

A
Project Report
on
**GRAPHICAL PASSWORD AUTHENTICATION USING
CUED CLICK POINTS**

Submitted in partial fulfilment of the requirement for the award degree of

Master of Science (Computer Science)

in

Department of Computer Science



Submitted by

JAMMULA SAI KUMAR (24000I1003)

UNDER THE GUIDANCE OF

Dr. D.RAMESH

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE, KAKATIYA UNIVERSITY

VIDYARANYAPURI, HANAMKONDA, WARANGAL-506009

2023-2025

DEPARTMENT OF COMPUTER SCIENCE
KAKATIYA UNIVERSITY
VIDYARANYAPURI, HANAMKONDA, WARANGAL-506009



CERTIFICATE

This is to certify that **JAMMULA SAI KUMAR** (2400011003) student of M.Sc.(Computer Science) has satisfactorily completed the dissertation work entitled **“GRAPHICAL PASSWORD AUTHENTICATION USING CUED CLICK POINTS”** in the partial fulfilment of the requirements of the MSC during this academic year 2023-2025 under the guidance of **Dr. D.RAMESH**, Assistant professor, Department of Computer Science, K.U.

SUPERVISOR

EXTERNAL

HEAD

DECLARATION BY THE CANDIDATE

I **JAMMULA SAI KUMAR (24000I1003)** do hereby certify that the project report entitled “**GRAPHICAL PASSWORD AUTHENTICATION USING CUED CLICK POINTS**“ ,under the guidance **Dr. D.RAMESH** , Assistant Professor, Department Of Computer Science, Kakatiya University, Warangal, submitted in partial fulfilment of the requirements for the Award of the Degree of MSc. Computer Science.

This is a record of Bonafide work carried out by me and the results embodied in this Project Report has not been produced / copied from any source. The results embodied in this Project Report have not been submitted to any other University or Institute for the Award of any other Degree or Diploma.

JAMMULA SAI KUMAR (24000I1003)

ACKNOWLEDGEMENT

I thank the almighty for giving me the courage and perseverance in completing the project. This project itself is an acknowledgement for all those people who have given me their heartfelt co-operation in making this project a success.

I take this opportunity to express my deep and sincere gratitude to the project guide [Dr. D.RAMESH](#), Assistant Professor, for her/his valuable advices at every stage of this work. I feel that without his/her supervision and many hours of devoted guidance, stimulating and constructive criticism, and this would never have Crone out in this form.

I greatly indebted to Dr. B. RAMA, HEAD, Department of Computer Science, Kakatiya University.

I am thankful to Dr. B. Rama, Dr. B. Manjula, Dr. D. Ramesh, Dr. T.Rajani Devi ,Ch.Neelima and Saloni Fathima, for their valuable support and providing the excellent facilities, motivation and valuable guidance throughout the project work. With their co-operation and encouragement, I have completed the project work in time.

Last but not least I would like to express my deep sense of gratitude and earnest thanks giving to my dear parents for their moral support and heartfelt co-operation in doing the project. I would also like to thank all the non-teaching staff and my friends, whose direct or indirect help enabled me to complete this work successfully.

JAMMULA SAI KUMAR (24000I1003)

ABSTRACT

Traditional knowledge-based authentication, such as text-based passwords, faces several well-known problems. Users generally choose passwords that are easy to remember, like names, birthdays, or simple patterns. However, these memorable passwords are often predictable and easy to guess by attackers. On the other hand, passwords that are assigned by the system (random and strong) tend to be difficult for users to remember and reuse.

This creates a conflict between security and usability in authentication systems. To address this issue, the paper focuses on the evaluation of the **Cued Click Points (CCP)** system

We propose and examine the usability and security of Cued Click Points (CCP), a cued-recall graphical password technique. Users click on one point per image for a sequence of images. The next image is based on the previous click-point. We present the results of an initial user study which revealed positive results. Performance was very good in terms of speed, accuracy, and number of errors. Users preferred CCP to Pass Points (Wiedenbeck et al., 2005), saying that selecting and remembering only one point per image was easier, and that seeing each image triggered their memory of where the corresponding point was located. We also suggest that CCP provides greater security than Pass Points because the number of images increases the workload for attackers.

CONTENTS	Page No
1. INTRODUCTION	01-07
1.1 Existing System	
1.2. Proposed System	
1.3. System Specification	
1.3.1 Hardware Requirements	
1.3.2 Software Requirements	
2. LITERATURE SURVEY	8-10
3. SYSTEM ANALYSIS	11-24
3.1. Modules	
3.2. Preliminary Investigation	
3.2.1 Request classification	
3.2.2 Feasibility Analysis	
3.2.3Request Approval	
3.3 Feasibility Study	
4. SYSTEM DESIGN & DEVELOPMENT	25-31
4.1 Input design	
4.2 Output design	
4.3 Use Case diagram	
4.4 Sequence diagram	
4.5 System Architecture	
5. METHODOLOGY	32-35
6. SYSTEM IMPLEMENTATION	36-38
6.1 Python	
6.2 Libraries used	
7. TESTING	39-42
7.1 Type of Testing	
8. PROGRAMMING CODE	43-59
8.1 Source code	
8.2 Output Screenshots	
9. CONCLUSION	60
10.REFERENCES	61-62

CHAPTER – 1

INTRODUCTION

Various graphical password schemes [14] have been proposed as alternatives to text-based passwords. Research and experience have shown that text-based passwords are fraught with both usability and security problems that make them less than desirable solutions [21]. Psychology studies have revealed that the human brain is better at recognizing and recalling images than text [8]; graphical passwords are intended to capitalize on this human characteristic in hopes that by reducing the memory burden on users, coupled with a larger full password space offered by images, more secure passwords can be produced and users will not resort to unsafe practices in order to cope. In this paper, we propose a new click-based graphical password scheme called CuedClickPoints (CCP). It can be viewed as a combination of PassPoints [19,20], Passfaces [9], and Story [5]. A password consists of one click-point per image for a sequence of images. The next image displayed is based on the previous click-point so users receive immediate implicit feedback as to whether they are on the correct path when logging in. CCP offers both improved usability and security. We conducted an in-lab user study with 24 participants and a total of 257 trials. Users had high success rates, could quickly create and re-enter their passwords, and were very accurate when entering their click-points. Participants rated the system positively and indicated that they preferred CCP to a PassPoints-style system. They also said that they appreciated the immediate implicit feedback telling them whether their latest click-point was correctly entered.

A preliminary security analysis of this new scheme is also presented. Hotspots (i.e., areas of the image that users are more likely to select) are a concern in click-based passwords [6,16], so CCP uses a large set of images that will be difficult for attackers to obtain. For our proposed system, hotspot analysis requires proportionally more effort by attackers, as each image must be collected and analyzed individually. CCP appears to allow greater security than PassPoints; the workload for attackers of CCP can be arbitrarily increased by augmenting the number of images in the system. As with most graphical passwords, CCP is not intended for environments where shoulder-surfing is a serious threat.

1.1 EXISTING SYSTEM

Graphical password schemes can be divided into three major categories based on the type of activity required to remember the password: recognition, recall, and cued recall. Recognition is the easiest for human memory whereas pure recall is most difficult since the information must be accessed from memory with no triggers. Cued recall falls somewhere between these two as it offers a cue which should establish context and trigger the stored memory. Among existing graphical passwords, CCP most closely resembles aspects of Pass faces, Story, and Pass Points Conceptually, CCP is a mix of the three; in terms of implementation, it is most similar to Pass Points. It also avoids the complex user training requirements found in a number of graphical password proposals, such as that of Weinshall. Passfaces is a graphical password scheme based primarily on recognizing human faces. During password creation, users select a number of images from a larger set. To log in, users must identify one of their pre-selected images from amongst several decoys. Users must correctly respond to a number of these challenges for each login. Davis et al implemented their own version called Faces and conducted a longterm user study. Results showed that users could accurately remember their images but that user-chosen passwords were predictable to the point of being insecure. Davis et al. proposed an alternative scheme, Story that used everyday images instead of faces and required that users select their images in the correct order. Users were encouraged to create a story as a memory aid. Faces for memorability, but user choices were much less predictable. The idea of click-based graphical passwords originated with Blonder who proposed a scheme where a password consisted of a series of clicks on predefined regions of an image. Later, Wiedenbeck et al proposed Pass Points, wherein passwords could be composed of several (e.g., 8) points anywhere on an image. They also proposed a “robust discretization” scheme, with three overlapping grids, allowing for login attempts that were approximately correct to be accepted and converting the entered password into a cryptographic verification key. Wiedenbeck et al. examined the usability of Pass Points in three separate in-lab user studies to compare text passwords to Pass Points, test whether the choice of image impacted usability, and determine the minimum size of the tolerance square. The overall conclusion was that Pass Points was a usable authentication scheme.

1.2 PROPOSED SYSTEM

Graphical passwords allow users to click on certain areas of the screen that are then converted by the computer to be used for authentications.

Picture Password

User is presented with a grid of pictures (photographs) or segments of a single picture, user clicks on a sequence of pictures each segment of the picture grid is associated with a value matrix.

Current authentication methods can be divided into three main areas:

- I. Token based authentication**
- II. Biometric based authentication**
- III. Knowledge based authentication**

Token based techniques, such as key cards, bank cards and smart cards are widely used. Many token-based authentication systems also use knowledge based techniques to enhance security. For example, ATM cards are generally used together with a PIN number. Biometric based authentication techniques, such as fingerprints, iris scan, or facial recognition, are not yet widely adopted. The major drawback of this approach is that such systems can be expensive, and the identification process can be slow and often unreliable. However, this type of technique provides the highest level of security. Knowledge based techniques are the most widely used authentication techniques and include both text-based and picture-based passwords. The picture-based techniques can be further divided into two categories: recognition-based and recall-based graphical techniques.

III(i) Recognition Based Techniques

Using recognition-based techniques, a user is presented with a set of images and the user passes the authentication by recognizing and identifying the images he or she selected during the registration stage. Using recall-based techniques, a user is asked to reproduce something that he or she created or selected earlier during the registration stage. We proposed a graphical password mechanism for mobile devices. During the enrolment stage, a user selects a theme (e.g. sea, cat, etc.) which consists of thumb nail photos and then registers a sequence of images as a password. During

the authentication, the user must enter the registered images in the correct sequence. One drawback of this technique is that since the number of thumbnail images is limited to 30, the password space is small.

III(ii) Cued Click Points

Cued Click Points (CCP) is a proposed alternative to Pass Points. In CCP, users click one point on each of $c = 5$ images rather than on five points on one image. It offers cued-recall and introduces visual cues that instantly alert valid users if they have made a mistake when entering their latest click-point (at which point they can cancel their attempt and retry from the beginning). It also makes attacks based on hotspot analysis more challenging, as we discuss later. As shown in Figure 1, each click results in showing a next-image, in effect leading users down a “path” as they click on their sequence of points. A wrong click leads down an incorrect path, with an explicit indication of authentication failure only after the final click. Users can choose their images only to the extent that their click-point dictates the next image. If they dislike the resulting images, they could create a new password involving different click-points to get different images. We envision that CCP fits into an authentication model where a user has a client device (which displays the images) to access an online server (which authenticates the user). We assume that the images are stored server-side with client communication through SSL/TLS. For further discussion, see Section 6. For implementation, CCP initially functions like Pass Points. During password creation, a discretization method is used to determine a click-point’s tolerance square and corresponding grid. For each click-point in a subsequent login attempt, this grid is retrieved and used to determine whether the click point falls within tolerance of the original point. With CCP, we further need to determine which next-image to display. Similar to the Pass Points studies, our example system had images of size 451x331 pixels and tolerance squares of 19x19 pixels. If we used robust discretization.

we would have 3 overlapping candidate grids each containing approximately 400 squares and in the simplest design, 1200 tolerance squares per image (although only 400 are used in a given grid). We use a function $f(\text{username}, \text{current Image}, \text{current Tolerance Square})$ that uniquely maps each tolerance square to a next-image. This suggests a minimum set of 1200 images required at each stage. One argument against

using fewer images, and having multiple tolerance squares map to the same next-image, is that this could potentially result in misleading implicit feedback in (albeit rare) situations where users click on an incorrect point yet still see the correct next-image. Each of the 1200 next-images would have 1200 tolerance squares and thus require 1200 next-images of their own. The number of images would quickly become quite large. So, we propose re-using the image set across stages. By re using images, there is a slight chance that users see duplicate images. During the 5 stages in password creation, the image indices i_1, \dots, i_5 for the images in the password sequence are each in the range $1 \leq i_j \leq 1200$. For a user's initial image is selected by the system based on some user characteristic (as an argument to f above; we used username). The sequence is re-generated on-the-fly from the function each time a user enters the password. If a user enters an incorrect click-point, then the sequence of images from that point onwards will be incorrect and thus the login attempt will fail.

