

ABSTRACT

Nowadays, Electronic Transactions are becoming popular everywhere. They can be used in a variety of applications such as Fund Transfer, Online Shopping, Bill Payments, e-Commerce, etc., The Personal Identification Number (PIN) is a well-known and popular authentication method used in Electronic Transaction. However, the traditional PIN Entry method is vulnerable to a variety of attacks such as Guessing Attacks, Human Shoulder Surfing Attacks (SSA), Hidden Cameras, Smudge Attacks, Side-Channel Attacks and many more. Many methods have been proposed to overcome one or more of these attacks. The Dual Pad solution proposed can be used to overcome many of the above mentioned attacks. The proposed solution consists of a Virtual Keypad, in which the keys are scrambled during every authentication attempt using Fisher-Yates Shuffling Algorithm. It also makes use of an OTP generation mechanism with the help of Pseudo-Random Number Generation Algorithm. However, the user does not directly enter their OTP, they combine it with a hidden code. The solution proposed requires lesser amount of time for authentication as the traditional one and also is easy to adapt to other systems such as electronic door locks, smartphones and ATMs.

CHAPTER 1

INTRODUCTION

1.1 ELECTRONIC TRANSACTION

An electronic transaction is the sale or purchase of goods or services, whether between businesses, households, individuals, governments, and other public or private organisations, conducted over computer-mediated networks. The goods and services are ordered over those networks, but the payment and the ultimate delivery of the good or service may be conducted on or off-line. When you pay for goods or services with your debit card, you have an option for the payment to be processed in two different ways: as an offline transaction via a credit card processing network, or as an online transaction via an EFT system, requiring a PIN to complete the process.

1.2 PERSONAL IDENTIFICATION NUMBER(PIN)

A PIN, is a numeric or alpha-numeric password used in the process of authenticating a user accessing a system. The personal identification number has been the key to flourishing the exchange of private data between different data-processing centres in computer networks for financial institutions, governments, and enterprises. PINs may be used to authenticate banking systems with cardholders, governments with citizens, enterprises with employees, and computers with users, among other uses. In common usage, PINs are used in ATM or POS transactions, secure access control (e.g. computer access, door access, car access), internet transactions or to log into a restricted website.

1.2.1 PIN USAGE

In the context of a financial transaction, usually both a private "PIN code" and public user identifier is required to authenticate a user to the system. In these situations, typically the user is required to provide a non-confidential user identifier or token (the user ID) and a confidential PIN to gain access to the system. The user is granted access only when the number entered matches with the number stored in the system.

1.2.2 PIN LENGTH

The international standard for financial services PIN management, ISO 9564-1, allows for PINs from four up to twelve digits, but recommends that for usability reasons the card issuer not assign a PIN longer than six digits. The most commonly used length in many places is four digits.

1.2.3 PIN SECURITY

Financial PINs are often four-digit numbers in the range 0000–9999, resulting in 10,000 possible combinations. Some systems set up default PINs and most allow the customer to set up a PIN or to change the default one, and on some a change of PIN on first access is mandatory. Customers are usually advised not to set up a PIN based on their or their spouse's birthdays, on driver license numbers, consecutive or repetitive numbers, or some other schemes.

1.3 ATTACKS AGAINST PIN

The PIN is a well-known and popular authentication method used in Electronic Transaction. However, it is susceptible to a number of attacks such as guessing attacks, shoulder surfing attacks, smudge attacks, and side-channel attacks.

1.3.1 SHOULDER SURFING ATTACKS

Shoulder surfing is a type of social engineering technique used to obtain information such as PIN, passwords and other confidential data by looking over the victim's shoulder. This attack can be performed either at close range (by directly looking over the victim's shoulder) or from a longer range, for example by using a pair of binoculars or similar hardware. To implement this technique attackers do not require any technical skills; keen observation of victims' surroundings and the typing pattern is sufficient. Crowded places are the more likely areas for an attacker to shoulder surf the victim.

1.3.2 HIDDEN CAMERA ATTACKS

A hidden camera or spy camera or security camera is a still or video camera used to record people without their knowledge. The camera may be "hidden" because it is not visible to the subject being filmed, or is disguised as another object. A hidden camera allows the attacker to capture whole login process and other confidential data of the victim, which ultimately could lead to financial loss or identity theft.

1.3.3 SMUDGE ATTACKS

A smudge attack is a method to discern the password pattern of a touchscreen device such as a cell phone or tablet computer. The smudge attack relies on detecting the oily smudges left behind by the user's fingers when operating the device using simple cameras and image processing software. Under proper lighting and camera settings, the finger smudges can be easily detected, and the heaviest smudges can be used to infer the most frequent user input pattern (the password).

1.3.4 SIDE CHANNEL ATTACKS

Side-channel attacks use various side-channel information to infer the entry position on the device screen. Using the video footage that captures the

spatial-temporal hand dynamics of a user typing on a device, the attack maps the position of the user's hands to the position of the keypad to infer the PIN. In average, the attack can break over 50% of the PINs on the first attempt and over 80% of the PINs in ten attempts.

