For future work, there are a few improvements that can be considered such as the tool can include more interactive control for more functionalities, provide more options of graphs to users or provide real-time data monitoring.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

Conflict of Interest

Authors declare there is no conflict of interest regarding the paper's publication.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design:** *J. X. Ng, N. H. A. Rahman*; **data collection***: J. X. Ng, N. H. A. Rahman*; **analysis and interpretation of results**: *J. X. Ng, N. H. A. Rahman*; **draft manuscript preparation***: J. X. Ng, N. H. A. Rahman*. All authors reviewed the results and approved the final version of the manuscript.

References

[1] H. Shidek, N. Cahyani, and A. A. Wardana, "WhatsApp Chat Visualizer: A Visualization of WhatsApp

Messenger's Artifact Using the Timeline Method," *Int. J. Inf. Commun. Technol.*, vol. 6, no. 1, p. 1, Jun. 2020, doi: 10.21108/ijoict.2020.61.489.
[2] Z. Doshi, S. Nadkarni, K. Ajmera, and N. Shah, "TweerAnalyzer: Twitter Trend Detection and Visualization," *2017 Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2017*, 2018, doi: 10.1109/ICCUBEA.2017.8463951.
[3] G. Osborne and J. Slay, "Digital forensics infovis: An implementation of a process for visualisation of digital evidence," *Proc. 2011 6th Int. Conf. Availability, Reliab. Secur. ARES 2011*, pp. 196–201, 2011, doi: 10.1109/ARES.2011.36.
[4] S. Pirzada, N. H. A. Rahman, N. D. W. Cahyani, and M. F. Othman, "A Survey of Forensic Analysis and Information Visualization Approach for Instant Messaging Applications," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 2, pp. 237–246, 2023, doi: 10.14569/IJACSA.2023.0140229.
[5] S. F. Silva and T. Catarci, "Visualization of linear time-oriented data: A survey," *Proc. 1st Int. Conf. Web Inf. Syst. Eng. WISE 2000*, vol. 1, no. May, pp. 310–319, 2000, doi: 10.1109/WISE.2000.882407.
[6] H. Henseler and J. Hyde, "Technology assisted analysis of timeline and connections in digital forensic investigations," *CEUR Workshop Proc.*, vol. 2484, pp. 32–37, 2019.
[7] D. R. Kamble and N. Jain, "Digital Forensic Tools : a Comparative Approach," *Int. J. Adv. Res. Sci. Eng. IJARSE*, vol. 8354, no. 4, pp. 157–168, 2015, [Online]. Available: http://www.ijarse.com
[8] M. M. A. Doultani and M. M. Vijayalakshmi, "Data Forensics on Social Media," *2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019*, pp. 1–5, 2019, doi: 10.1109/I2CT45611.2019.9033627.
[9] A. A. G. S. Utama and B. Basuki, "Exploration of themes based twitter data in fraud-forensic accounting studies," *Cogent Bus. Manag.*, vol. 9, no. 1, 2022, doi: 10.1080/23311975.2022.2135207.
[10] H. Anber, A. Salah, and A. A. A. El-Aziz, "A Literature Review on Twitter Data Analysis," *Int. J. Comput. Electr. Eng.*, vol. 8, no. 3, pp. 241–249, 2016, doi: 10.17706/ijcee.2016.8.3.241-249.
[11] C. Boididou, S. Papadopoulos, M. Zampoglou, L. Apostolidis, O. Papadopoulou, and Y. Kompatsiaris, "Detection and visualization of misleading content on Twitter," *Int. J. Multimed. Inf. Retr.*, vol. 7, no. 1, pp. 71–86, 2018, doi: 10.1007/s13735-017-0143-x.
[12] B. Meyer, K. Bryan, Y. Santos, and B. Kim, "Twitterreporter: Breaking news detection and visualization through the geo-tagged twitter network," *Proc. ISCA 26th Int. Conf. Comput. Their Appl. CATA 2011*, no. January 2011, pp. 84–89, 2011.
[13] B. De Longueville, R. S. Smith, and G. Luraschi, "'OMG, from here, I can see the flames!': A use case of mining location based social networks to acquire spatio-temporal data on forest fires," *GIS Proc. ACM Int. Symp. Adv. Geogr. Inf. Syst.*, no. November 2009, pp. 73–80, 2009, doi: 10.1145/1629890.1629907.
[14] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *J. Comput. Sci.*, vol. 2, no. 1, pp. 1–8, 2011, doi: 10.1016/j.jocs.2010.12.007.
[15] Muskan, G. Singh, J. Singh, and C. Prabha, "Data Visualization and its Key Fundamentals: A Comprehensive Survey," *7th Int. Conf. Commun. Electron. Syst. ICCES 2022 - Proc.*, no. Icces, pp. 1710–1714, 2022, doi: 10.1109/ICCES54183.2022.9835803.
[16] A. E. Shadare, S. M. Musa, and C. Akujuobi, "Data visualization," no. December, 2016.
[17] S. Margret Anouncia, H. A. Gohel, and S. Vairamuthu, "Data visualization: Trends and challenges toward multidisciplinary perception," *Data Vis. Trends Challenges Towar. Multidiscip. Percept.*, no. March 2020, pp. 1–179, 2020, doi: 10.1007/978-981-15-2282-6.
[18] W. S. Ong and N. H. Ab Rahman, "A Forensic Analysis Visualization Tool for Mobile Instant Messaging Apps," *Int. J. Inf. Commun. Technol.*, vol. 6, no. 2, pp. 78–87, 2020, doi: 10.21108/ijoict.2020.62.530.
[19] E. Casey and C. W. Rose, *Handbook of Digital Forensics and Investigation*. Elsevier Academic Press, 2010. doi: 10.1016/C2009-0-01683-3.
[20] Y. W. Si, S. H. Cheong, S. Fong, R. P. Biuk-Aghai, and T. M. Cheong, "A layered approach to link analysis and visualization of event data," *7th Int. Conf. Digit. Inf. Manag. ICDIM 2012*, no. August, pp. 181–185, 2012, doi: 10.1109/ICDIM.2012.6360101.
[21] P. H. Nguyen, K. Xu, R. Walker, and B. L. W. Wong, "TimeSets: Timeline visualization with set relations," *Inf. Vis.*, vol. 15, no. 3, pp. 253–269, 2016, doi: 10.1177/1473871615605347.
[22] Montgomery, R. P., "Automated Reconstructions for the Digital Forensic Examiner Workflow," Air force institute of technology, pp.1-77, 2022.
[23] D. Stojanovski, I. Dimitrovski, and G. Madjarov, "TWEETVIZ: TWITTER DATA VISUALIZATION." [Online]. Available: http://radimrehurek.com/gensim/
[24] Chatvisualizer, "Chatvisualizer," Chatvisualizer. [Online]. Available: https://chatvisualizer.com/
[25] Y. HSingh and R. Malhotra, *Object-Oriented Software Engineering*. PHI Learning Pvt. Ltd, 2012. [Online]. Available:https://books.google.com.my/books?id=SDjyukTRYbcC&printsec=frontcover&num=15&redir_esc=y#v=onepage&q&f=false