For an Attacker who does not know the correct sequence of images, this cue will not be helpful. We expect that hotspots will appear as in Pass Points, but since the number of images is significantly increased, analysis will require more effort which increases proportionally with the configurable number of images in the system. For example, if attackers identify thirty likely click-points on the first image, they then need to analyse the thirty corresponding second images (once they determine both the indices of these images and get access to the images themselves), and so on, growing exponentially. A major usability improvement over PassPoints is the fact that legitimate users get immediate feedback about an error when trying to log in. When they see an incorrect image, they know that the latest click-point was incorrect and can immediately cancel this attempt and try again from the beginning. The visual cue does not explicitly reveal "right" or "wrong" but is evident using knowledge only the legitimate user should possess. As with text passwords, PassPoints can only safely provide feedback at the end and cannot reveal the cause of error. Providing explicit feedback in PassPoints before the final click-point could allow PassPoints attackers to mount an online attack to prune potential password subspaces, whereas CCP's visual cues should not help attackers in this way. Another usability improvement is that being cued to recall one point on each of five images appears easier than remembering an ordered sequence of five points on one image.

1.3 SYSTEM SPECIFICATIONS

1.3.1 HARDWARE REQUIREMENTS

Processor	:	Intel i3 or equivalent and above
RAM	:	Minimum 4 GB
Hard Disk	:	Minimum 100 MB free space
Display	:	1366x768 resolution or higher
Input Devices	:	Keyboard, Mouse (for point selection)

1.3.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows / Linux (Ubuntu recommended)
Programming Language	:	Python 3.8 or above
GUI Library	:	Tkinter (Python standard GUI library)
Image Handling	:	PIL (Python Imaging Library - Pillow)
File Operations	:	OS, shutil modules
Data Storage	:	JSON or local file-based system

CHAPTER – 2

LITERATURE SURVEY

LITERATURE SURVEY

The research paper titled **"Graphical passwords: Learning from the first twelve years"** by Biddle, Chiasson, and Van Oorschot is a comprehensive survey of graphical password systems and their evolution over the past twelve years. The paper aims to provide a critical evaluation of graphical password authentication, its advantages, disadvantages, and potential for future development. The research paper **"Is a Picture Really Worth a Thousand Words? Exploring the Feasibility of Graphical Authentication Systems"** by De Angeli, Coventry, Johnson, Renaud, and Sassoon, published in the International Journal of Human-Computer Studies in 2005, focuses on the feasibility of using graphical authentication systems as an alternative to traditional text-based passwords. The paper presents the results of a usability study comparing two graphical authentication systems, Passfaces and Persuasive Cued Click Points (PCCP), with a text-based password system. The research paper **"Pass-Image: A user-friendly graphical password system"** by Gao and Liu, published in the journal Computer Standards & Interfaces in 2014, proposes a new graphical password authentication system called Pass-Image. The paper presents the design, implementation, and evaluation of the Pass Image system, which is designed to be more user-friendly and secure than existing graphical password systems. The research paper **"The design and analysis of graphical passwords"** presents an analysis of graphical passwords as an alternative to traditional alphanumeric passwords. The paper examines the usability and security implications of graphical passwords and proposes a framework for designing and analysing such systems. The authors also present a novel graphical password scheme called Passfaces, which uses facial images as authentication tokens. The scheme is evaluated in terms of its security against various attacks, including guessing, shoulder-surfing, and dictionary attacks.

The research paper **"A survey of visual password schemes"** by Li, Yang, and Xu provides an overview of the existing visual password schemes and evaluates their security, usability, and memorability. The authors begin by describing the need for alternative authentication methods and introduce the concept of visual passwords. They then review and classify various visual password schemes, including recall-based schemes, recognition-based schemes, and cued-recall-based schemes.

Password authentication has been widely studied in the field of computer security and privacy. Traditional text based passwords have been widely used, but they have several disadvantages, such as password fatigue, password reuse, and weak passwords. To address these issues, researchers have proposed various alternative authentication methods, including graphical password authentication. This method has been found to be more natural and intuitive for users, reducing the likelihood of password fatigue, password reuse, and weak passwords. In conclusion, the literature supports the use of graphical password authentication as a more natural and intuitive alternative to text-based passwords. The use of SSIM algorithms has been found to be effective in comparing the user's drawn pattern with the previously stored pattern. Future studies could investigate the use of additional algorithms or features, such as gesture

CHAPTER – 3

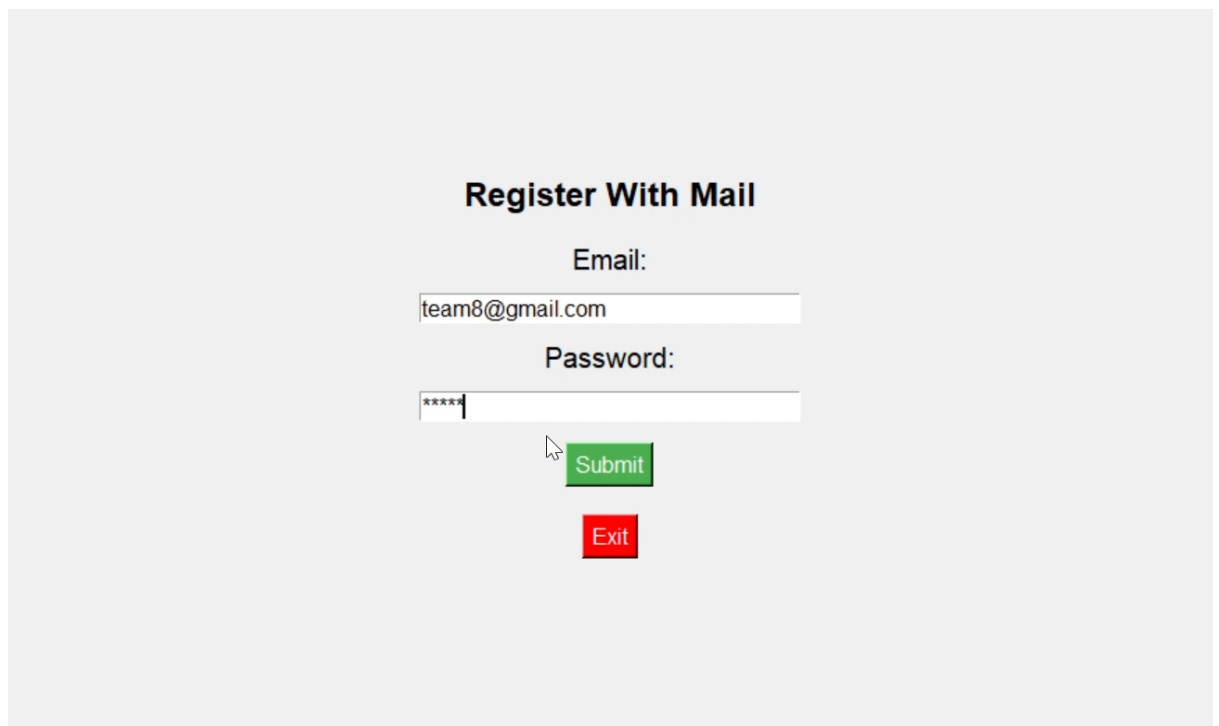
SYSTEM ANALYSIS

3.1 MODULES

1. User Registration Module :

This module handles the process of creating a new user account. Users are required to input their email address and create a text-based password. Once the credentials are validated (e.g., ensuring email format is correct and fields are not empty), they are stored securely in a local storage file, typically a JSON file.

- After successful registration, the user is immediately directed to the graphical password setup.
- This module ensures only valid and unique users are registered into the system.
- All data is stored locally in a structured format for easy retrieval.



The image shows a registration form titled "Register With Mail". It contains two input fields: "Email:" with the value "team8@gmail.com" and "Password:" with masked characters "*****". Below the fields are two buttons: a green "Submit" button and a red "Exit" button. A mouse cursor is pointing at the "Submit" button.

Fig 1.1 : Registration Module

2.Graphical Password Setup Module

After the user registers, they proceed to this module to create their **graphical password**.

- The user is shown **five predefined images**, one by one.
- For each image, the user selects **one specific point (cued click)** that acts as part of their graphical password.
- The selected coordinates and image names are saved locally (e.g., in a JSON file).
- Each image is presented in a full screen window with a **blue border** and visual feedback (a **red dot**) showing where the user clicked.
- This process enhances security by relying on visual memory and spatial recognition.

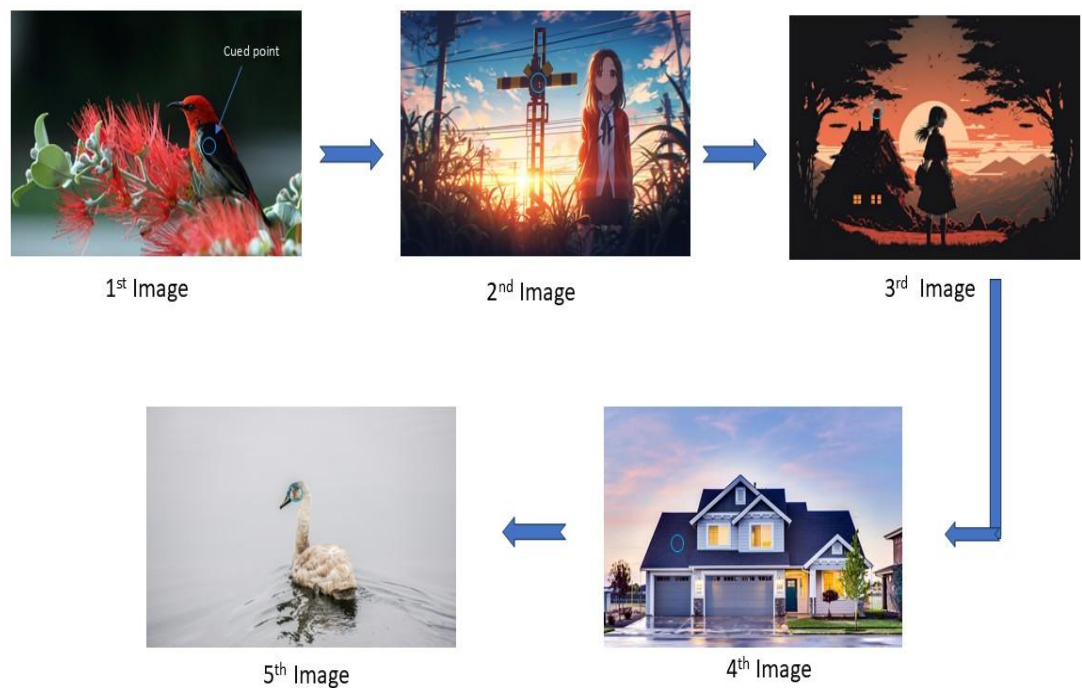
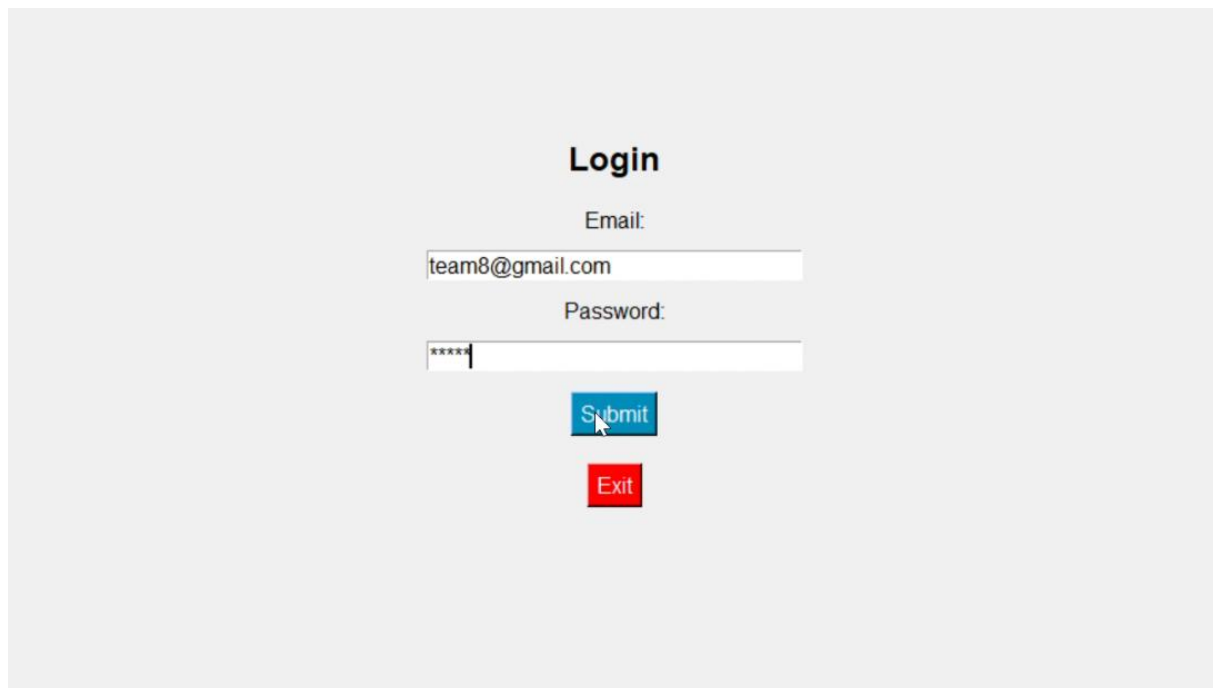


