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

AITCS

Vol. 6 No. 1 (2025) 505-520
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

JPEGHRepairKit: Corrupted JPEG Header Repair Kit using Header Substitution Method with JPEG Carving Feature

Farhana Syamimi Misran[1], Nurul Azma Abdullah[1*]

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

*Corresponding Author: azma@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.028

Article Info

Received: 30 July 2024
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

JPEG file, header repair, corrupted JPEG header, header substitution

Abstract

Recovering damaged image files is essential in digital forensics for determining and finding evidence, as corrupted JPEG files can delay investigations. This project aims to develop JPEGHRepairKit, a tool for repairing corrupted JPEGs, using Object-Oriented System Development methodology and Microsoft Visual Studio. The study employs header substitution methods to regain access to damaged JPEG files by focusing on their headers. The results indicate that JPEGHRepairKit effectively recovers images with header-related corruption, providing a significant contribution to digital forensic investigations. Future work may explore enhancements to the tool's algorithms and extend its capabilities to other image formats.

1. Introduction

The advent of the internet and the proliferation of digital devices have transformed how people communicate and access information. At the heart of these technological advancements is the concept of digital files, which serve as repositories for information accessible to computer programs. One of the earliest introductions to this concept was the digital image, invented by Russell Kirk[1] , which could convert physical images into digital ones using a scanner. Among the common digital image file formats is the Joint Photographic Experts Group (JPEG), which balances visual quality and memory size through its lossy compression method .

However, the widespread use of JPEG images across devices like digital cameras and smartphones makes them more susceptible to corruption compared to other image file formats[2]. JPEG header corruption, a common type of JPEG corruption, is particularly prevalent. This corruption can occur due to defective or damaged sectors on the storage disk, abrupt system shutdowns, or malicious malware attacks on the JPEG file[3].

In response to this issue, this project proposes the creation of a tool called JPEGHRepairKit, which employs a header replacement approach to address current challenges. The aim is to increase the success rate in fixing damaged files caused by missing headers. The problem this project seeks to address is the unreadability of images due to data corruption that occurs during processing and transmission, including editing processes and modifications made in applications like Google Photos [3].

The project's objectives include proposing a manageable corrupted JPEG header repair kit using a header substitution method, developing the JPEGHRepairKit tool, and testing its functionality. The tool will be developed using Microsoft Visual Studio version 17.7.0. It will not support batch repair and can only repair one file per session.

This tool also needs to be able to carve JPEG files from RAW file format and enable JPEG images recovered from embedded in the RAW files. All the images carved from the RAW files will then be saved to the user folder directly. The scope of the JPEGHRepairKit system includes functionalities for both regular users and digital forensic professionals. Regular users can utilize the tool to repair and recover corrupted JPEG files easily, while digital

forensic professionals can benefit from its ability to recover valuable data from corrupted images embedded in RAW files.

2. Related Work

The literature review provides a comprehensive overview of existing research related to JPEG image files, focusing on their structure, compression techniques, and common issues such as header corruption. This section aims to identify current repair methodologies and underscore the significance of addressing these vulnerabilities. By establishing the context and relevance of the proposed JPEG header repair kit, the literature review lays the groundwork for developing a customized solution.

2.1 Introduction

The idea of the proposed tool and issues have been discussed to establish the framework for the creation of the JPEG header repair kit. This section presents a literature review to discuss the related past system to the proposed tool which are the overview of JPEG, JPEG compression, JPEG header structure, common causes of JPEG header corruption, as well as significance and existing approaches to corrupted JPEG repair.

By navigating the complex terrain of this format, this literature review aims to clarify the basic principles of JPEG compression and the crucial function that the JPEG header structure plays. Potential vulnerabilities are revealed by investigating frequent sources of header corruption, and this section attempts to close these gaps in the knowledge base of current repair methods. This section also will explore the importance of header repair, and this evaluation provides a basis that drives the next conversation about creating a customized solution to deal with the problems observed in JPEG header repair.

2.2 JPEG Overview

Joint Photographic Experts Group (JPEG) is a widely known image compression used in digital images and the most used image format in various platform[4]. JPEG is a graphic image file that using lossy compression method produced by digital photography. Introduced to world in early 1990s to lead the image file format compression schemes[5], JPEG have two file structures which are JPEG file integrated format (JFIF) and JPEG exchangeable image file format (EXIF). Usually, JFIF can be found on the internet platform and EXIF easily found in images from digital cameras[6].

At first JPEG Interchange Format (JIF) used as specified container standard but it didn't be used by many people and later, JFIF was introduced and well use by a lot of people because it provides more and specific information than JIF. JFIF also allow JPEG bitstream to be exchanged through different platforms[6] as it was created solely for storing information and to transfer photographic images[5]. The only difference between JFIF and JIF is the additional requirement which is the need to have APP0 marker after the Start of Image (SOI). The APP0 marker are important to identify and specify a lot of things, including the Units, X and Y pixel density, the picture thumbnail, and information for specific applications.

Next, EXIF first launched in 1995. It was created by a company in Japan called Electronic Industries Development Association (JEIDA). EXIF can be embedded within JPEG files and other image file formats such as TIFF (Tag Image File Format) by digital camera. EXIF also stores important information metadata about the image. For examples the camera settings, date, time, and even the exposure level of the image. The detail information can be exposed to the internet when the image was uploaded and allows other users to access the detailed information[7]. EXIF metadata also often includes the thumbnail of the image as a quick display of the photo[8]. Initially, this file format was created to store basic information about the images, but now, EXIFs, can hold both image and video metadata.

2.3 JPEG Header Structure

JPEG files are composed of segments, each starting with a marker identified by a marker flag and the byte 0xFF. Common JPEG marker types include:

1. SOI (Start of Image): 0xFF, 0xD8. Marks the start of the image.
2. SOS (Start of Scan): 0xFF, 0xDA. Signals the start of the actual scan data.
3. DHT (Define Huffman Tables): 0xFF, 0xC4. Describes the use of Huffman tables during compression.
4. DQT (Define Quantization Table(s)): 0xFF, 0xDB. Defines the quantization tables used during compression.
5. DRI (Define Restart Interval): 0xFF, 0xDD. Specifies the interval between RST markers.
6. EOI (End of Image): 0xFF, 0xD9. Marks the end of the image.

Each marker type carries different payloads. For instance, the SOS marker includes details like the picture's color components and the Huffman table to be used [9]. The JPEG standard doesn't specify the resolution and aspect ratio of an image; these are provided by the JFIF application segment extension for JPEG. The JPEG file

format was designed to allow interchange of files containing JPEG-encoded data streams across different programs and systems.