CHAPTER 2

LITERATURE SURVEY

2.1 SPY RESISTANT KEYBOARD: MORE SECURE PASSWORD ENTRY ON PUBLIC TOUCH SCREEN DISPLAYS

Current software interfaces for entering text on touch screen devices mimic existing mechanisms such as keyboard typing or handwriting. These techniques are poor for entering private text such as passwords since they allow observers to decipher what has been typed simply by looking over the typist's shoulder, an activity known as shoulder surfing. This paper outlines a general approach for designing security-sensitive onscreen virtual keyboards that allow users to enter private text without revealing it to observers. This paper presents one instantiation, the Spy-Resistant Keyboard[2], and discuss design decisions leading to the development of this keyboard. It also describes the results of a user study exploring the usability and security of our interface. Results indicate that although users took longer to enter their passwords, using the Spy-Resistant Keyboard rather than a standard soft keyboard resulted in a significant increase in their ability to protect their passwords from a watchful observer.

2.2 REDUCING SHOULDER SURFING BY USING GAZE BASED PASSWORD ENTRY

Shoulder-surfing – using direct observation techniques, such as looking over someone's shoulder, to get passwords, PINs and other sensitive personal information – is a problem that has been difficult to overcome. This paper presents EyePassword[6], a system that mitigates the issues of shoulder surfing via a novel approach to user input. With EyePassword, a user enters sensitive input (password, PIN, etc.) by selecting from an on-screen keyboard using only the orientation of their pupils (i.e. the position of their gaze on screen), making eavesdropping by a malicious observer largely impractical. It presents a number

of design choices and discuss their effect on usability and security. The paper conducted user studies to evaluate the speed, accuracy and user acceptance of our approach. The results demonstrate that gaze-based password entry requires marginal additional time over using a keyboard, error rates are similar to those of using a keyboard and subjects preferred the gaze-based password entry approach over traditional methods.

2.3 SECURE AND USER FRIENDLY PIN ENTRY METHOD

Personal identification number (PIN) is a common user authentication method widely used for ATMs and mobile phones. However, an attacker can obtain the PIN easily by looking over a user's shoulder. This paper presents a new PIN entry method[4] which makes this kind of attack significantly harder.

2.4 USING FAKE CURSORS TO SECURE ON-SCREEN PASSWORD ENTRY

This paper presents a concept using fake cursors[5] to disguise on-screen password entry. It performed two user studies with different amounts of dummy cursors and differently coloured cursors. The results show that dummy cursors significantly improve security. At the same time, decrease in performance is kept within an acceptable range. Depending on the required degree of security, the studies favour 8 or 16 differently coloured cursors as the best trade-off between security and usability.

2.5 DESIGNING LEAKAGE-RESILIENT PASSWORD ENTRY ON TOUCHSCREEN MOBILE DEVICES

Touch screen mobile devices are becoming commodities as the wide adoption of pervasive computing. These devices allow users to access various services at anytime and anywhere. In order to prevent unauthorized access to these services, passwords have been pervasively used in user authentication. However, password-based authentication has intrinsic weakness in password

leakage. Most prior research on improving leakage resilience of password entry focuses on desktop computers, where specific restrictions on mobile devices such as small screen size are usually not addressed. This paper proposes a user authentication scheme named CoverPad[7] for password entry on touchscreen mobile devices. CoverPad improves leakage resilience by safely delivering hidden messages, which break the correlation between the underlying password and the interaction information observable to an adversary. It is also designed to retain most benefits of legacy passwords, which is critical to a scheme intended for practical use. The usability of CoverPad is evaluated with an extended user study which includes additional test conditions related to time pressure, distraction, and mental workload. The results of this user study show the impacts of these test conditions on user performance as well as the practicability of the proposed scheme.

2.6 IDENTIFICATION OF PRESSED KEYS FROM MECHANICAL VIBRATIONS

This paper describes an attack that identifies the sequence of keystrokes analysing mechanical vibrations generated by the act of pressing keys. It uses accelerometers [3] as vibration sensors. The apparatus necessary for this attack is inexpensive and can be unobtrusively embedded within the target equipment. We tested the proposed attack on an ATM keypad and a PIN-pad. It achieved the key recognition rates of 98.4% in ATM keypad, 76.7% in PIN-pad resting on a hard surface, and 82.1% in PIN-pad held in hand.

2.7 AN OBSERVATION ATTACKS RESISTANT PIN-ENTRY SCHEME USING LOCALISED HAPTIC FEEDBACK

Common PIN-entry schemes are vulnerable to observation attacks. To enhance the resistance to observation attacks, some observation attacks resistant PIN-entry schemes for mobile devices based on audios and/or haptics have been proposed. However, none of existing observation attacks resistant PIN-entry

schemes can achieve both good security and high usability. This paper proposes a new observation attacks resistant PIN-entry scheme, Loc-HapPIN[9], for touchscreen devices providing localized haptic feedback. By using the technology of localized haptic feedback, the usability and the resistance to observation attacks are improved. Furthermore, the user can choose the efficiency-security setting suitable for him.