Fig 2.1 : Image Selection Module

3. User Login Module

This module allows existing users to log in using their **email and text-based password**.

- On submission, the entered credentials are checked against the stored data.
- If credentials match, the system proceeds to the **graphical authentication phase**.
- Invalid attempts prompt appropriate error messages without revealing sensitive details.



The screenshot displays a login interface on a light gray background. At the top center, the word "Login" is written in a bold, black, sans-serif font. Below this, the label "Email:" is positioned to the left of a white text input field containing the email address "team8@gmail.com". Further down, the label "Password:" is to the left of another white text input field, which is filled with six asterisks "*****". Below the password field, there are two buttons: a blue button with the word "Submit" in white text, and a red button with the word "Exit" in white text, both centered horizontally. A mouse cursor is visible over the "Submit" button.

Fig 3.1 : Login Module

4. Graphical Authentication Module

This module serves as the core of the authentication mechanism.

- The same five images used during setup are shown again, one at a time.
- The user is required to **click on the same cued points** that were selected during registration.
- The system checks whether the clicked points are within an **acceptable threshold** distance from the original coordinates (to account for human precision).
- Upon successfully matching all five points, the user is granted access.
- Visual indicators (red dot for click, blue border for image) help guide the user during the process.

5. Data Storage & Retrieval Module

This module manages all data-related operations in the system.

- Stores user information including email, text password, selected image names, and cued coordinates.
- Uses **JSON format** or a similar structured approach for easy access and modification.
- Handles reading and writing operations securely.
- Ensures data integrity and prevents unauthorized tampering by separating authentication logic from storage logic.

5. File/Folder Lock & Unlock Module

This module provides secure access to a **predefined file or folder** after successful login.

- Once authenticated, the user is taken to a simple GUI with options to **unlock a specific file or folder**.
- The system performs the unlocking operation automatically, without requiring manual file browsing.
- This makes the application useful for protecting sensitive files and folders using an intuitive graphical password mechanism.

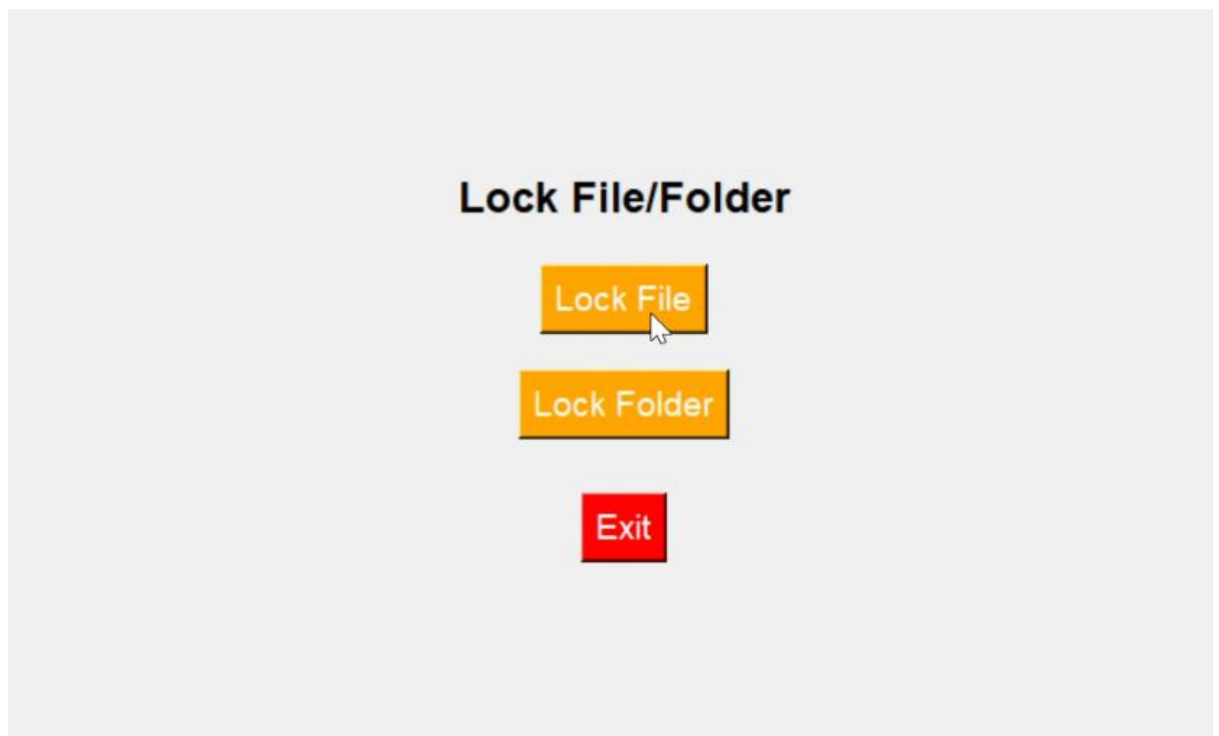


Fig 2.1 : Locking Module

3.2 PRELIMINARY INVESTIGATION

In today's digital era, safeguarding sensitive information and resources has become a top priority. Most systems rely on **text-based passwords**, which, despite being widely used, come with a range of vulnerabilities. Many users choose weak, easy-to-remember passwords, or reuse the same credentials across multiple platforms, making them vulnerable to attacks like **brute-force**, **dictionary attacks**, **phishing**, and **keylogging**. Additionally, users often struggle to remember complex passwords, leading to frustration and potential security risks due to bad password practices.

To overcome these issues, the concept of **graphical password authentication** has emerged as a more secure and user-friendly alternative. Unlike traditional methods, graphical passwords use **images and spatial memory**, allowing users to authenticate by clicking on specific areas of an image. This technique takes advantage of the human brain's natural ability to remember visual information better than text.

3.2.1 REQUEST CLASSIFICATION

Every software project originates from a specific type of request or need. Understanding the classification of that request helps in identifying the motivation behind the system, the stakeholders involved, and the purpose it serves. The **Graphical Password Authentication System** falls under several relevant request categories, each explained below:

New System Development Request

This project is categorized as a **New System Development Request**, as it is being developed entirely from the ground up. There is no existing system or software being enhanced or modified. Instead, this application is designed from scratch to meet the specific goal of providing an innovative and secure authentication mechanism using **graphical passwords**. This request arises from the demand for a unique authentication method that enhances security beyond traditional text-based systems.

Security Enhancement Request

The main purpose of this project is to **enhance security** by introducing a new way of user authentication. Traditional passwords are vulnerable to various attacks like brute force, phishing, and keylogging. This project addresses those concerns by using **image-based password cues**, which are significantly more secure and harder to guess or replicate. Additionally, the system allows for **file/folder protection**, which is a critical feature for local data privacy. Thus, it serves as a response to the need for more robust personal security tools.

User Interface Improvement Request

Another classification this project falls under is the **User Interface Improvement** request. It offers a **simple, intuitive, and user-friendly GUI** that guides the user through the registration, graphical password setup, and authentication process smoothly. Instead of complex command-line interactions, it uses **visual design elements** like buttons, image borders, red cue dots, and fullscreen layouts to enhance the overall experience. This ensures even non-technical users can comfortably interact with the system.

Utility Tool Request

This project can also be viewed as a **utility software request**. The system doesn't just offer authentication but also provides practical functionality by **locking and unlocking files or folders**. It becomes a useful tool for users who want to protect personal or sensitive data on their computers. This utility nature makes it relevant for daily use and adds real-world value beyond academic interest.

Research and Academic Request

Lastly, this system also serves the purpose of a **research-based or academic request**. It is ideal for students, researchers, or developers who want to study and demonstrate alternative authentication methods. The use of cued click-points and image recognition falls within the realm of **experimental security systems**, which

makes the project suitable for final year academic submissions or cybersecurity research demonstrations.

3.2.2 FEASIBILITY ANALYSIS

Before developing any software project, it is essential to conduct a **feasibility analysis** to evaluate whether the project is practical, viable, and beneficial to proceed with. This process determines if the project can be successfully implemented with the available resources, technologies, and within a reasonable time frame. The feasibility analysis for the **Graphical Password Authentication System** is described in terms of four major aspects

1. Technical Feasibility

Technical feasibility assesses whether the current technology, tools, and resources can support the development and execution of the proposed system.

In this case, the **Graphical Password Authentication System** is highly technically feasible because:

- It is developed using **Python**, a widely supported, open-source programming language known for its flexibility and ease of use.
- The **Tkinter** library is used for creating the GUI, which is pre-installed with Python and does not require any external dependencies.
- The **Pillow** library is used for image handling, which is lightweight and easily installable.
- File and folder locking is managed using Python's **os** and **shutil** modules, eliminating the need for additional software.
- The system does not require advanced hardware or a server. It runs smoothly on any modern computer with basic specifications.

Thus, from a technical standpoint, the project is not only feasible but also efficient and adaptable.

2. Economic Feasibility

Economic feasibility determines whether the system can be developed and maintained within the available budget and whether the benefits justify the costs involved.

This system is **economically very feasible** due to the following reasons:

- There are **no licensing costs** since all tools and libraries used are open-source.
- Development can be done using **free resources** like Python, Tkinter, and community support.
- There are **no hardware investments** needed, as the system can run on existing computers or laptops.
- It does not require internet access, a server, or cloud infrastructure, further reducing costs.

In conclusion, the system is **cost-effective** and offers high value without financial overhead, making it ideal for students, individuals, or small organizations.

3. Operational Feasibility

Operational feasibility assesses how well the proposed system solves the problem and whether it can be used effectively by the target users.