2.4 Common Causes of JPEG Header Corruption

JPEG file corruption, particularly header corruption, can be caused by various factors such as abrupt system shutdowns, malware attacks, and virus-contaminated files[10]. These events can disrupt data writing to the storage medium, leading to incomplete files and errors in JPEG files. Additionally, malware can infect a system and corrupt data, including JPEG files, while viruses can infiltrate a system and cause damage to data, including JPEG images. For instance, Yvonne Tunnat's study found that out of 3070 JPEG files tested, 1007 had issues like faulty data, premature endings to data segments, missing markers, and incorrect file structures[4]. Regular backups of important JPEG files and maintaining robust security measures can help prevent such corruption.

The main problem with JPEG header corruption is that it renders the files unreadable by standard software. This is because the header contains crucial information about the file structure, such as the start of image markers (SOI), quantization tables, and Huffman tables. Without a properly functioning header, the file cannot be parsed correctly, leading to errors in displaying the image or, in some cases, complete inability to open the file. This is particularly problematic because there is a lack of discussion and focus on header corruption in many data recovery and image repair tools. Often, these tools focus on other types of corruption, leaving users without a reliable solution for header-specific issues.

2.5 Significance of JPEG Header Repair

JPEG Header Repair is essential for preserving the integrity of image files and digital data, as it stores metadata like image size, color space, and compression method. Corrupted headers can render files unopenable, display incorrectly, or even result in data loss. Serious consequences can occur, especially when dealing with large volumes of photos or critical data, such as in professional photography or scientific research. Data recovery often necessitates JPEG Header Repair to restore accessibility to corrupted image files. Therefore, understanding and implementing effective JPEG Header Repair techniques is crucial for maintaining data integrity across various applications.

2.6 Study of Existing Header Repair Technique and Tools

This study delves into the realm of JPEG header repair techniques and tools, focusing on the advancements and innovative approaches developed by researchers in the field of JPEG picture correction.

One such significant development is the DiskTuna: Photo Repair & Photo Recovery approach. This novel tool is for repairing corrupted JPEG files. It is a simple tool, and it has an effective solution for recovering corrupted JPEG files that a user has accidentally deleted or corrupted. A user-friendly interface shall be integrated to help users take the different steps with instructions. It executes several different techniques to repair a corrupted JPEG header DiskTuna algorithm. The first one is by using the header replacement method. This method works by replacing the corrupted header with healthy header that user choose as a reference[11]. This method will ensure that the corrupted JPEG header is replaced with compatible header.

Another notable tool is the JPG.Repair tool is a web-based application specifically designed to fix damaged JPEG files. The repair process involves replacing the damaged portion of the JPEG file with a new JPEG header from another image file that was captured under identical conditions[12]. Given that both images were likely taken with the same camera and at the same resolution, this technique is highly effective. To use JPG.Repair, users simply upload their damaged JPEG file to the website. The system then uses its proprietary technologies to attempt to recreate the image. If successful, the user can download the repaired image for free. However, for those who require more assistance or a manual review by an engineer, JPG.Repair offers a customized recovery solution at an additional cost.

Lastly, the manual method of repairing corrupted JPEG header first starts with examining and comparing healthy JPEG header with the corrupted one in hex editor[13]. By examining the differences between the two headers, the user will then be able to identify the main problem and which part is not properly saved to cause the images to be corrupted and would not display. This step can be achieved by comparing the images captured using the same digital camera. If there are no images from the same digital camera can be used, user can create new JPEG file by saving any doodling images from Paint in Windows 10 as JPEG format. Then the created files can be opened on hex editor to be compare with the corrupted files. Once the correct JPEG format is identified, the corrupted header will then be replaced by the new header that is not corrupted. This process requires the user to know the knowledge about file bits and bytes and can consume a lot of time[13]. Table 1 shows the comparison between the existing techniques and tools.

Table 1 *Hardware Requirements*

Category	Description	Ease of use	Success Rate	Additional features
DiskTuna	Header replacement method.	Easy	High	Photo recovery.
JPG.Repair	Replaces damaged portion using new JPEG header.	Easy	Medium	Manual review by engineer.
Manual method	Examines and comparing healthy and corrupted headers using hex editor.	Difficult	Low	None.
JPEGHRepairKit	Employs header substitution method by calculating the header length and substitute using possible header with same length.	Easy	High	JPEG carving feature.

The comparison includes three tools and one technique to recover corrupted JPEG due to corrupted header, which are DiskTuna, JPG.Repair, JPEGHRepairKit and a manual method. All techniques used by the existing tools and technique to repair the corrupted header is by using header replacement. Although all of them use the header replacement method, JPEGHRepairKit is the only one that did not require another file to be used as reference to replace the header. Users will not be burdened with finding file references by using this method.

3. Methodology

This section covers the development process for the JPEGHRepairKit utility..

3.1 Introduction

Effective project management is the foundation of any successful project. When a project is well managed, this will guarantee the attainment of the project's goals. and the expected outcome will be achieved on time. Hence, the OOSD Model, or object-oriented software development, is the approach used to complete this project. Plan, analyze, design, execute, and test are the five stages of this process.

3.2 Object-oriented Software Development (OOSD)

Phases analysis, design, implementation, and testing are the four key stages of object-oriented software development (OOSD). The methodology for object-oriented development is depicted in Fig. 1. According to [14], there are four different types of development cycles: development from scratch [9], development via reuse [15], development with reuse, and development of reusable [16]. In this project, the problem definition, analysis, requirement collection, system analysis, system design, implementation, and training and review phases are the six main stages of the process that will be used and implemented.

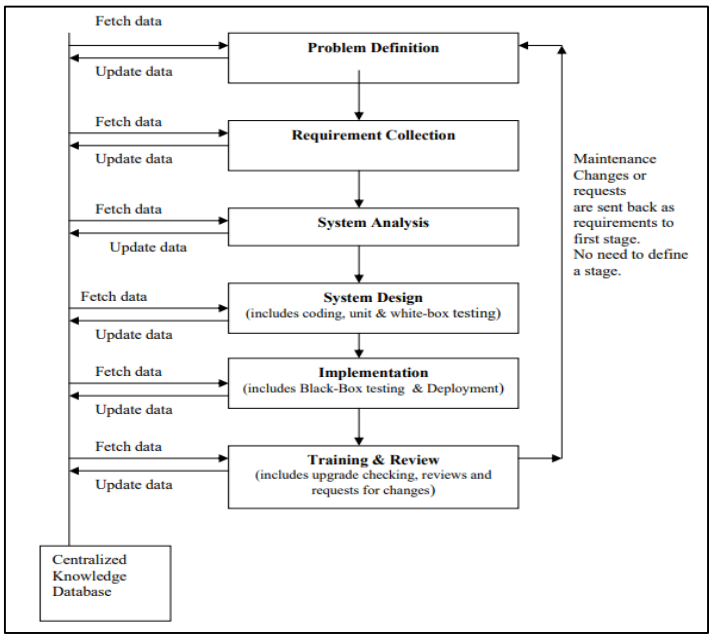

Fig. 1 *Object-Oriented Software Development (OOSD) Model*

Problem definition is the initial phase for this project, and it involves definition and identification of the main issue: data corruption in JPEG files due to header corruption or missing headers. Data corruption will occur when the data is altered during the processing or transmission phase. It will result in unreadable file images. To address this issue a problem statement was established. It will be focusing on developing a solution to repair corrupted JPEG headers. Thus, in this project, JPEGHRepairKit will utilize a header replacement.