2.8 DYNAMICPIN: A NOVEL APPROACH TOWARDS SECURE ATM AUTHENTICATION

Along with the popularity and widespread use of automated teller machines (ATMs), ATM frauds are also increasing drastically these days. Shoulder-surfing attacks, such as card skimming, PIN capturing using fake machines or fake PIN pads, are the most common methods used by adversaries to capture data from the magnetic stripe on the back of the ATM card. The main problem lies in the existing static PIN-based authentication mechanism which does not provide any security measure (besides displaying asterisks when an user enters a pre-assigned PIN to the ATM). This paper gives a novel approach called DynamicPIN[8] for secure ATM authentication, which is resilient to shoulder-surfing attacks. DynamicPIN is very simple, does not require any hardware changes, and does not pose any significant overhead to the system. A real time experimental study showed that DynamicPIN improves significantly the ATM authentication compared to the existing static PIN-based authentication mechanism.

2.9 A STUDY OF THE SECURITY AND USABILITY OF SCRAMBLE KEYPAD FOR PIN UNLOCK ON SMARTPHONES

PIN unlock is a popular screen locking mechanism used for protecting the sensitive private information on smartphones. However, it is susceptible to a number of attacks such as guessing attacks, shoulder surfing attacks, smudge attacks, and side-channel attacks. Scramble keypad [10] changes the keypad

layout in each PIN-entry process to improve the security of PIN unlock. In this paper, the security and usability of scramble keypad for PIN unlock are studied. Our security analysis shows that scramble keypad can defend smudge attacks perfectly and greatly reduce the threats of side-channel attacks. A user study is conducted to demonstrate that scramble keypad has a better chance to defend shoulder surfing attacks than standard keypad. We also investigate how the usability of scramble keypad is compromised for improved security through a user study.

2.10 REVOLVING FLYWHEEL PIN ENTRY METHOD TO PREVENT SHOULDER SURFING ATTACKS

Nowadays smart phones are replacing computer, laptops and many other electronic gadgets. Smart phones are used for performing transaction, accessing private information like emails, contact details, personal data etc. For the security purpose PIN (Personal identification numbers) are used to lock the smart devices. Many graphical and textual authentication methods are proposed to prevent from shoulder surfing attacks, but the challenging problem is to provide security and usability with less time for authentication. This paper begins with study and analysis of Tictoc PIN entry method [12]. It proposes a new scheme named revolving flywheel PIN-Entry method to prevent from shoulder-surfing attack. The new scheme holds a revolving flywheel and downward color pad. User will register digits as PIN and for authentication, instead of digits, four color pad will be used to enter digits of PIN. Proposed method is easy to adapt and did not have any cognitive burden. This method will be easy to follow by ATM machines or wherever PIN is used for authentication.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

ColorPIN is an authentication mechanism that uses indirect input to provide security enhanced PIN entry. At the same time, ColorPIN retains a one-to-one relationship between the length of the PIN and the required number of clicks. A user study showed that ColorPIN is significantly more secure than standard PIN entry while enabling good authentication speed in comparison with related systems.

3.1.1 COLORPIN CONCEPT



3.1: Exemplary PIN entry with ColorPIN

A PIN in this system contains of a combination of digits, of which each digit is combined with a colour (black, red or white). That is, a four-digit PIN in the ColorPIN system could look like the following:

1 (Black) – 2 (Red) – 3 (White) – 4 (Black)

The user interface consists of a keypad representation depicting the digits 1 – 9 (please note that the digit 0 has been removed due to reasons of simplicity) as shown in figure 3.1. On the bottom of each number, three differently coloured letters can be found. Those letters are randomly assigned at the

beginning of the interaction. Additionally, due to security reasons the letters are newly assigned each time the user presses a key.

To input a digit of their PIN, the user has to input the letter that is displayed at the digit's bottom in the respective colour. Input is done on a conventional keyboard with one key per letter.

3.1.2 LIMITATIONS OF EXISTING SYSTEM

- The additional information “colour” and the lack of exploiting the users’ motor memory, could lead to worse performance with respect to memorability.
- The indirect input creates extra cognitive load.
- ColorPIN is slower than standard PIN entry.

3.2 PROPOSED SYSTEM

The proposed Dual Pad solution can be used to overcome many of the previously mentioned attacks. It consists of a Virtual Keypad, in which the keys are scrambled during every authentication attempt. The keys are scrambled using Fisher-Yates Shuffling Algorithm. The virtual keypad is composed of two keypads with different digit orderings, blended in a single hybrid image. The user registers them through a sign up page and receives a PIN, which can be used to access their hidden code. Random Generation Algorithm is used to generate OTP which is send to our email or phone number. The OTP is generated out of a combination of 12 colors, namely- Red, Green, Yellow, Blue, White, Lavender, Pink, Orange, Violet, Azure, Maroon, Silver. This OTP can be used for electronic transactions in petrol bunk, shopping mall and in other public places. The user enters their OTP in accordance to their hidden code. This technique will enhance the protection mechanism while performing electronic transactions.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Keypad shuffling eliminates shoulder-surfing problem
- No complex calculations required
- No additional time required
- No extra cognitive load

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS

- **Processor** : intel Core i3
- **Hard Disk** : 500 GB.
- **Monitor** : LCD/LED.
- **RAM** : 4 GB.