This system is highly **operationally feasible** because:

- It offers a **simple and user-friendly GUI**, allowing even non-technical users to register, log in, and lock/unlock their data.
- The graphical password mechanism is **intuitive**, making it easier to remember and harder to guess than traditional passwords.
- Users receive clear feedback through visual cues (e.g., red dots for click points, blue image borders, and guided transitions).
- It fulfils a real-world need by providing **personal file/folder security**.

Hence, the system can be smoothly adopted and operated by its intended users without requiring any specialized training.

4. Behavioural and Social Feasibility

This aspect evaluates how the end-users and other stakeholders will respond to the system once implemented.

The **Graphical Password Authentication System** is **socially and behaviourally feasible** because:

- Users are becoming increasingly aware of digital security, and are more open to innovative solutions.
- The use of graphical passwords appeals to **visual learners** and provides a fresh user experience.
- It eliminates the stress of remembering complex text passwords, improving **user satisfaction**.
- As more users seek privacy in local file storage, this system addresses that concern effectively.

There are no major cultural, ethical, or behavioural obstacles anticipated with this project.

3.2.3 REQUEST APPROVAL

After conducting a thorough preliminary investigation, request classification, and a detailed feasibility analysis, the proposed project titled "Graphical Password Authentication System with File/Folder Protection" is found to be technically viable, economically feasible, operationally practical, and socially acceptable. The findings from all analysis phases clearly demonstrate that this project meets the identified security needs and offers a robust, innovative alternative to traditional password systems.

The core purpose of this project is to enhance data security through a graphical authentication mechanism that allows users to select specific points on images as passwords. Furthermore, the system allows users to lock and unlock files or folders using these image-based credentials, offering real-world utility and personal data protection.

This project is especially valuable in today's digital environment, where threats to user privacy and information integrity are increasing. Its simplicity, usability, and effectiveness make it suitable for both academic demonstration and practical implementation on personal devices. Considering all the above aspects, this project is hereby approved for development. The approval is granted based on the following grounds:

- The project introduces a unique, secure, and user-friendly authentication method.
- It serves as a relevant academic submission demonstrating real-world application of security concepts.
- All resources required for the development are open-source and readily available.
- The project addresses a genuine need for local file security and password innovation.
- No significant financial, technical, or operational barriers were identified.

This approval marks the official commencement of the design and development phase of the project. All future work will now proceed according to the defined objectives, planned modules, and technical specifications outlined in the initial documentation.

3.3 FEASIBILITY STUDY

A **Feasibility Study** is a crucial phase in the software development life cycle. It aims to evaluate whether the proposed system is **achievable**, **sustainable**, and **worth pursuing** based on technical, economic, operational, and social perspectives. Below is a comprehensive feasibility study for the **Graphical Password Authentication System**.

1. Technical Feasibility

This part determines whether the current technology is sufficient to implement the system successfully.

The **Graphical Password Authentication System** is technically feasible because:

- It uses **Python**, a high-level, open-source programming language that is ideal for rapid development.
- The system's **GUI** is developed using **Tkinter**, which is lightweight and part of the Python standard library.
- Image handling and click-point functionality are implemented using the **Pillow (PIL)** library and basic Python modules like `os` and `shutil`.
- The system is designed to run **locally**, which means no server or internet connection is required.
- It can function on **basic hardware** setups, such as standard laptops or desktops running Windows/Linux.

Conclusion: The required technology and resources are available and sufficient. No advanced hardware or third-party services are needed, making it technically sound and low-risk.

2. Economic Feasibility

This evaluates whether the cost of developing and maintaining the system is justified by the benefits it delivers.

- All tools and libraries used (Python, Tkinter, Pillow) are **free and open-source**.
- No need for **paid servers, APIs, or cloud infrastructure**.
- The development can be completed on a **single system** with no additional hardware cost.
- The solution enhances local security, potentially **preventing costly data breaches** or unauthorized access.

Conclusion: With **zero development cost** and significant real-world utility, the system is highly economical and delivers **excellent return on investment (ROI)** for individuals or small organizations.

3. Operational Feasibility

This focuses on how well the proposed solution solves the problem and whether it can be used effectively by its intended users.

- The system has an intuitive **user-friendly interface**, making it easy for users to register, select images, click cued points, and authenticate themselves.
- Unlike complex cryptographic methods, this solution uses **simple visual interactions**.
- Visual feedback (e.g., red dots for clicks, blue borders for image selection) guides the user throughout the process.
- The system also supports **secure file and folder access**, making it practical and not just theoretical.

Conclusion: The system is operationally feasible. It is easy to use, effective in solving the intended problem, and highly relevant for individuals who value local data protection.

4. Social/Behavioural Feasibility

This evaluates how the system will be received by users and whether they will adapt to the new method.

- Graphical passwords offer a more **engaging and memorable experience** compared to traditional text passwords.
- Users are increasingly aware of **cybersecurity threats**, making them more open to trying **secure and innovative** authentication methods.
- There is **no social or ethical resistance** to the system, as it respects user privacy and enhances personal data protection.

Conclusion: The system aligns well with modern user behaviour and expectations. It is socially acceptable and would likely gain positive reception from its target users.

CHAPTER 4

SYSTEM DESIGN AND DEVELOPMENT

4.1 INPUT DESIGN

The input design of this project plays a vital role in ensuring that the system receives accurate and reliable data in a user-friendly and secure manner. Since the system deals with sensitive operations such as user registration, authentication, and file/folder protection, the inputs are carefully structured to support both usability and security.

During the user registration phase, the system requires the user to enter an email ID and a password. The email input is a standard text field, while the password input is masked for privacy. Once these credentials are submitted, the user is directed to set up a graphical password. This graphical password is created by selecting five images from a set provided by the system. For each selected image, the user is required to click on a specific point on the image which serves as the cued point. These coordinates are securely stored for future authentication.

In the login phase, the user is prompted to enter their registered email ID. After entering the email, the system automatically presents the five previously selected images one by one in the same order as during registration. The user must then click on the exact cued point on each image. The system compares the new click positions with the stored coordinates and validates the input within a certain tolerance range to account for slight variations in clicking. If all five points are verified correctly, the user is authenticated successfully.

After successful authentication, the user is taken to a screen where they can choose to lock either a file or a folder. This is done using a simple selection mechanism such as radio buttons or clickable icons labelled appropriately. Once the choice is made, the system automatically locks the selected item without requiring the user to enter any further input.

To unlock a protected file or folder, the system prompts the user for graphical authentication again. The five images appear in sequence, and the user must click the original points as before. If all points match, the file or folder is unlocked instantly, and access is granted. No additional input is required during this stage, as the system verifies access solely based on the graphical password.

Throughout the application, the input design emphasizes clarity, accuracy, and ease of use. Error messages are displayed if invalid or incomplete data is entered. Visual feedback such as red dots on click points and blue borders around images ensure that users are aware of their interactions with the system. All inputs are processed securely and are validated to prevent unauthorized access or data breaches.

This input design ensures that the application remains intuitive for users while maintaining a high level of security and efficiency.

4.2 OUTPUT DESIGN

The output design of this system plays a crucial role in providing users with clear, meaningful feedback throughout their interaction with the application. It ensures that users are informed of their actions' results, such as successful registration, authentication, file/folder locking, or unlocking.

Once a user completes the registration process, the system displays a confirmation message indicating that the registration was successful. This feedback reassures the user and provides direction to proceed to the login page. The success message is displayed on a well-structured GUI, styled with clear fonts and colours for better visibility.

During login, after the user enters their email and attempts graphical authentication by clicking on the cued points on each image, the system verifies these inputs. If the inputs match the stored graphical password, the system provides a success message such as "Login Successful." This message may be followed by a transition to the next step, such as the file/folder locking interface. If the input does not match, the system displays an error message like "Incorrect click points. Please try again." The output messages are visually distinguished using colours like green for success and red for errors, helping users understand outcomes easily.

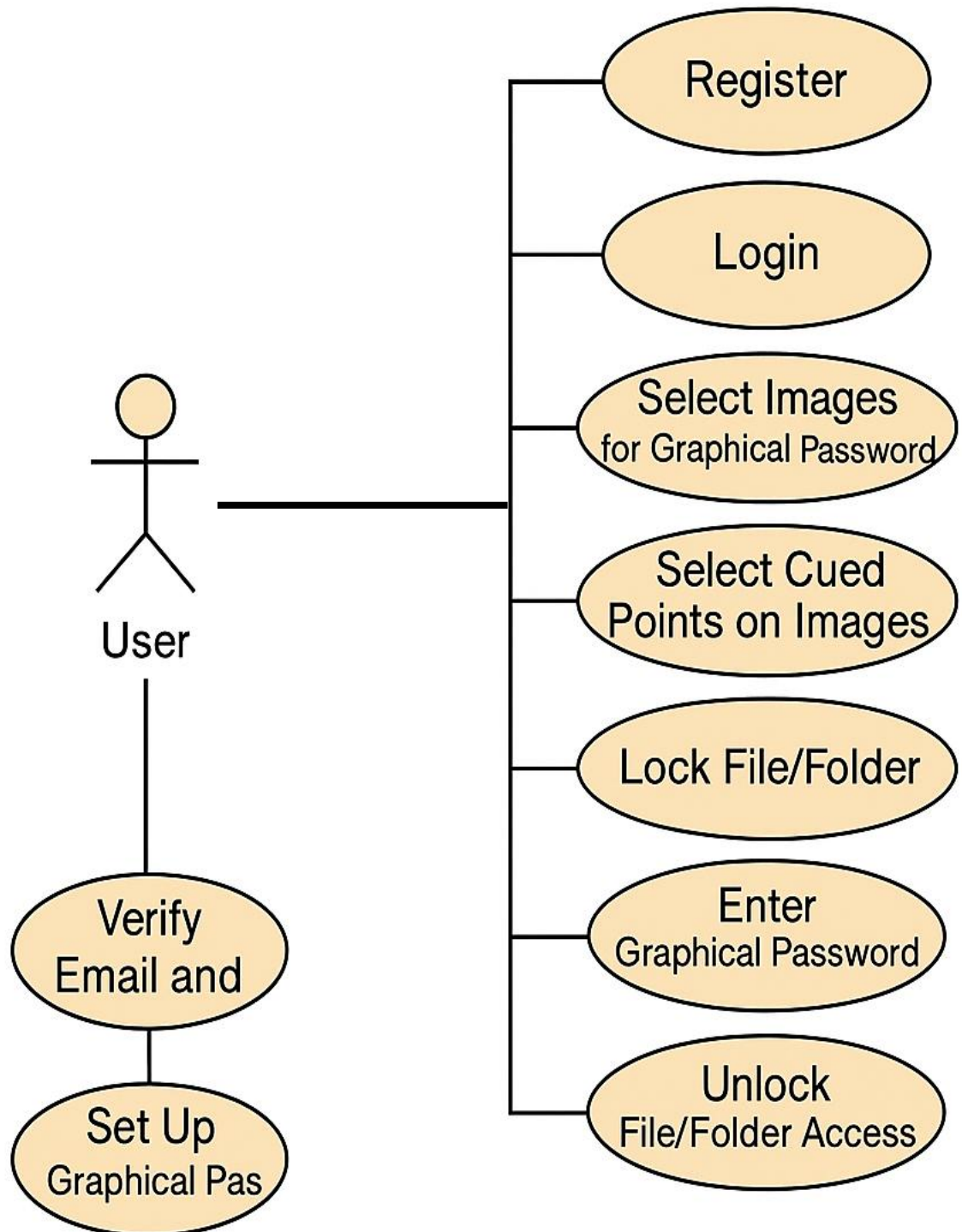
When the user successfully locks a file or folder, the system immediately confirms the action with a message like "Folder has been locked successfully." This is shown on the same screen without refreshing or disrupting the interface flow. Similarly,

when the user unlocks a previously protected file or folder after successful authentication, the system responds with a confirmation such as “Folder unlocked. You can now access your files.”