A careful selection of hardware, operating systems, and software tools was done during the requirement collection phase to develop a successful application. Table 2 and 3 provide a detailed summary of the necessary components used in the project, including the operating system, hardware setups, and software programs that are necessary to complete this project. The project's goals may be realized more easily when these technologies are strategically used as the basis for the framework. This thorough selection of hardware and software components highlights the methodical and purposeful steps taken to provide the best possible development environment to produce a functional application.

Table 2 *Hardware Requirements*

No.	Hardware	Specification
1.	Processor	AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
2.	Random Access Memory	8.00 GB
3.	System type	64-bit Operating System

Table 3 *Software Requirements*

No.	Software	Specification
1.	Microsoft Visual Studio 2022	Version 17.7.0
2.	Visual C#	Programming Language

In the analysis phase of the project, current existing algorithm was meticulously looked at, with a strong emphasis on a thorough comprehension of the JPEG format and header structures. Using the object-oriented software development process, we carried out a comprehensive comparison study to determine differences and determine whether the system could support the suggested repair tool. A review of the literature clarifying the functional requirements for JPEGHRepairKit served as the phase's conclusion. The succeeding development phases will be guided by the insights that are obtained, which will provide a reliable and efficient solution for correcting faulty JPEG headers using the novel header substitution approach.

The JPEGHRepairKit tool will have a Graphical User Interface (GUI) during the system design stage. A user can interact with computer software, such as an operating system, by using the GUI (graphical user interface), which is a set of interactive elements including icons and other graphical objects [17]. A user interface consisting of the dashboard, repair, view, login, and registration modules is available for the JPEGHRepairKit utility. The computer language C# will be utilized to construct the project. This project's overall purpose will be explained using a use-case diagram. The project's sequence diagram will specify the functions of each object and their relative orders.

In the system design phase, white-box testing was not utilized because the focus of this project was primarily on ensuring that the system meets the specified functional requirements rather than examining the internal structures or workings of the application. Similarly, during the implementation phase, black-box testing was chosen over white-box testing as it aligns better with the project's emphasis on validating output correctness without delving into the code structure. These choices were made to streamline the testing process and ensure that the system functionality meets user expectations effectively.

During the project's implementation phase, the actual coding of every system module becomes the main emphasis. This critical phase entails the painstaking conversion of conceptual concepts into the physical parts that make up the JPEGHRepairKit tool. At this stage, the classes of the objects and their connections are interpreted in a methodical manner, considering their inherent structure. The modules are made tangible by using C# programming language. Furthermore, databases are carefully designed, adding to the application's functional outline. The basis for a reliable and functional JPEG header restoration solution is laid by this painstaking coding project.

Finally, the training and review phase will be involved in educating users on how to effectively use the JPEGHRepairKit and gathering feedback for continuous improvement. The training sessions were conducted to

ensure that user could navigate the interface and utilize the tool's features. Periodic reviews were carried out to check for necessary upgrades and to address any user requests for changes. This phase ensured that the tool remained user-friendly and efficient. All feedback and review results were documented in the centralized knowledge for future enhancements.

4. Analysis and Design

The analysis and design phase of the JPEGHRepairKit tool is essential for ensuring that the system meets user needs and functions effectively. During this phase, the system requirements are identified, and a detailed architecture is defined to address the problem of repairing corrupted JPEG headers. This section outlines the methodologies and processes used to create a robust and user-friendly tool for digital forensics and data recovery.

4.1 Introduction

The Software Development Life Cycle (SDLC) improves the process and quality of software development by providing guidelines for project design, development, and maintenance. This ensures all functional requirements and project objectives are met. Before starting the implementation phase, these steps are crucial to ensure the application delivers the desired results. Use-case, sequence, and class diagrams from the Unified Modeling Language (UML) are used in the analysis. An entity relationship diagram is used to explain the database architecture, and a flowchart illustrates the application flow during the design stages. The section also describes the interface design.

4.2 System Requirements Analysis

The JPEGHRepairKit tool's functional and non-functional needs will be discussed in this section. To achieve the goals and get the desired results, the system requirements must be assessed. The system operation is described by functional requirements. These may be found in Table 4 for the JPEGHRepairKit utility. Table 5 of the JPEGHRepairKit utility includes the outlines of the non-functional requirements for the system.

Table 4 *Functional Requirements*

No.	Requirements	Description
1.	Insert corrupted JPEG file	The software will allow the user to choose any image to be repaired.
2.	Repair corrupted JPEG file	The chosen picture file can be fixed by the user. Following the procedure, the repaired file can be saved by the user.
3.	View repaired JPEG file	The file that has been repaired using the header substitution approach will be visible to the user.
4.	Save repaired JPEG file	The repaired image file will be able to be save by the user.
5.	Insert .raw file	The software will allow the user to upload a .raw file for processing.
6.	Carve JPEG from .raw file	The tool will enable the user to extract or carve JPEG images from the uploaded .raw file.
7.	Save carved images	The user will be able to save the extracted JPEG images from the .raw file.

The table outlines the functional requirements of the software, detailing the specific tasks and functionalities it must perform to meet user needs. Each requirement is accompanied by a brief description clarifying its purpose and functionality within the software. These requirements range from basic functionalities like inserting and repairing corrupted JPEG files to more advanced features like carving JPEG images from .raw files and saving the extracted images. Overall, these requirements serve as a comprehensive guide for the development team to ensure that the software meets user expectations and delivers the necessary functionalities.

Table 5 outlines the non-functional requirements for the JPEGHRepairKit tool, focusing on essential aspects that ensure its proper operation and user experience. The 'Compatibility' requirement specifies that the tool should be capable of running on Windows 10 and higher operating systems, ensuring it can be used on modern computer systems. The 'Performance' requirement mandates that all functions of the program should be readily accessible every time the user turns it on, ensuring reliable and consistent operation. The 'Usability' requirement highlights the need for the application's menus to be easy to navigate, with clear and understandable buttons, facilitating ease of use even for users with basic computer skills. Finally, the 'Error Handling' requirement emphasizes that the tool should provide basic error messages to inform users of any issues encountered during the repair process, thereby enhancing user support and troubleshooting capabilities.

Table 5 *Non-Functional Requirements*

No.	Requirements	Description
1.	Compatibility	The tool should run on Windows 10 and higher operating systems.
2.	Performance	Every time the user turns on the program, all its functions ought to be accessible to them.
3.	Usability	The application's menus should be simple for users to navigate, with understandable buttons.
4.	Error handling	The tool should provide basic error messages to inform users of any issues during the repair process.