4.2 SOFTWARE REQUIREMENTS

- **Operating system** : Windows 7/8.
- **Front End** : Net beans 8.0.2/ Java
- **Back End** : MYSQL

4.3 SOFTWARE SPECIFICATION

4.3.1 FRONT END – NETBEANS / JAVA

4.3.1.1 INTRODUCTION

NetBeans is an Integrated Development Environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, Mac OS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

4.3.1.2 NETBEANS PLATFORM

The NetBeans platform is a framework for simplifying the development of Java swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plug-ins and NetBeans platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the update centre module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management
- User settings management
- Storage management
- Window management
- Wizard framework
- NetBeans visual library
- Integrate development tools

4.3.1.3 NETBEANS IDE

NetBeans IDE is an open source integrated development environment. NetBeans IDE supports development of all Java application types out of the box. Among other features is an Ant-based project system, Maven support, Refactoring, Version Control.

4.3.2 BACK END – MYSQL

Mysql is an open-source relational database management system (RDBMS). Mysql is free and open-source software under the terms of the GNU General Public Licence, and is also available under a variety of proprietary licences. Mysql was owned and sponsored by the Swedish company, Mysql AB, which was bought by Sun Microsystems. In 2010, when Oracle acquired Sun, Widenius forked the open-source Mysql project to create MariaDB.

MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Google, Facebook, Twitter, Flickr, and YouTube.

4.3.2.1 FEATURES

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors

- Updatable views
- Online Data Definition Language (DDL) when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
- A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine
- Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.
- ACID compliance when using InnoDB and NDB Cluster Storage Engines
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Built-in replication support
- Full-text indexing and searching
- Embedded database library
- Unicode support
- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