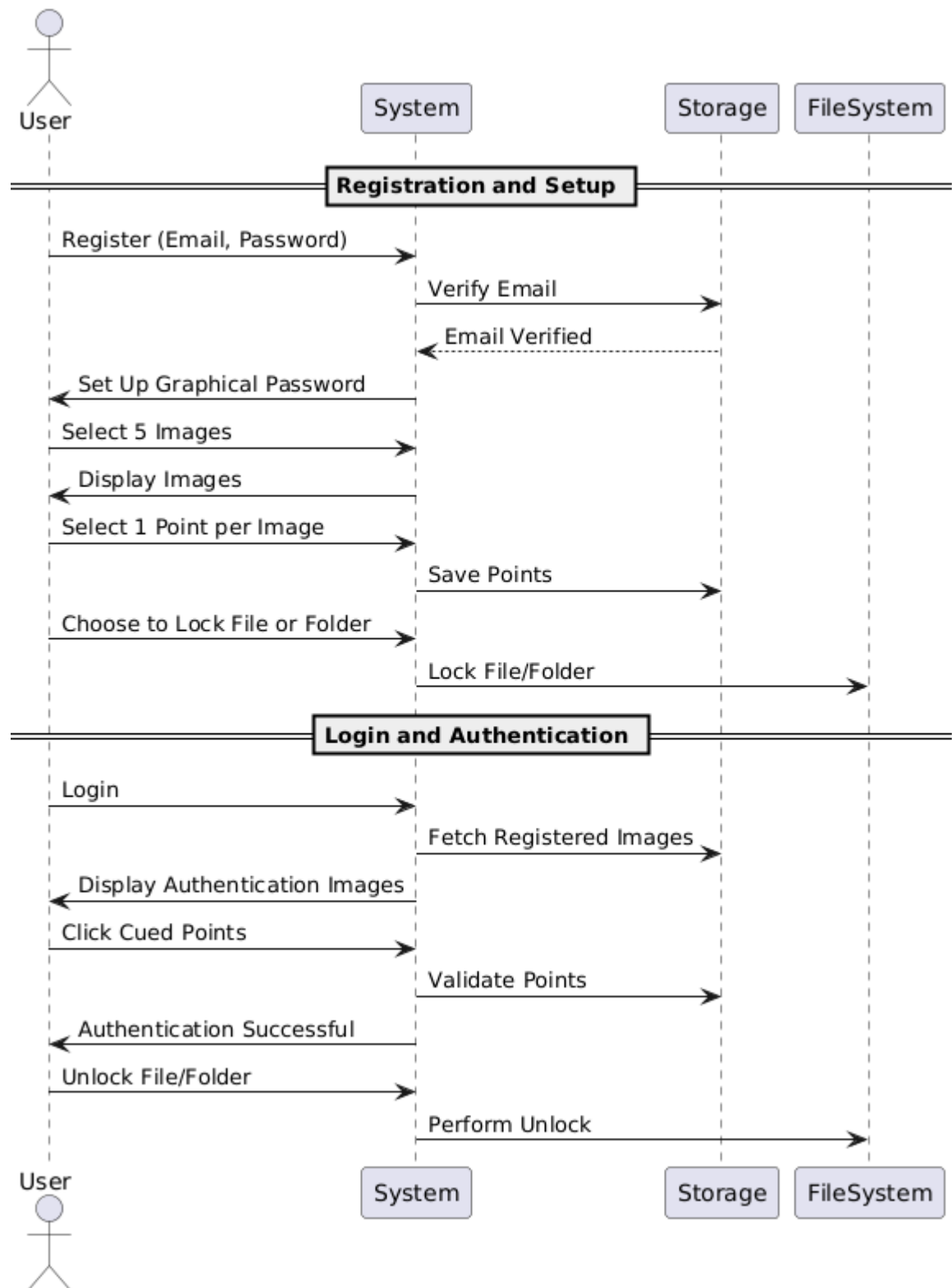
These outputs are not only textual but are also supported by visual cues, such as border colours, message boxes, and optional icons to enhance clarity. In some versions of the system, an additional message like “HELLO, HOW ARE YOU?” may be displayed as a greeting after login, giving a personalized touch to the interface.

The overall output design of the system is focused on clarity, simplicity, and providing users with immediate, understandable responses to their actions. This reduces confusion, enhances user satisfaction, and ensures a smooth flow through the system’s functions.

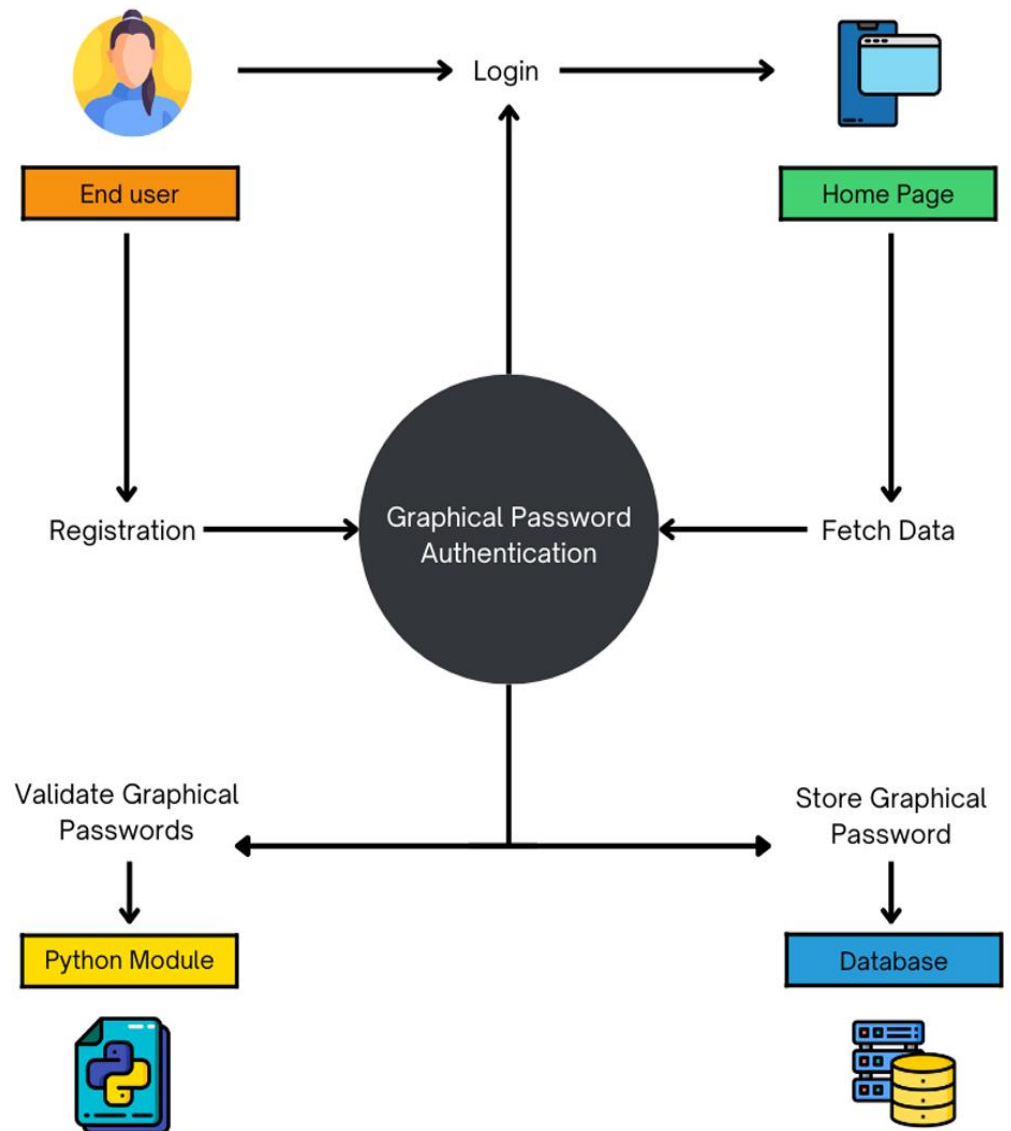
4.3 USE CASE DIAGRAM



4.4 SEQUENCE DIAGRAM



4.5 SYSTEM ARCHITECTURE



CHAPTER 5

METHODOLOGY

To build a secure and user-friendly Graphical Password Authentication System with file and folder locking functionality, a structured step-by-step approach was followed throughout the development of the project. This methodology is designed to ensure a smooth user experience, reliable authentication, and effective protection of sensitive data.

Understanding the Project Requirements

The first step involved identifying the core goal of the project — to create an authentication system that goes beyond traditional text-based passwords. The focus was on improving security and user experience by using graphical images and cued-point selection for login. Additionally, the system needed to lock and unlock specific files or folders after a successful login.

Designing the User Flow and Features

Once the concept was clear, the overall user flow was designed. The application was structured to have a simple yet effective interface with three main options: Register, Login, and Exit.

- Register allows new users to sign up by entering basic credentials like an email and password.
- After registration, users are guided through setting a graphical password by selecting specific points on five images.
- Login allows users to authenticate themselves using the graphical password they created.
- Upon successful authentication, the system provides options to unlock a locked file or folder.

This user journey was carefully planned to be intuitive, especially for users unfamiliar with graphical password systems.

Selecting Tools and Technologies

The application was developed using Python, due to its simplicity and powerful libraries. Tkinter was used to create the graphical user interface, as it offers the

flexibility needed to design custom layouts, handle user interactions, and manage multiple windows.

Built-in Python modules like `os`, `shutil`, and `pickle` were used to handle file operations and securely store user data (including credentials and graphical password coordinates).

Graphical Password Setup

During registration, users are shown five different images, one at a time. For each image, they are asked to click on a point of their choice — this point becomes part of their graphical password.

To help users remember their selected spots:

- A red dot appears at the clicked point.
- Each image is displayed with a blue border and is shown in its original size within a centered window.
- Images are displayed one after another in a clean, full screen layout to keep the user focused.

The coordinates of each selected point (relative to the image) are stored securely for use during authentication.

Authentication Using Graphical Passwords

When the user chooses to log in:

- The same five images are shown again, one at a time, but in a random order.
- The user is required to click on the same points they selected during registration.
- A tolerance range is defined around each saved coordinate to allow minor variations in clicking accuracy.

If the user clicks within the correct area for all five images, authentication is considered successful.

Locking and Unlocking Files/Folders

After a successful login, the user is taken to a new interface with two buttons:

- Unlock File
- Unlock Folder

These buttons are linked to predefined files or folders that were previously "locked" (e.g., made inaccessible or moved to a secure location). Clicking the appropriate button makes the selected item accessible again, giving the user control over their private content.

This feature is useful for securing sensitive documents or directories on the user's computer.

User Interface Design

Special attention was given to UI design. The entire application maintains a consistent brown background (#8B4513), modern fonts, and clear navigation buttons. The full screen mode ensures that the user stays engaged during both registration and login processes.

All UI components from the title to buttons and images are centre-aligned, and transitions between screens are smooth, without opening multiple windows that could confuse the user.

CHAPTER 6

SYSTEM IMPLEMENTATION

PYTHON

Python served as the core programming language for developing the entire graphical password authentication system. Its simplicity, readability, and rich ecosystem of libraries made it ideal for implementing both the backend logic and the graphical user interface. Python was used to build the GUI using the Tkinter library, which provided a robust and customizable interface for user interactions such as registration, login, and image-based password selection. It also handled essential backend operations like storing user credentials and graphical password data using the pickle module for serialization. Furthermore, Python's built-in libraries such as os and shutil were used to implement file and folder locking mechanisms after successful authentication. The language's flexibility allowed easy integration of image handling, coordinate matching, and secure storage functionalities into a cohesive and user-friendly desktop application.

LIBRARIES USED

1. TKINTER

Tkinter is used to create the graphical user interface (GUI) of the application. It enables the design of various screens such as the registration, login, and file unlocking interfaces. Tkinter helps manage user inputs and display images with interactive elements like buttons, labels, and canvases.

2. OS

The os module is used to interact with the operating system, particularly for managing file and folder paths. It checks if specific paths exist, changes file permissions, and modifies access to locked files or directories based on user authentication.

3. SHUTIL

Shutil allows for high-level file operations like copying, moving, and deleting files. In this project, it's used for unlocking files or folders by moving or changing their visibility, ensuring that users can access protected resources after successful authentication.