Table 6 outlines the essential user requirements for the JPEGHRepairKit tool. Firstly, users must have the ability to upload a file using the tool, ensuring an easy and straightforward file submission process. Secondly, the tool should allow users to view the uploaded file within the software interface, providing a preview for verification. Thirdly, users should be able to initiate the repair process for the submitted file with a simple command. Fourthly, once the repair process is complete, users should be able to view and select from multiple versions of the repaired file to choose the best result. Finally, the tool must enable users to download the necessary file from the tool, ensuring they can easily retrieve their repaired images. These requirements ensure comprehensive and user-friendly experience for individuals seeking to repair corrupted JPEG files.

Table 6 *User Requirements*

No.	Requirements
1.	User should be able to upload a file using the tool.
2.	User should be able to view the file that was uploaded using the software.
3.	User should be able to begin fixing the file that was submitted.
4.	User should be able to view and select from several repaired file.
5.	User should be able to download the necessary file from the tool.

4.3 Unified Modelling Language Specification

Software documentation may now be done in a variety of methods with the help of Unified Modelling Language, or UML [18]. The purpose of the Unified Modelling Language is to serve as a visual modelling language for system analysis, design, and implementation. UML creates a visual model of the software system using graphic notation. This project presents two different forms of UML diagrams: sequence diagrams and use-case diagrams.

The use-case diagram serves as an illustration of how the system's items interact with one another in the sequence diagram. The user and the use-case are the two key components of a use-case diagram. The use-case is a collection of actions that specifies the interaction between the actor and the system, whilst the user is represented as the external entity that interacts with the system.

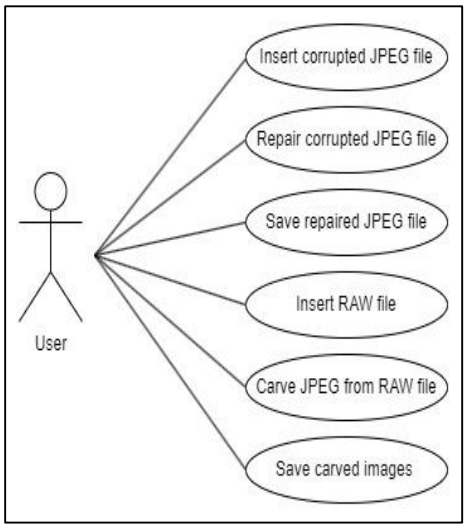

Fig. 2 *Use-Case Diagram for JPEGHRepairKit*

The use case in Fig. 2 illustrates the interactions between a single actor, the User, and the system for repairing corrupted JPEG files and carving JPEG images from .raw files. The user can perform several tasks: inserting a corrupted JPEG file into the system, initiating the repair process for the corrupted file, and viewing the repaired JPEG files. Additionally, the user can insert a .raw file, execute the carving process to extract JPEG images from this file, and subsequently save the carved images to a specified folder. These interactions encapsulate the primary functionalities of the system aimed at JPEG file repair and carving operations.

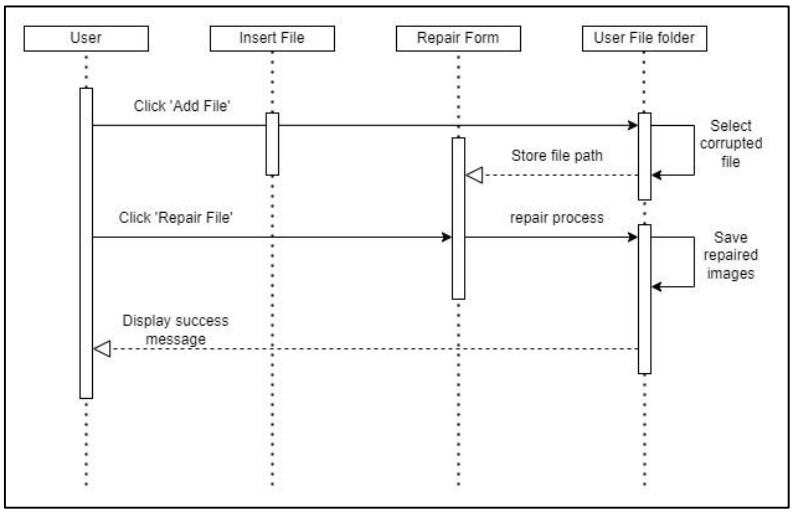

Fig. 3 *Sequence Diagram for Repair and View*

Next, the system's functional requirements are examined using common software development techniques, such as sequence diagrams [19] (as shown in Fig. 3). The corrupted JPEG file can be selected by the user from their PC. Once the file has been chosen, the database will save it. The user can choose the corrupted file to be repaired from the database. The user can save the file to their devices once the file repair is complete.

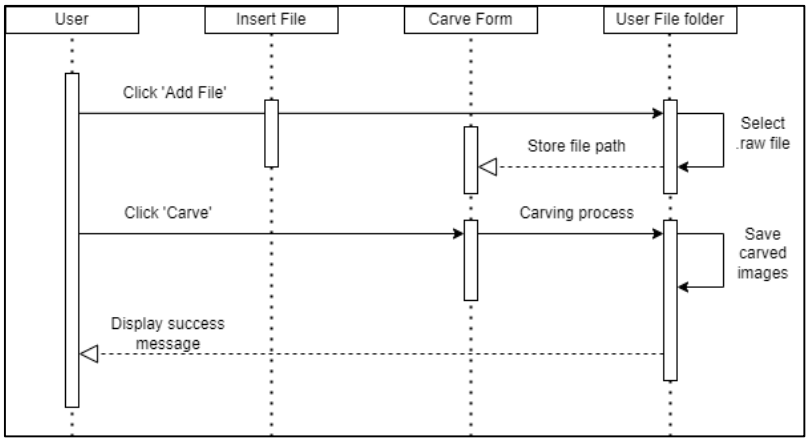

Fig. 4 *Sequence Diagram for File Carving*

The system architecture depicted in Fig. 4 illustrates the interaction and components of the JPEG Repair and Carving tool. The architecture comprises four main entities: the User, Insert File, Carve Form, and User File Folder. The User initiates the process by interacting with the user interface to select and upload .raw or corrupted JPEG files through the Insert File and User File Folder. Then the file path will be stored in carve form. The user will then click ‘Carve’ causing Carve Form to initiate the carving process. Once the carving process is done, the system will save the carved images into the User File Folder and display a message that the carving process succeed.

Fig. 5 shows the JPEGHRepairKit tool's system architecture. There are two different paths that the user can go along. The first one will direct the user to go to the repair page. The repair page will prompt the user to choose corrupted jpeg file to be repaired. Then after the file is repaired, the system will display the successfully repaired JPEG file. User will then be able to save and quit the system if user wants to, if not, user may go to the next path which is, going to the carving page. The carving page also will ask users to choose a file, but this time, the file format should have RAW extension. Then the system will carve JPEG image and save it to the user computer.