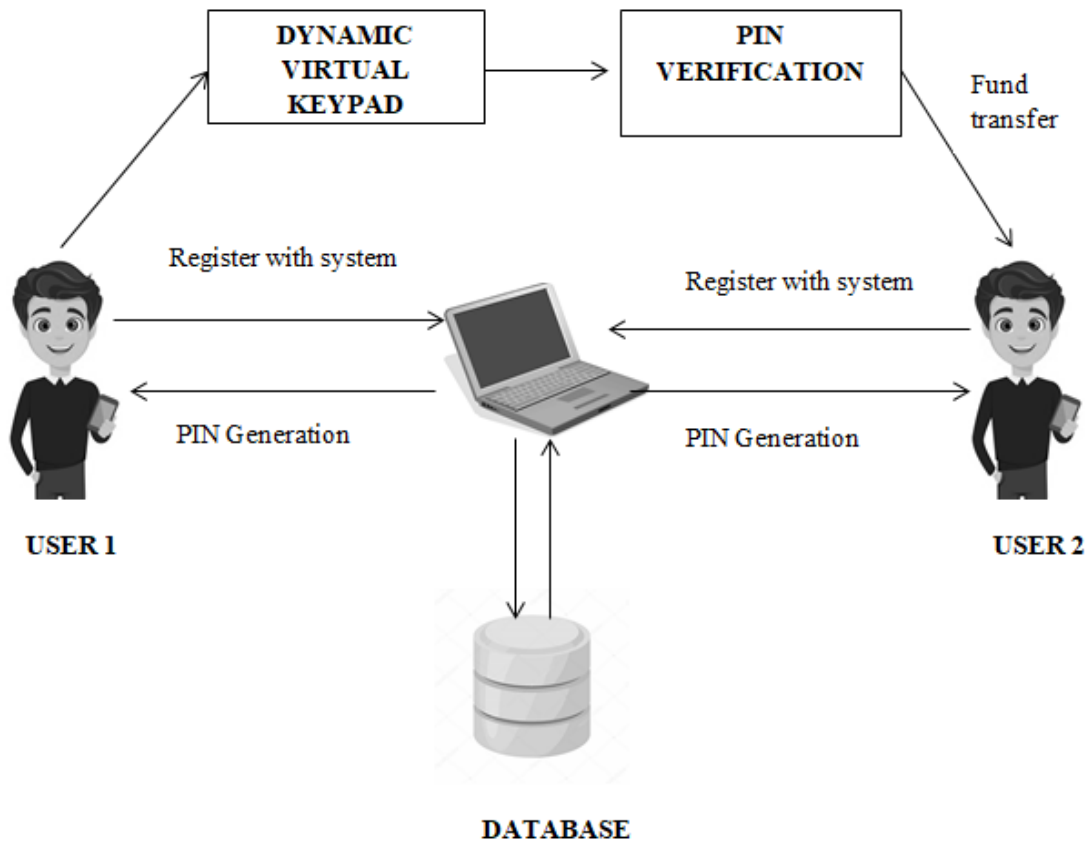


Fig 5.1 Transfer of Funds

The user registers their existing account with some bank mandatory fields. The details are stored in the database. The system generates unique four digit PIN for the user. The PIN is generated using a Random Number Generation algorithm. The PIN is used to access the user's hidden code. The user enters their PIN in a virtual keypad whose keys are scrambled during every attempt. The keypad is scrambled using Fisher-Yates Shuffling algorithm. To transfer funds, the user enters an OTP which is sent to their mail, with their hidden code. Once the verification is over, the user transfers funds to another user registered with the system.

CHAPTER 6

SYSTEM IMPLEMENTATION

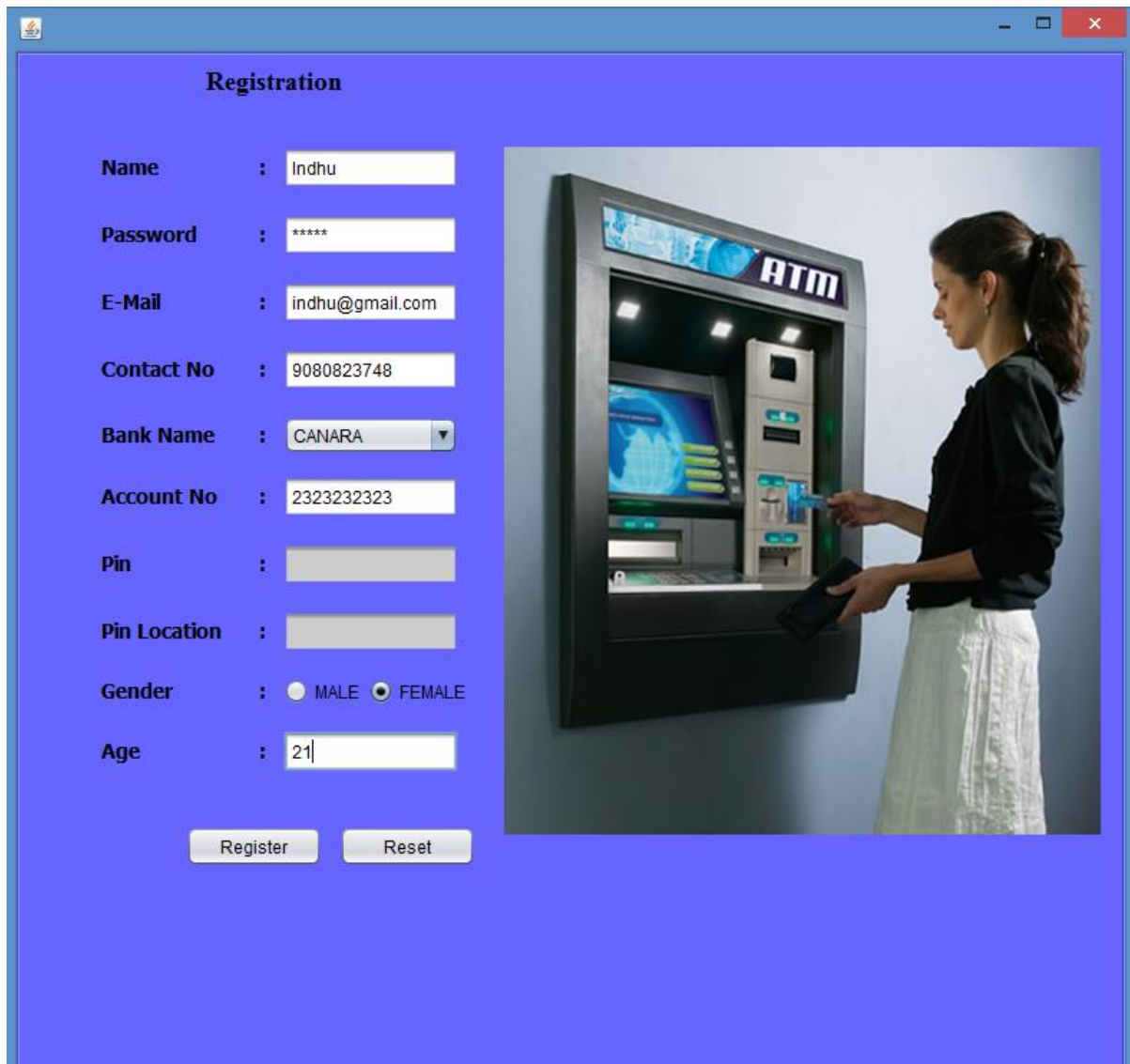
6.1 MODULES

- Account Registration module
- PIN Generation Module
- Virtual keyboard Module
- OTP Generation Module
- Transaction Module

6.2 MODULE DESCRIPTION

6.2.1 ACCOUNT REGISTRATION MODULE

In this module, the user registers their existing account with some bank mandatory fields. The user enters details such as contact number, address, and mail id along with their account number and creates a username and password. Validation has been performed to ensure that only valid phone numbers and mail ids are entered and also to ensure only users of legal age are able to use the system. In order to avoid users registering an account multiple times, a complete scan of the database is conducted every time a new user registers.