4. **PICKLE**

Pickle is used to serialize and deserialize Python objects, allowing the storage and retrieval of user credentials and graphical password data. It securely saves the email, password, image names, and click points locally, so they can be accessed during the authentication process.

5. **RANDOM**

The random module is utilized to shuffle the order of images during the authentication process. This ensures the images are presented in a random sequence each time, preventing predictable patterns and enhancing security.

6. **MATH**

The math module is used to calculate the Euclidean distance between the stored and current click coordinates. This helps determine if the clicked points are within an acceptable range of the registered points during the login process.

7. **PIL**

Pillow (PIL) is an image processing library used for tasks such as resizing, cropping, and converting image form

CHAPTER 7

TESTING

1. Unit Testing

Unit testing involves checking each individual component of the application separately to ensure it works as expected. In your project, unit testing was used to validate the core functionalities like storing user credentials, capturing click points on images, matching the graphical password during login, and executing the file or folder unlocking logic. Each of these functionalities was tested independently. For instance, the correctness of storing email and password using the pickle module was verified, as well as whether the coordinates of user clicks were recorded accurately. The point matching algorithm, based on Euclidean distance, was tested to confirm it accepted points only within a defined threshold. Similarly, the unlocking mechanism was tested to make sure it triggered only after successful authentication.

2. Integration Testing

Integration testing ensures that all the separate modules of the application work together smoothly. In this project, the process of moving from user registration to login and finally to unlocking a file or folder was tested in a connected flow. This involved checking that the graphical password created during registration was correctly retrieved during login and used to verify the user's identity. Additionally, the system was tested to confirm that once the authentication succeeded, the options to unlock a specific file or folder became available. It also ensured that invalid data or failed authentication did not crash the application and that appropriate error messages were shown.

3. System Testing

System testing was conducted to evaluate the entire application as a single, unified system. This tested the end-to-end scenario from opening the app to registering a user, logging in with graphical authentication, and finally unlocking a protected file or folder. It ensured that the user interface displayed properly across all screens, that windows opened and closed correctly, and that the full workflow functioned seamlessly. Any system-level issues, such as freezing, unresponsive buttons, or window misbehaviour, were also identified and corrected during this testing phase.

4. Usability Testing

Usability testing was carried out to assess how user-friendly and intuitive the application is. This involved evaluating the design and layout of the graphical user interface built with Tkinter, checking if users could easily understand and navigate through the steps of registration, login, and unlocking. It included testing the alignment of buttons, the readability of text, the colour contrast (especially the brown background colour used throughout), and the clarity of instructions. Feedback visuals, such as the red dot appearing after image clicks and confirmation messages for successful or failed login, were tested for their effectiveness in guiding users.

5. Security Testing

Security testing focused on ensuring that the system provided adequate protection against unauthorized access. The authentication mechanism was rigorously tested to ensure that only users who clicked on the correct cued points could gain access. The application was also tested to confirm that user credentials and graphical password data were securely stored and not easily tampered with or accessed by external users. File and folder unlocking features were tested to ensure they only functioned after valid authentication, preventing any direct access without passing through the security layer.

6. Edge Case Testing

Edge case testing explored how the system would behave under unusual or incorrect usage scenarios. This included testing what happens if a user submits empty fields during registration or login, or clicks on the wrong points during image authentication. The system was also tested to see how it handles duplicate registrations, missing image files, and other unexpected user actions. These tests helped ensure that the system was robust enough to handle irregular inputs gracefully without crashing or exposing vulnerabilities.

7. Performance Testing

Performance testing was done to measure how efficiently the application responds during various operations. This included checking the loading time of high-

resolution images, responsiveness of the interface during clicks, and the speed of transitioning between different windows. It was also tested for performance under multiple user records to ensure that registration and login remained quick and smooth even with more data stored.

CHAPTER 8

PROGRAMMING CODE

8.1 SOURCE CODE

```
import cv2

import tkinter as tk

from tkinter import filedialog, messagebox

import json

import os

import sys

import subprocess

from PIL import Image, ImageTk

selected_points = []

selected_images = []

data_file = "password_data.json"

user_data_file = "user_data.json"

current_image_index = 0

canvas = None

window = None

TOLERANCE = 10

locked_file_path = None

locked_folder_path = None

def resource_path(relative_path)

if hasattr(sys, '_MEIPASS'):

return os.path.join(sys._MEIPASS, relative_path)

return os.path.join(os.path.abspath("."), relative_path)

def save_user_credentials(email, password):

data = {"email": email, "password": password}

with open(user_data_file, "w") as f:
```

```

json.dump(data, f)

def validate_user(email, password):
    try:
        with open(user_data_file, "r") as f:
            stored_data = json.load(f)
            return stored_data["email"] == email and stored_data["password"] ==
            password
    except FileNotFoundError:
        return False

def show_home():
    for widget in app.winfo_children():
        widget.destroy()

    home_frame = tk.Frame(app)
    home_frame.place(relx=0.5, rely=0.5, anchor="center")

    tk.Label(home_frame, text="Graphical Password Authentication",
            font=("Arial", 35, "bold")).pack(pady=10)

    try:
        image_path = resource_path("C:\\FINAL-PROJECT\\lock.PNG")
        lock_image = tk.PhotoImage(file=image_path)
        lock_image = lock_image.subsample(4, 4)
        image_label = tk.Label(home_frame, image=lock_image)
        image_label.image = lock_image # Keep a reference
        image_label.pack()
    except Exception as e:
        tk.Label(home_frame, text="[Image missing: lock.png]", font=("Arial",
            12), fg="red").pack(pady=5)

```

```

tk.Button(home_frame, text="Register", command=show_register,
font=("Arial", 16),
width=15, height=2, bg="#4CAF50", fg="white").pack(pady=10)

tk.Button(home_frame, text="Login", command=show_login,
font=("Arial", 16),
width=15, height=2, bg="#008CBA", fg="white").pack(pady=10)

tk.Button(home_frame, text="Exit", command=app.quit, font=("Arial",
16),
width=15, height=2, bg="red", fg="white").pack(pady=10)

def show_register():
for widget in app.winfo_children():
widget.destroy()

frame = tk.Frame(app)
frame.place(relx=0.5, rely=0.5, anchor="center")

tk.Label(frame, text="Register With Mail", font=("Arial", 18,
"bold")).pack(pady=10)

tk.Label(frame, text="Email:", font=("Arial", 15)).pack(pady=5)

email_entry = tk.Entry(frame, font=("Arial", 12), width=30)
email_entry.pack(pady=5)

tk.Label(frame, text="Password:", font=("Arial", 15)).pack(pady=5)

password_entry = tk.Entry(frame, font=("Arial", 12), show="*",
width=30)
password_entry.pack(pady=5)

def submit():
email = email_entry.get()
password = password_entry.get()

if not email or not password:
messagebox.showerror("Error", "Both fields are required!")

```

```

return

save_user_credentials(email, password)

messagebox.showinfo("Success", "Registration Successful! Now set
your graphical password.")

select_images()

tk.Button(frame, text="Submit", font=("Arial", 12), command=submit,
bg="#4CAF50", fg="white").pack(pady=10)

tk.Button(frame, text="Exit", font=("Arial", 12), command=app.quit,
bg="red", fg="white").pack(pady=10)

def show_login():
for widget in app.winfo_children():
widget.destroy()

frame = tk.Frame(app)
frame.place(relx=0.5, rely=0.5, anchor="center")

tk.Label(frame, text="Login", font=("Arial", 18,
"bold")).pack(pady=10)

tk.Label(frame, text="Email:", font=("Arial", 12)).pack(pady=5)
email_entry = tk.Entry(frame, font=("Arial", 12), width=30)
email_entry.pack(pady=5)

tk.Label(frame, text="Password:", font=("Arial", 12)).pack(pady=5)
password_entry = tk.Entry(frame, font=("Arial", 12), show="*",
width=30)
password_entry.pack(pady=5)

def submit():
email = email_entry.get()
password = password_entry.get()
if validate_user(email, password):
authenticate()

```

```

else:

messagebox.showerror("Error", "Invalid credentials! Try again.")

tk.Button(frame, text="Submit", font=("Arial", 12), command=submit,
bg="#008CBA", fg="white").pack(pady=10)

tk.Button(frame, text="Exit", font=("Arial", 12), command=app.quit,
bg="red", fg="white").pack(pady=10)

def select_images():

global selected_images

files = filedialog.askopenfilenames(title="Select 5 Images",
filetypes=[("Image Files", "*.jpg;*.png;*.jpeg")])

if len(files) != 5:

messagebox.showerror("Error", "You must select exactly 5 images!")

return

selected_images = list(files)

show_image_selection()

def show_image_selection():

global current_image_index, selected_points, window, canvas

selected_points = []

current_image_index = 0

window = tk.Toplevel(app)

window.title("Select Cued Points")

window.attributes('-fullscreen', True)

top_frame = tk.Frame(window, bg="white")

top_frame.pack(fill="x")

tk.Label(top_frame, text="Select cued points", font=("Arial", 18,
"bold"), bg="white", fg="black").pack(pady=10)

canvas = tk.Canvas(window, bg="white")

```

```

canvas.pack(fill="both", expand=True)

tk.Button(window, text="Exit", font=("Arial", 14), bg="red",
fg="white", command=window.destroy).pack(pady=10, side="bottom")

def next_image(event):
    global current_image_index
    x, y = event.x, event.y
    selected_points.append((x, y))
    canvas.create_oval(x-5, y-5, x+5, y+5, fill="red")
    canvas.update()
    if len(selected_points) == 5:
        data = {"image_paths": selected_images, "password_points":
selected_points}
        with open(data_file, "w") as f:
            json.dump(data, f)
        messagebox.showinfo("Success", "Graphical password set
successfully!")
        window.destroy()
        show_file_folder_window(lock_mode=True)
        return
    current_image_index += 1
    if current_image_index < 5:
        load_image()
    def load_image():
        canvas.delete("all")
        img = Image.open(selected_images[current_image_index])
        img_tk = ImageTk.PhotoImage(img)

```

```

canvas.create_rectangle(0, 0, img_tk.width(), img_tk.height(),
outline="blue", width=5)

canvas.create_image(img_tk.width()/2, img_tk.height()/2,
image=img_tk)

canvas.image = img_tk

canvas.bind("<Button-1>", next_image)

load_image()

def authenticate():
    try:
        with open(data_file, "r") as f:
            stored_data = json.load(f)
        except FileNotFoundError:
            messagebox.showerror("Error", "No password setup found!")
        return

        selected_images = stored_data["image_paths"]
        stored_points = stored_data["password_points"]

        global current_image_index, selected_points, window, canvas
        selected_points = []
        current_image_index = 0

        window = tk.Toplevel(app)
        window.title("Authenticate")
        window.attributes('-fullscreen', True)

        top_frame = tk.Frame(window, bg="white")
        top_frame.pack(fill="x")

        tk.Label(top_frame, text="Select cued points", font=("Arial", 18,
"bold"), bg="white", fg="black").pack(pady=10)

        canvas = tk.Canvas(window, bg="white")

```



```

canvas.pack(fill="both", expand=True)

tk.Button(window, text="Exit", font=("Arial", 14), bg="red",
fg="white", command=window.destroy).pack(pady=10, side="bottom")

def next_image(event):
    global current_image_index
    x, y = event.x, event.y
    selected_points.append((x, y))
    canvas.create_oval(x-5, y-5, x+5, y+5, fill="red")
    canvas.update()
    if len(selected_points) == 5:
        for i in range(5):
            sx, sy = stored_points[i]
            cx, cy = selected_points[i]
            if not (abs(sx - cx) <= TOLERANCE and abs(sy - cy) <=
TOLERANCE):
                messagebox.showerror("Error", "Authentication Failed!")
                window.destroy()
                show_home()
        return
    messagebox.showinfo("Success", "Authentication Successful!")
    window.destroy()
    show_file_folder_window(lock_mode=False)
    return
    current_image_index += 1
    if current_image_index < 5:
        load_image()
    def load_image():