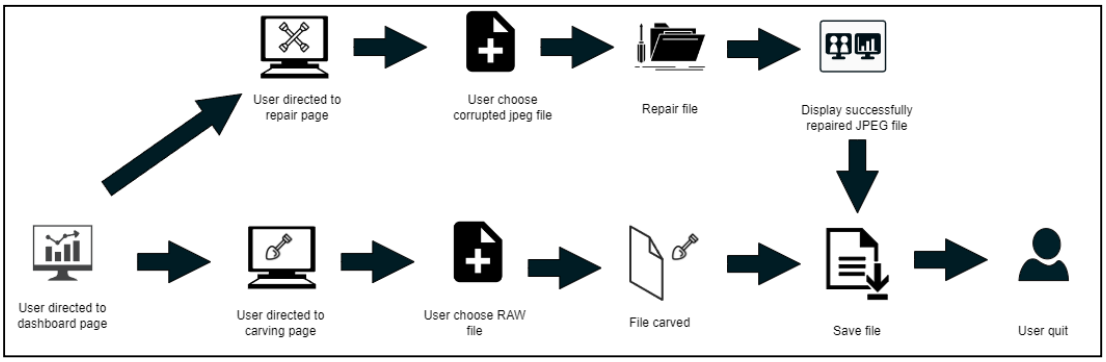

Fig. 5 *System Architecture for JPEGHRepairKit*

4.4 Interface Design

In order to develop a functioning and user-friendly environment, user interface (UI) design is crucial. It speaks about the way a user may engage visually with an application. The user interface (UI) of an application may also impact its usability and user experience. Based on each of the processes shown in Fig. 3 and Fig. 4, the interfaces listed below have been created. Microsoft Visual Studio 2022 will be used to design each interface.

Fig. 6 *Interface Design for Dashboard JPEGHRepairKit*

Based on Fig. 6, the interface design features a navigation bar positioned at the top of the dashboard, comprising four buttons: "Home," "About," "User Manual," and "Help." Additionally, the navigation bar includes a "Quit" button for exiting the application and the UTHM logo, providing branding and easy access to essential features. In the center of the interface, two prominent buttons labeled "Repair File" and "Carve File" are positioned, allowing users to navigate to other pages for repairing corrupted files and carving JPEG images from .raw files, respectively. This layout prioritizes user accessibility and functionality, providing clear and intuitive navigation options while maintaining a cohesive design aesthetic.

Fig. 7 *Interface Design for JPEG repair page*

The JPEG repair interface allows users to choose a corrupted JPEG file from their computer. Once selected, the interface displays the file path and provides buttons for adding the file and initiating the repair process. This simple design ensures easy navigation and usability for repairing JPEG files without overwhelming the user with technical details.

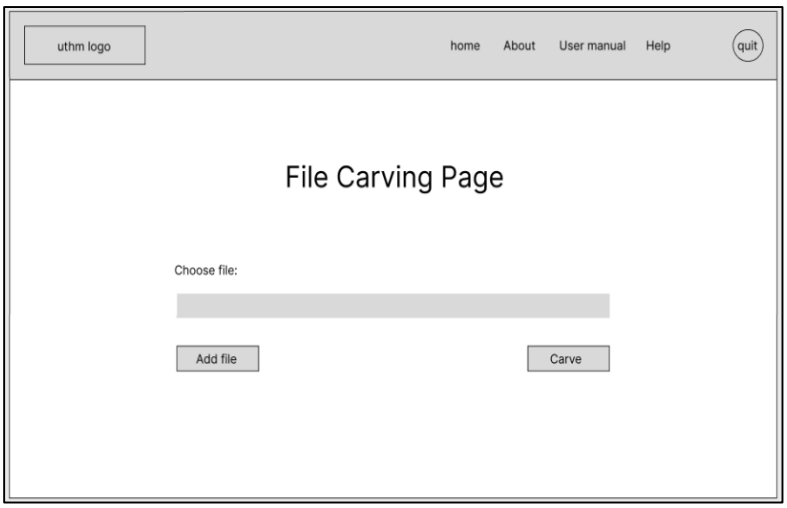

Fig. 8 *Interface Design for File Carving Page*

The carve module interface in the JPEGHRepairKit facilitates the extraction of data from RAW files. It presents users with a simple yet intuitive design to guide them through the data recovery process. The interface includes two primary elements: an 'Add File' button, allowing users to select their desired file through a file dialog, and a 'Carve' button, which initiates the carving process. This design ensures ease of use and straightforward navigation for users seeking to salvage data from RAW files.

4.5 Interface Design

The interface design phase is crucial for developing a user-friendly and intuitive experience for the JPEGHRepairKit tool. This phase focuses on creating a visually appealing and functional interface that allows users to easily navigate and utilize the tool's features for repairing corrupted JPEG headers. In this section, we outline the design principles, user interface components, and interaction flows that ensure the tool is accessible and efficient for all users.

Fig. 9 *View Repaired Interface Design for JPEGHRepairKit*

Fig. 9 shows the interface of dashboard that consists of several buttons for navigation. The buttons included at the top bar are home button, about button, user manual button, help button and quit button. The buttons at the center are repair file button and carve file which will lead user to the other module in the system.

Fig. 10 *Repair module interface*

The interface for JPEG repair page is designed to allow users to select a corrupted JPEG file from user's computer files. This interface will display the chosen file path and execute the repair process once the user clicks assigned button. In Fig 10, the repair module interface contained two functional buttons. The 'Add File' button will open file dialog and the 'Repair File' button will execute the repair mechanism embedded in the system.

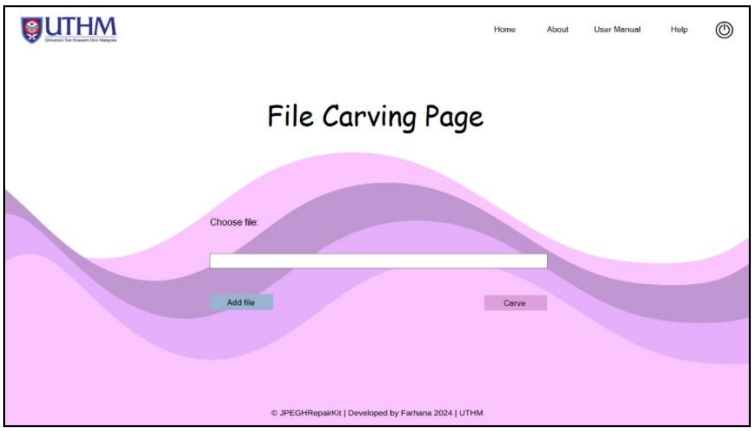

Fig. 11 *File carving module interface*

Carve module is one of the vital modules in the JPEGHRepairKit. This module is used to extract and salvage data from RAW files. The carve module interface will guide users through data recovery. Fig 11, illustrates the carve module interface that is intuitively to provide user guidance to use the system. The system consists of a few elements that will facilitate the carving algorithm. The elements are 'Add File' button, that will act as a gateway for user to choose their desire file in file dialog, and 'Carve' button that will carry out the carving.