Registration

Name : Indhu

Password : *****

E-Mail : indhu@gmail.com

Contact No : 9080823748

Bank Name : CANARA

Account No : 2323232323

Pin :

Pin Location :

Gender : ☐ MALE ☒ FEMALE

Age : 21

Register Reset




Fig 6.1 User Registration

6.2.2 PIN GENERATION MODULE

In this module, a Random number generation algorithm is used to generate a unique four digit PIN for every registered account. The PIN is used as a method of authentication when the user logs into the system. The PIN can be also be used by the user to access their hidden code.



Fig 6.2 PIN Generation

6.2.3 VIRTUAL KEYBOARD MODULE

This module generates a virtual keyboard, using which the user enters their PIN. The keys of the virtual keyboard are scrambled using Fisher-Yates Shuffle Algorithm, during every PIN entry attempt. This reduces the possibility of success for SSA smudge attacks. The Fisher-Yates shuffle is an algorithm for generating a random permutation of a finite sequence—in plain terms, the algorithm shuffles the sequence. The algorithm effectively puts all the elements into a hat; it continually determines the next element by randomly drawing an element from the hat until no elements remain. The algorithm produces an unbiased permutation: every permutation is equally likely. The modern version of the algorithm is efficient: it takes time proportional to the number of items being shuffled and shuffles them in place.