```

```

canvas.delete("all")

img = Image.open(selected_images[current_image_index])
img_tk = ImageTk.PhotoImage(img)

canvas.create_rectangle(0, 0, img_tk.width(), img_tk.height(),
outline="blue", width=5)

canvas.create_image(img_tk.width()/2, img_tk.height()/2,
image=img_tk)

canvas.image = img_tk

canvas.bind("<Button-1>", next_image)

load_image()

def show_file_folder_window(lock_mode=True):
    global locked_file_path, locked_folder_path

    window = tk.Toplevel(app)

    window.attributes('-fullscreen', True)

    window.title("File/Folder Access")

    frame = tk.Frame(window)

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

    title = "Lock File/Folder" if lock_mode else "Unlock File/Folder"
    tk.Label(frame, text=title, font=("Arial", 18, "bold")).pack(pady=10)

    def lock_file():
        global locked_file_path

        file_path = filedialog.askopenfilename(title="Select File to Lock")

        if file_path:
            try:
                subprocess.run(["attrib", "+h", file_path], shell=True)

                locked_file_path = file_path

                messagebox.showinfo("Locked", f"File locked: {file_path}")

```

```

window.destroy()

show_home()

except Exception as e:
    messagebox.showerror("Error", f"Failed to lock file: {str(e)}")

def lock_folder():
    global locked_folder_path

    folder_path = filedialog.askdirectory(title="Select Folder to Lock")

    if folder_path:
        try:
            subprocess.run(["attrib", "+h", folder_path], shell=True)

            locked_folder_path = folder_path

            messagebox.showinfo("Locked", f"Folder locked: {folder_path}")

            window.destroy()

            show_home()

        except Exception as e:
            messagebox.showerror("Error", f"Failed to lock folder: {str(e)}")

def unlock_file():
    global locked_file_path

    if locked_file_path:
        try:
            subprocess.run(["attrib", "-h", locked_file_path], shell=True)

            messagebox.showinfo("Unlocked", f"File unlocked:
{locked_file_path}")

            locked_file_path = None

            window.destroy()

            show_home()

        except Exception as e:

```

```

messagebox.showerror("Error", f"Failed to unlock file: {str(e)}")

else:

messagebox.showwarning("No File", "No file has been locked in this
session.")

def unlock_folder():

global locked_folder_path

if locked_folder_path:

try:

subprocess.run(["attrib", "-h", locked_folder_path], shell=True)

messagebox.showinfo("Unlocked", f"Folder unlocked:
{locked_folder_path}")

locked_folder_path = None

window.destroy()

show_home()

except Exception as e:

messagebox.showerror("Error", f"Failed to unlock folder: {str(e)}")

else:

messagebox.showwarning("No Folder", "No folder has been locked in
this session.")

if lock_mode:

tk.Button(frame, text="Lock File", font=("Arial", 14), bg="#FFA500",
fg="white", command=lock_file).pack(pady=10)

tk.Button(frame, text="Lock Folder", font=("Arial", 14),
bg="#FFA500", fg="white", command=lock_folder).pack(pady=10)

else:

tk.Button(frame, text="Unlock File", font=("Arial", 14),
bg="#32CD32", fg="white", command=unlock_file).pack(pady=10)

```

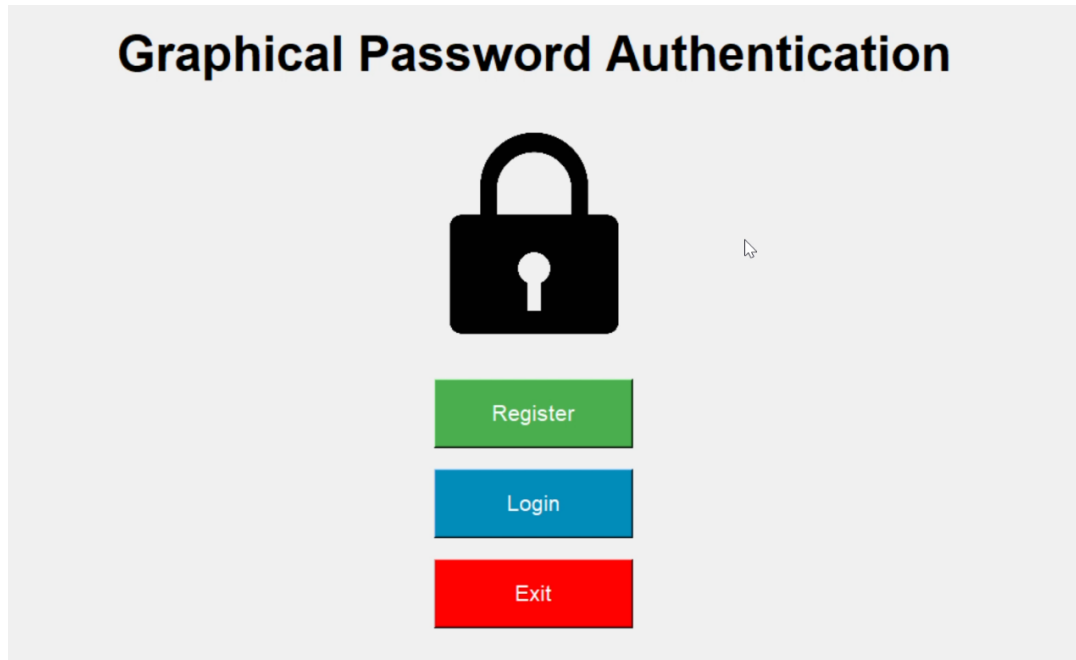
```
tk.Button(frame, text="Unlock Folder", font=("Arial", 14),
bg="#32CD32", fg="white", command=unlock_folder).pack(pady=10)

tk.Button(frame, text="Exit", font=("Arial", 14), bg="red", fg="white",
command=window.destroy).pack(pady=20)

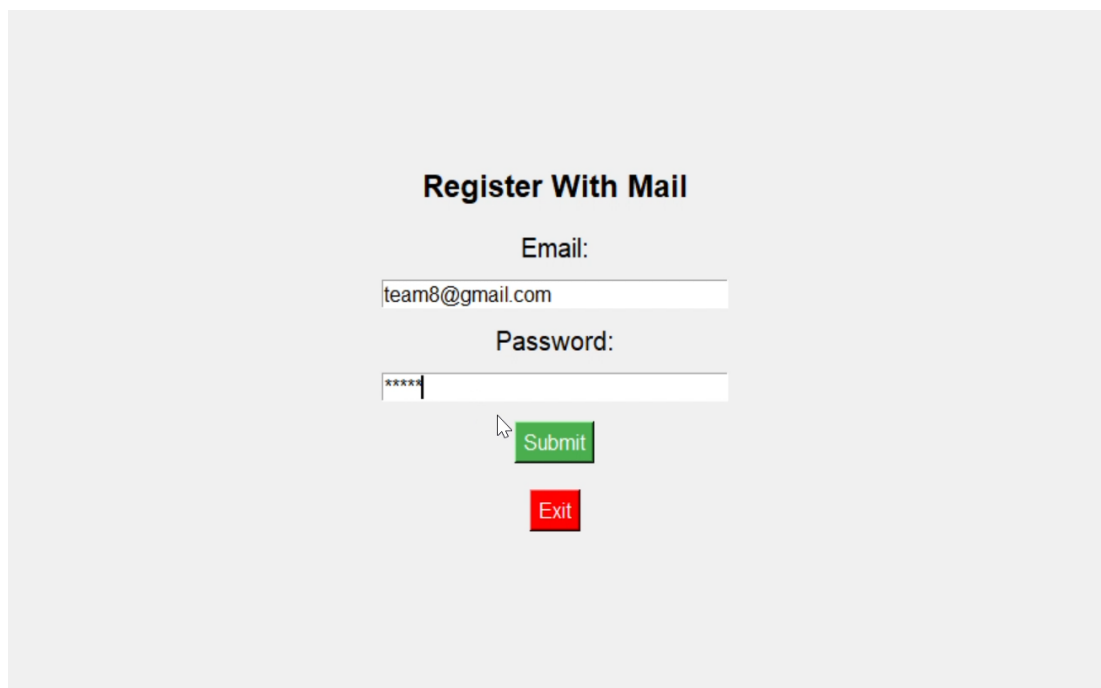
app = tk.Tk()                                #RUN THE APP
app.attributes('-fullscreen', True)
app.title("Graphical Password Authentication")
show_home()
app.mainloop()
```