4.6 Functional Testing

The test phase is developed for functional testing. It is to ensure that the module is performing as reported by the requirement specified in the system analysis and design phases. The test plan consists of a test case, expected output, and actual output. This is the situation that will be tested which is whether the system performing just like what is being expected or not. The functions are tested based on the output to ensure that the system acquires all the needs and functionality. Each module is tested to make sure there are no errors happening in the system. There are two tests that were carried out, which are application functionality testing and user acceptance testing.

During functional testing, no existing datasets were utilized because there were no suitable datasets containing corrupted JPEG headers from FF D8 until FF DB. Consequently, a custom dataset was created specifically for this project to ensure comprehensive and accurate testing of the JPEGHRepairKit functionalities. The user acceptance testing involved 7 testers, consisting of both common people, including those with and without IT backgrounds. This diverse group of users was selected to ensure that the system is user-friendly and functional across different levels of technical expertise.

Application functionality testing is a critical phase in the software development lifecycle, focusing on verifying that the application operates according to its specified requirements and delivers the intended functionalities to the users. This process involves systematically testing each feature of the application to ensure it behaves

correctly and meets user expectations under various conditions. By conducting comprehensive functionality testing, developers can identify and rectify defects early, thus enhancing the overall reliability and user experience of the application.

Table 7 *Test case result for Dashboard module*

No.	Test case	Expected results	Results
1.	Home button functionality	The homepage user control is displayed	Success
2.	About home button functionality	The about page user control is displayed	Success
3.	User manual button functionality	The user manual page user control is displayed	Success
4.	Help button functionality	The help page user control is displayed	Success
5.	Quit button functionality	The application closes	Success
6.	Repair file button functionality	The repair module user control is displayed	Success
7.	Carve file button functionality	The carve module user control is displayed	Success
8.	Cancel quit functionality	The application remains open, and the dashboard still displayed	Success
9.	Navigation persistence	The dashboard correctly displays after returning from other modules	Success
10.	Display correct content based on user input	Each button click dynamically updates the dashboard to display the appropriate user control	Success

The "Test case result for Dashboard module" table details the outcomes of various tests conducted on the dashboard module of the JPEGHRepairKit tool. Each test was designed to verify the proper functioning of specific buttons and navigation elements within the dashboard. The tests confirmed that the navigation buttons (Home, About, User Manual, Help) correctly display their respective user controls, indicating successful transitions between different sections of the tool. The 'Quit' button effectively closed the application, while the 'Cancel Quit' functionality ensured the application remained open if the user chose not to quit. Additionally, the 'Repair File' and 'Carve File' buttons successfully navigated to their respective modules. The tests also verified that navigation persisted correctly, and the dashboard dynamically updated to display appropriate content based on user input. All tests were successful, confirming the dashboard's reliability and user-friendly interface.

Table 8 *Test case result for Repair module*

No.	Test case	Expected results	Results
1.	Add File button functionality	File dialog opens, displaying only JPEG files (.jpg, .jpeg).	Success
2.	Display selected file path	Selected file path is displayed in the textbox.	Success
3.	Repair File button without file selection	Message box prompts: "Please select a file first."	Success
4.	Repair corrupted JPEG file	System checks file corruption, executes repair, displays success message, saves repaired file.	Success
5.	Handle non-corrupted JPEG file	Message box prompts: "The selected file is not corrupted."	Success
6.	Verify multiple header replacements	System creates up to ten possible repaired files using different headers.	Success
7.	Repaired file existence check	Repaired file is created and exists at specified file path.	Success
8.	Invalid file type handling	File dialog filters non-JPEG files, error message displayed if selected.	Success

Table 8 summarizes the test case results for the Repair module of the JPEGHRepairKit tool. Each test case was designed to validate key functionalities, such as the "Add File" button opening a file dialog that displays only JPEG files and displaying the selected file path in a textbox. It also includes scenarios like handling file repair without selection, which prompts a message, and the successful repair of corrupted JPEG files. The tests ensure the system handles non-corrupted JPEG files and multiple header replacements correctly, confirms the existence of repaired files, and filters out non-JPEG files in the dialog. All tests were successfully passed, indicating the module's robustness and reliability.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 505-520

Table 9 *Test case result for Carve module*

No.	Test case	Expected results	Results
1.	Add File button functionality	File dialog opens, displaying only RAW files (.raw).	Success
2.	Display selected file path	Selected file path is displayed in the textbox.	Success
3.	Carve button without file selection	Message box prompts: "Please select a file first."	Success
4.	Carve RAW file	System executes carving process, displays completion message, and saves carved images to specified folder.	Success
5.	Save carved images to specified folder	System prompts user to select a folder for saving carved images and saves images there.	Success
6.	Handle invalid file type	File dialog filters non-RAW files, error message displayed if selected.	Success
7.	Verify carved images existence	Carved images are created and exist in the specified folder.	Success
8.	Error handling during carving process	Error message displayed if carving process encounters issues, and no images are saved.	Success
9.	Carve button functionality	Carve button initiates the carving process and completes without errors.	Success
10.	Valid JPEG extraction	System accurately identifies and extracts valid JPEG images from RAW file.	Success

Table 9 outlines the test case results for the Carve module of the JPEGHRepairKit tool. The first test case verifies that the "Add File" button opens a file dialog displaying only RAW files, ensuring the correct file type is selected. The second test case confirms that the selected file path is shown in the textbox, providing user confirmation. Subsequent tests check the system's response when the "Carve" button is clicked without file selection, ensuring the user is prompted to choose a file first. The fourth and fifth test cases validate the entire carving process, from executing the carving to saving the carved images in a user-specified folder and displaying a completion message. Handling invalid file types and verifying the existence of carved images are also tested, ensuring robustness and correctness in image extraction. Additionally, the table includes tests for error handling during the carving process, the functionality of the "Carve" button, and the accurate identification and extraction of valid JPEG images from the RAW file. All test cases passed successfully, demonstrating the reliability and effectiveness of the Carve module.

The next test carried out was user acceptance testing. These testing plays a pivotal role in software development, ensuring that the application meets user requirements and functions correctly in real-world scenarios. Unlike other testing phases, UAT focuses on validating the software's usability, functionality, and adherence to business objectives from the end user's perspective. It involves stakeholders and end users testing the software to identify any discrepancies between expected and actual outcomes, aiming to refine the product and ensure it delivers the intended value before deployment. UAT not only enhances software quality but also boosts user confidence and satisfaction in the final product.