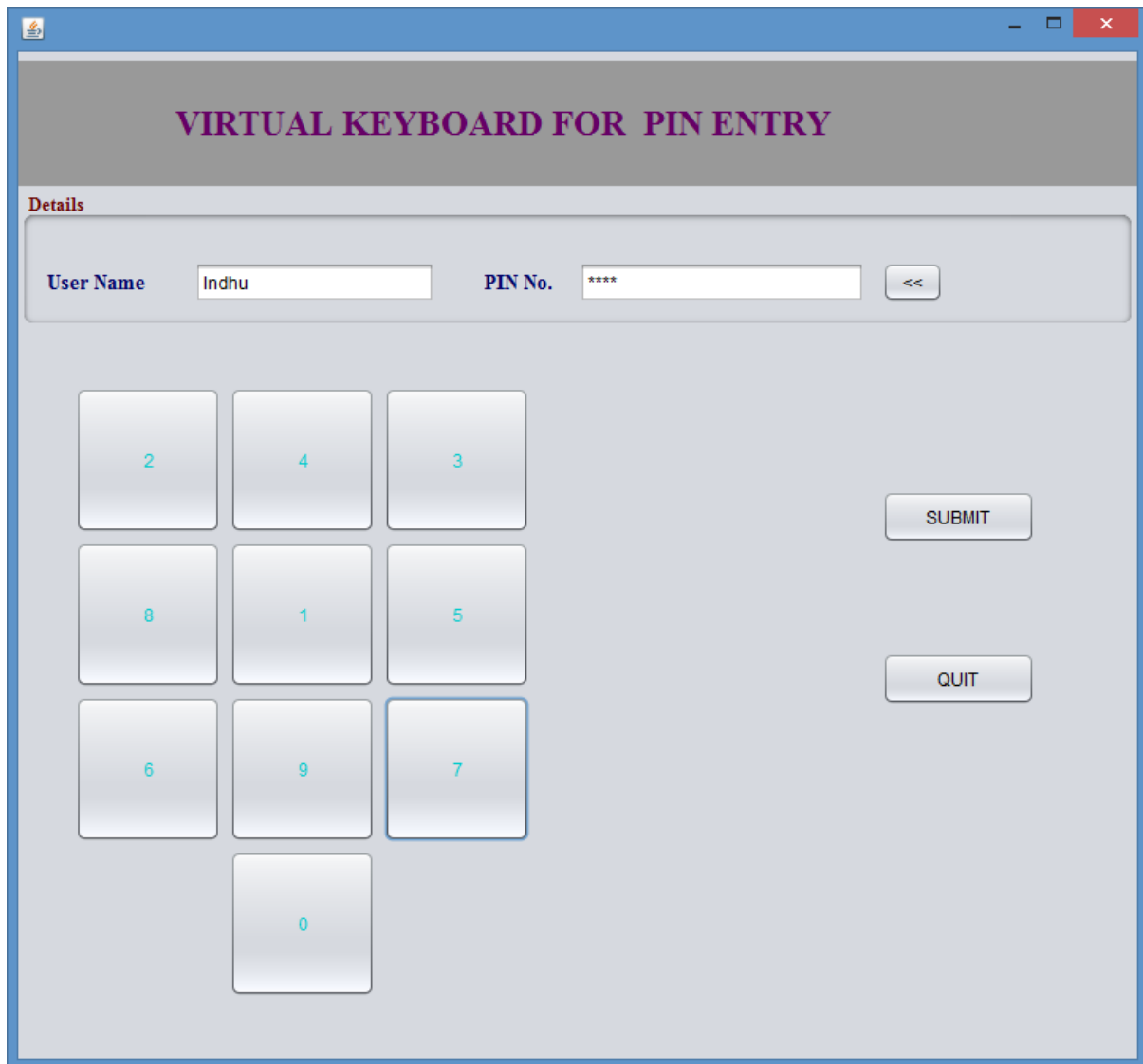


Fig 6.3 Virtual Keypad

6.2.4 OTP GENERATION MODULE

This module generates a four character OTP, which is used as an alternative method of authentication during each transaction. The OTP is generated out of a combination of 12 colors, namely- Red, Green, Yellow, Blue, White, Lavender, Pink, Orange, Violet, Azure, Maroon, Silver. The algorithm used for this module is Pseudo-Random Number Generation Algorithm. The generated OTP is sent to the user's registered mail.

The screenshot shows a software application window titled "VIRTUAL KEYBOARD FOR OTP VERIFICATION". The window is divided into two main sections: "Details" and "Virtual Keyboard".

Details Section:

- User Name:** A text input field containing the name "Indhu".
- OTP Is.:** A text input field containing four asterisks "****". To the right of this field is a button with the text "<<".

Virtual Keyboard Section:

The virtual keyboard is organized into a 4x3 grid of colored boxes, each representing a letter and containing three numeric buttons:

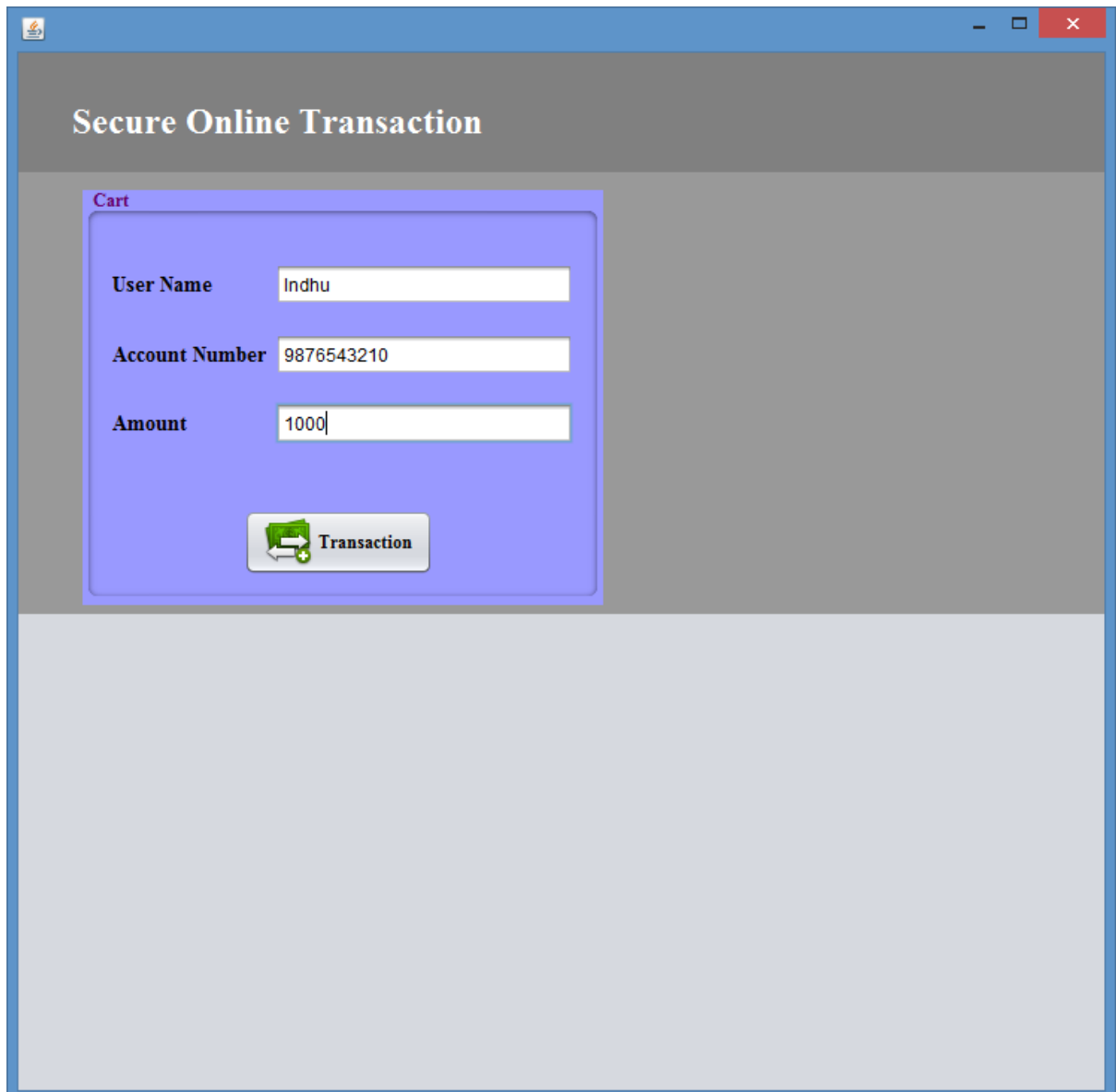
Letter	Buttons
R (Red)	1, 2, 3
G (Green)	2, 3, 1
Y (Yellow)	3, 1, 2
B (Blue)	4, 5, 6
W (White)	5, 6, 4
L (Purple)	6, 4, 5
P (Pink)	9, 7, 8
O (Orange)	7, 8, 9
V (Dark Purple)	8, 9, 7
A (Cyan)	*, 0, #
M (Maroon)	0, #, *
S (Grey)	#, *, 0