8.2 OUTPUT SCREENSHOTS

1.HOME PAGE



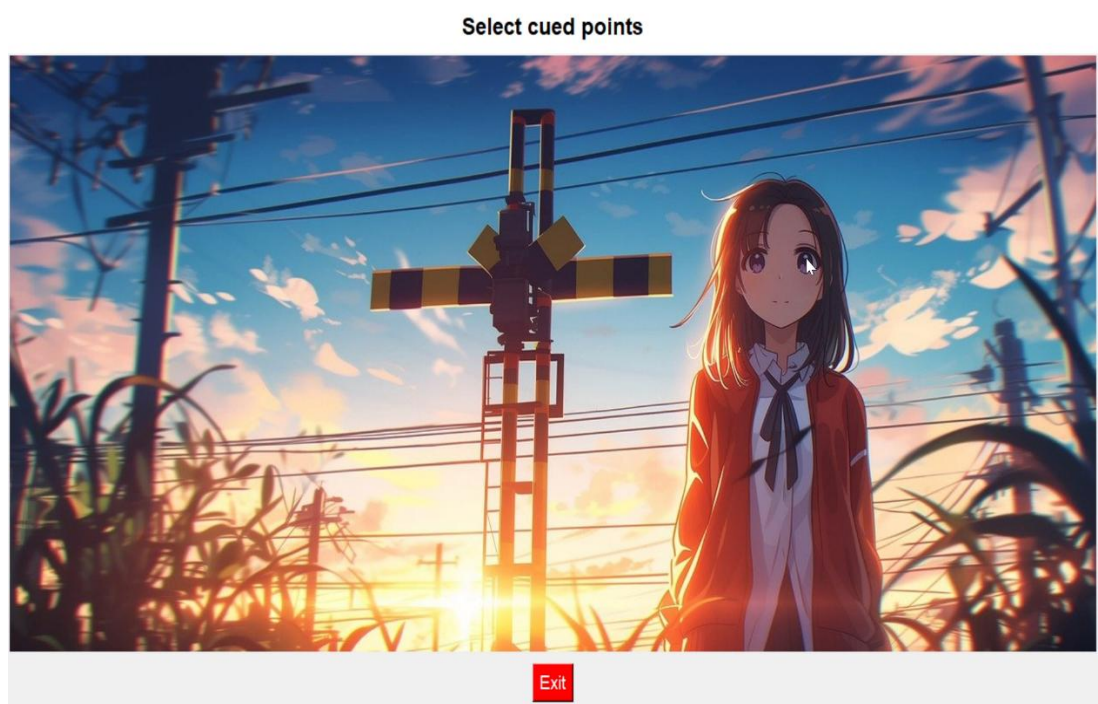
2.REGISTRATION PHASE



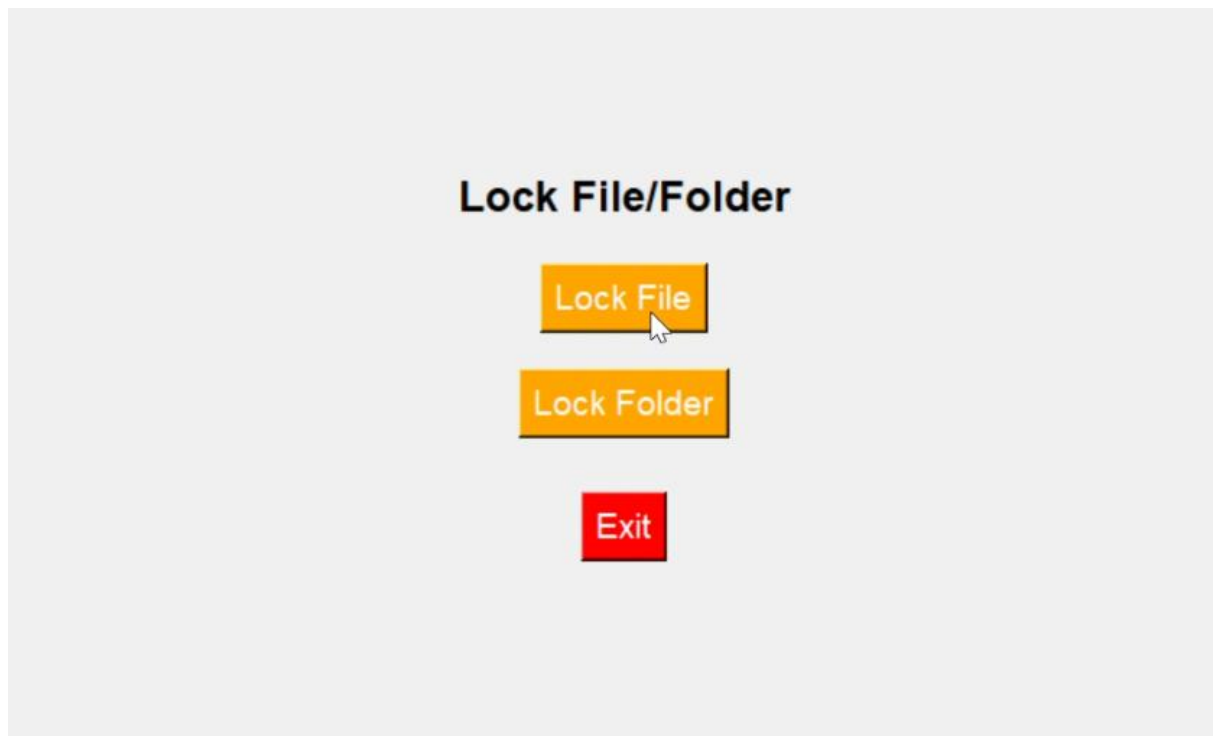
3.IMAGE SELECTION PHASE



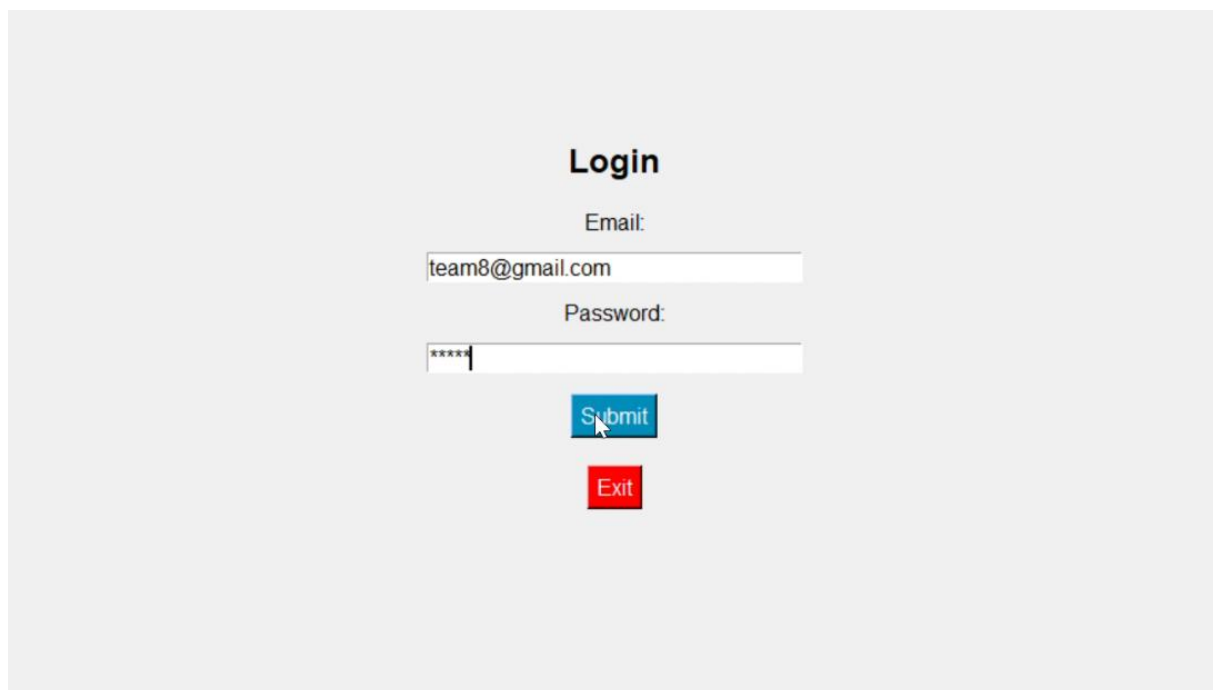
4.CUED POINT SELECTION PHASE



5.FOLDER/FILE LOCKING



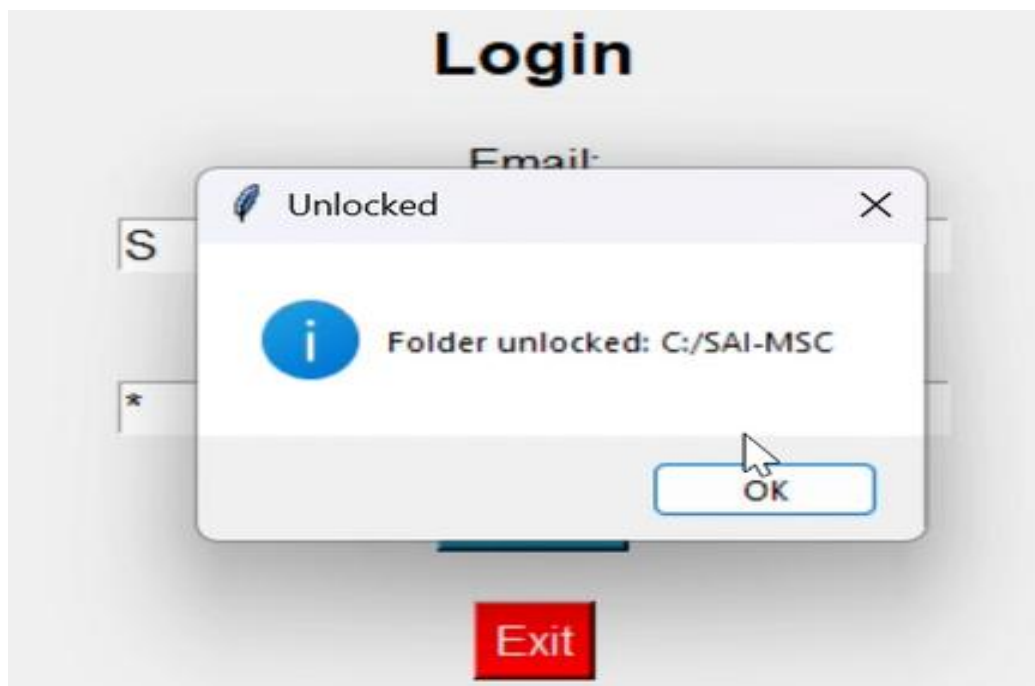
6.LOGIN PHASE



7.UNLOCKING FILE/FOLDER



8.FOLDER UNLOCKED SUCCESSFULLY



CONCLUSION

The proposed Cued Click Points scheme shows promise as a usable and memorable authentication mechanism. By taking advantage of users' ability to recognize images and the memory trigger associated with seeing a new image, CCP has advantages over PassPoints in terms of usability. Being cued as each image is shown and having to remember only one click-point per image appears easier than having to remember an ordered series of clicks on one image. In our small comparison group, users strongly preferred CCP. We believe that CCP offers a more secure alternative to PassPoints. CCP increases the workload for attackers by forcing them to first acquire image sets for each user, and then conduct hotspot analysis on each of these images. The security of CCP also deserves closer examination, and should address how attackers might exploit the emergence of hotspots. The purpose of this abstract is to introduce an improved authentication system that is both user-friendly and resistant to shoulder surfing attacks. The system incorporates both text-based and graphical-based approaches and aims to reduce the memory efforts required by the end-user. The current technological advancements suggest that system security will be a critical aspect of the next era, and graphical passwords may become a prevalent authentication method. The system is designed as a recognition and recall-based model, making it more usable and secure compared to previous graphical password authentication systems. Its password space is extensive, providing robust security against brute force attacks. Password creation and memorization are easy, and randomization in both authentication steps offers high security against shoulder surfing and other possible attacks. The system can be used for highly secure applications. A future addition to the system could be the implementation of a password retrieval feature through email and mobile messages, ensuring users can receive system updates even when offline. Overall, the system aims to provide both security and ease of access, with potential future improvements for even greater security.

REFERENCES

1. Birget, J.C., Hong, D., Memon, N.: Graphical Passwords Based on Robust Discretization. *IEEE Trans. Info. Forensics and Security* 1(3) (September 2006)
2. Blonder, G.E.: Graphical Passwords. United States Patent 5,559,961 (1996)
3. Chiasson, S., Biddle, R.R., van Oorschot, P.C.: A Second Look at the Usability of Click-based Graphical Passwords. *ACM SOUPS* (2007)
4. Cranor, L.F., Garfinkel, S.: *Security and Usability*. O'Reilly Media (2005)
5. Davis, D., Monroe, F., Reiter, M.K.: On User Choice in Graphical Password Schemes. In: *13th USENIX Security Symposium* (2004)
6. Dirik, A.E., Menon, N., Birget, J.C.: Modeling user choice in the PassPoints graphical password scheme. *ACM SOUPS* (2007)
7. Jermyn, I., Mayer, A., Monroe, F., Reiter, M.K., Rubin, A.D.: The Design and Analysis of Graphical Passwords. In: *8th USENIX Security Symposium* (1999)
8. Nelson, D.L., Reed, U.S., Walling, J.R.: Picture Superiority Effect. *Journal of Experimental Psychology: Human Learning and Memory* 3, 485–497 (1977)
9. Passfaces (last accessed: December 1, 2006), <http://www.realuser.com>
10. Peters, M.: Revised Vandenberg & Kuse Mental Rotations Tests: forms MRT-A to MRT-D. Technical Report, Department of Psychology, University of Guelph (1995)
11. Pinkas, B., Sander, T.: Securing Passwords Against Dictionary Attacks. *ACM CCS* (2002)
12. Renaud, K.: Evaluating Authentication Mechanisms. In: [4], ch. 6
13. Renaud, K., De Angeli, A.: My password is here! An investigation into visio-spatial authentication mechanisms. *Interacting with Computers* 16, 1017–1041 (2004)
14. Suo, X., Zhu, Y., Owen, G.S.: Graphical Passwords: A Survey. In: *Annual Computer Security Applications Conference* (2005)

15. Tari, F., Ozok, A.A., Holden, S.H.: A Comparison of Perceived and Real Shoulder surfing Risks between Alphanumeric and Graphical Passwords. ACM SOUPS (2006)
16. Thorpe, J., van Oorschot, P.C.: Human-Seeded Attacks and Exploiting Hot-Spots in Graphical Passwords. In: 16th USENIX Security Symposium (2007)
17. van Oorschot, P.C., Stubblebine, S.: On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop. ACM Trans. Information and System Security 9(3), 235–258 (2006)
18. Weinshall, D.: Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In: IEEE Symposium on Security and Privacy (2006)
19. Wiedenbeck, S., Birget, J.C., Brodskiy, A., Memon, N.: Authentication Using Graphical Passwords: Effects of Tolerance and Image Choice. ACM SOUPS (2005)
20. Wiedenbeck, S., Waters, J., Birget, J.C., Brodskiy, A., Memon, N.: PassPoints: Design and longitudinal evaluation of a graphical password system. International Journal of Human-Computer Studies 63, 102–127 (2005)
21. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password Memorability and Security: Empirical Results. IEEE Security & Privacy Magazine 2(5) (2004)