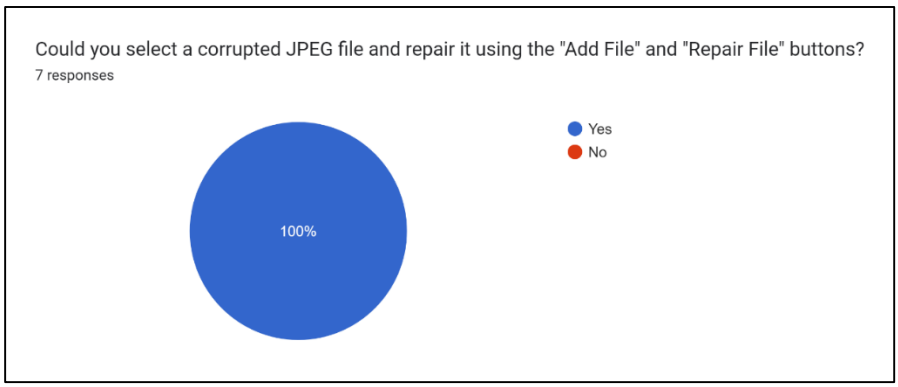

Fig. 12 *Questionnaire 1 result*

Fig. 12 displays the results from Questionnaire 1, which asked participants if they could successfully select a corrupted JPEG file and repair it using the "Add File" and "Repair File" buttons. The responses indicate a 100%

affirmative rate, with all seven participants answering "yes." This suggests that the interface design and functionality of the repair module, particularly the file selection and repair process, were effective and intuitive, meeting users' expectations and needs without reported issues or failures.

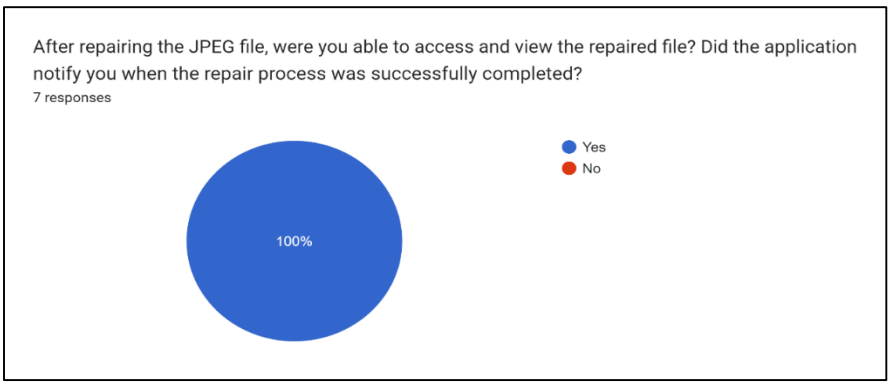

Fig. 13 *Questionnaire 2 result*

Fig. 13 presents the results from Questionnaire 2, which asked participants two key questions regarding their experience with the application. The first question inquired whether users could access and view the repaired JPEG file after the repair process. The results indicated a 100% affirmative response, indicating that all participants were able to successfully access and view their repaired files. The second question asked whether the application notified users when the repair process was completed successfully. Again, the results showed a 100% positive response, indicating that all participants received notifications upon successful completion of the repair process. These results underscore the effectiveness and reliability of the application in repairing and notifying users about the completion of the JPEG file repair process.

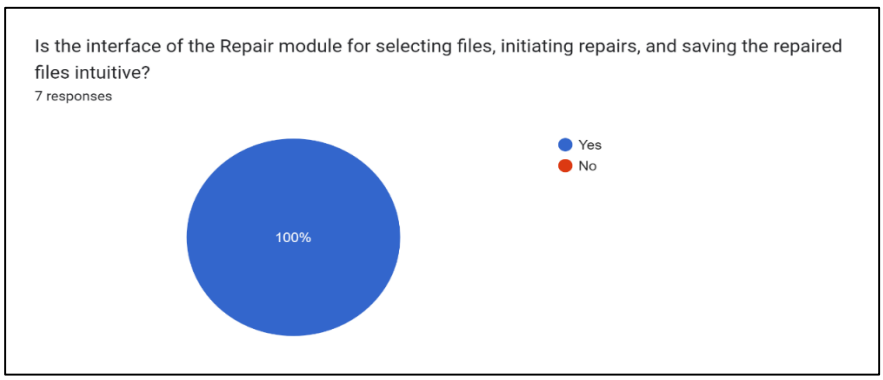

Fig. 14 *Questionnaire 3 result*

Fig. 14 displays the results from Questionnaire 3, specifically addressing the Repair module's interface usability. The question queried whether users found the interface intuitive for tasks such as selecting files, initiating repairs, and saving the repaired files. The results indicate that all seven respondents answered "yes," representing 100% agreement that the interface design effectively supports these functionalities without any reported issues or reservations.

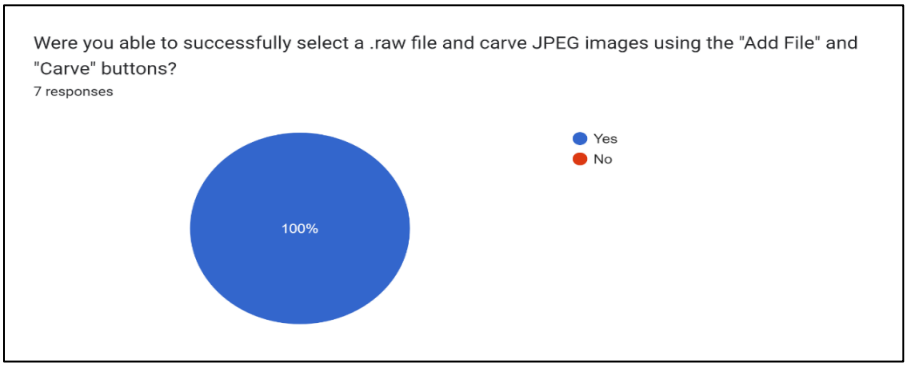

Fig. 15 *Questionnaire 4 result*

Fig. 15 presents the results of Questionnaire 4, which asked users if they were able to successfully select a .raw file and carve JPEG images using the 'Add File' and 'Carve' buttons. All 7 respondents (100%) answered 'Yes,' indicating that every user could effectively use these features to extract JPEG images from .raw files. This unanimous positive feedback highlights the tool's intuitive design and reliable functionality, confirming that the carving module works as intended and enhances the JPEGHRepairKit's overall effectiveness in recovering images from raw data files.

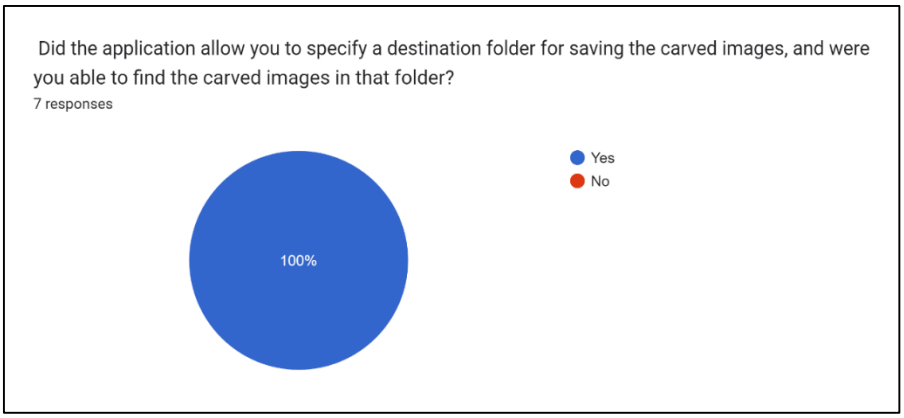

Fig. 16 *Questionnaire 5 result*

Fig. 16 presents the results of Questionnaire 5, where participants were asked, "Were you able to successfully select a .raw file and carve JPEG images using the 'Add File' and 'Carve' buttons?" The data shows that 100% of respondents answered "yes," indicating that all users were able to successfully navigate the interface, select a .raw file, and initiate the JPEG carving process without encountering any issues or difficulties.

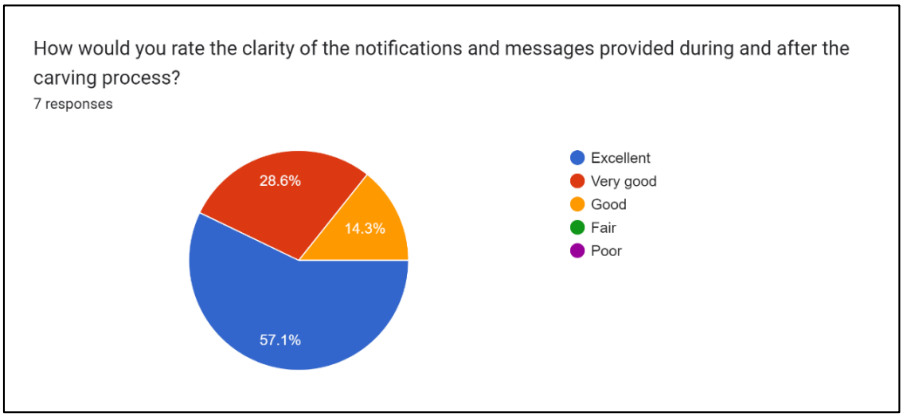

Fig. 17 *Questionnaire 6 result*

Fig. 17 displays the results of Questionnaire 6, focusing on user feedback regarding the functionality of selecting a .raw file and carving JPEG images using the "Add File" and "Carve" buttons. The responses indicate a positive reception, with 57.1% rating the experience as excellent, 14.3% as good, and 28.6% as very good. These findings suggest that most users found the process of selecting and carving files straightforward and effective, reflecting a high level of satisfaction with the carve module's usability and functionality.

5. Conclusion

The JPEGHRepairKit project successfully achieved its primary objectives by developing a tool that utilizes header substitution methods to repair corrupted JPEG files, allowing users to recover and view damaged images. The system demonstrated several advantages, including a user-friendly interface and reliable functionality in repairing JPEG headers and carving images from RAW files. However, the tool's limitations include its inability to support batch processing and restricted file format support, which currently focuses only on JFIF and EXIF formats. Future improvements should address these weaknesses by incorporating batch repair capabilities, expanding file format compatibility, enhancing error detection algorithms, and refining the user interface based on feedback to ensure seamless user experience and broader applicability.

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 that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design****: F. S. Misran, N. A. Abdullah;* ***data collection****: F. S. Misran, N. A. Abdullah;* ***analysis and interpretation of results****: F. S. Misran, N. A. Abdullah;* ***draft manuscript preparation****: F. S. Misran, N. A. Abdullah. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] A. E. Ilesanmi, T. Ilesanmi, and G. A. Gbotoso, "A systematic review of retinal fundus image segmentation and classification methods using convolutional neural networks," *Healthcare Analytics*, vol. 4, p. 100261, 2023, doi: https://doi.org/10.1016/j.health.2023.100261.

[2] P. Mullan, C. Riess, and F. Freiling, "Forensic source identification using JPEG image headers: The case of smartphones," *Digit Investig*, vol. 28, pp. S68–S76, 2019, doi: https://doi.org/10.1016/j.diin.2019.01.016.

[3] "JPG Signature Format: Documentation & Recovery Example." Accessed: May 23, 2024. [Online]. Available: https://www.file-recovery.com/jpg-signature-format.htm

[4] A. Shawahna, Md. E. Haque, and A. Amin, "JPEG Image Compression using the Discrete Cosine Transform: An Overview, Applications, and Hardware Implementation," *CoRR*, vol. abs/1912.10789, 2019, [Online]. Available: http://arxiv.org/abs/1912.10789

[5] R. H. Wiggins, H. C. Davidson, H. R. Harnsberger, J. R. Lauman, and P. A. Goede, "Image file formats: past, present, and future," *Radiographics*, vol. 21, no. 3, pp. 789–798, 2001.

[6] S. A. Sari and K. M. Mohamad, "A review of graph theoretic and weightage techniques in file carving," in *Journal of Physics: Conference Series*, 2020, p. 52011.

[7] "What are EXIF files and how to open them? | Adobe." Accessed: May 31, 2024. [Online]. Available: https://www.adobe.com/my_en/creativecloud/file-types/image/raster/exif-file.html

[8] T. Tachibanaya, "Description of Exif file format," *URL http://park2. wakwak. com/tsuruzoh/Computer/Digicams/exif-e. html*, 2001.

[9] A. N. E. Abdi, "Corrupted MP4 carving using MP4-Karver," Universiti Tun Hussein Onn Malaysia, 2016.

[10] Y. Tang *et al.*, "Recovery of heavily fragmented JPEG files," *Digit Investig*, vol. 18, pp. S108–S117, 2016.

[11] "Repair Corrupt JPEG Header – DiskTuna // Photo Repair & Photo Recovery." Accessed: Jun. 26, 2024. [Online]. Available: https://www.disktuna.com/repair-corrupt-jpeg-header/

[12] "Repair corrupt, unreadable, encrypted JPEG pictures online." Accessed: Jun. 13, 2024. [Online]. Available: https://jpg.repair/

[13] "How to Fix Broken or Corrupt JPEG Header - Stellar Data Recovery." Accessed: Jun. 26, 2024. [Online]. Available: https://www.stellarinfo.com/blog/fix-broken-corrupt-jpeg-header/

[14] G. Engels and G. Kappel, "Object-Oriented System Development: Will the New Approach Solve Old Problems?," in *Proceedings of the IFIP 13th World Computer Congress on Information Processing, Hamburg (Germany)*, Elsevier, 1994, pp. 434–441.

[15] Eoghan Casey, "Digital_Evidence_and_Computer_Crime".

[16] P. Daras, I. Kompatsiaris, T. Raptis, and M. G. Strintzis, "MPEG-4 authoring tool for the composition of 3D audiovisual scenes," in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, 2001, pp. 201–204 vol. 2. doi: 10.1109/ISCAS.2001.921042.

[17] D. Farmer and W. Venema, *Forensic discovery*. Addison-Wesley Professional, 2009.

[18] B. Rumpe, *Modeling with UML*, vol. 98. Springer, 2016.

[19] F. Facts, "UML Sequence Diagrams".