To the right of the keyboard grid are two buttons: "Submit" and "Quit".

Fig 6.4 Virtual Keypad for OTP Verification

6.2.5 TRANSACTION MODULE

This module can be used to perform secure electronic fund transfer, among users. The system requires that both participants are registered with this system before transferring funds. The minimum amount that can be transferred is Rs 35.



A screenshot of a web application window titled "Secure Online Transaction". The window has a blue title bar with standard Windows controls. The main content area has a grey background. A purple-bordered box labeled "Cart" is positioned on the left. Inside this box, there are three input fields: "User Name" with the value "Indhu", "Account Number" with the value "9876543210", and "Amount" with the value "1000". Below these fields is a button with a green icon of a document with a plus sign and the text "Transaction".

Secure Online Transaction

Cart

User Name

Account Number

Amount


 **Transaction**

Fig 6.5 Fund Transfer

CHAPTER 7

SYSTEM TESTING

7.1 TYPES OF TESTING

7.1.1 UNIT TESTING

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented produces expected output against given input.

7.1.2 INTEGRATION TESTING

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

7.1.3 FUNCTIONAL TESTING

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the category of black box testing.

7.1.4 SYSTEM TESTING

System testing is performed to ensure that the software still works by putting in different environments. System testing is done with full system implementation and environment. It falls under the category of black box testing.

7.1.5 WHITE BOX TESTING

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called a structural testing and glass box testing. White box testing is used for verification.

7.1.6 BLACK BOX TESTING

Black box testing is attesting technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing. it is used for validation.

7.1.7 ACCEPTANCE TESTING

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the expected. It falls under the category of black box testing.

7.2 TEST RESULTS

All the above types of testing were performed and no defect is found.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Nowadays, electronic transactions are becoming very common. PIN is a popular mode of authentication. However, the traditional PIN Entry method is vulnerable to a variety of attacks such as Guessing Attacks, Human Shoulder Surfing Attacks (SSA), Hidden Cameras, Smudge Attacks, Side-Channel Attacks and many more. Many solutions have been proposed to overcome these attacks. However, these methods proposed require complex calculations and create extra cognitive workload. These techniques also require the user to remember extra information, while overcoming only a few of the above mentioned attacks. The DualPad solution proposed here combines the technique of scrambling the user's keypad and an OTP generation mechanism. This method does not require the user to remember any additional information and does not require any complex calculations. This method also overcomes a majority of the above mentioned attacks. The method proposed, provides more security and also requires less time for authentication as compared to other systems.

8.2 FUTURE ENHANCEMENT

In the future, only a limited number of wrong attempts per user is allowed. After a predefined number of wrong attempts, the user's hidden code is regenerated. This would eliminate the possibility of success in guessing attacks, thus further enhancing the security of the system.

APPENDIX –I SOURCE CODE

START.java

```
package atm;

public class Start extends javax.swing.JFrame {

    public Start() {

        initComponents();

    }

    @SuppressWarnings("unchecked")

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        this.setVisible(false);

        new AtmLogin().setVisible(true);

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        this.setVisible(false);

        new otp.OtpLogin().setVisible(true);

    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {

                new Start().setVisible(true);

            }

        });

    }

}
```

```

        }

    });

}

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

// End of variables declaration

}

```

IPINENTER1.java

```

package atm;

import static DB.database.con;

import static DB.database.rs;

import java.awt.*;

import java.sql.*;

import java.util.*;

```

```

public class IPINENTER1 extends javax.swing.JFrame {

    Statement st=null;

    ResultSet rs=null;

    private String tmp;

    private String otp1 = "";

    private String location = "";

    String email;

    String name;

    private JButton[] buttons;

    private Component frame;

    private String loc;

    public IPINENTER1() {

        initComponents();

        buttons = new JButton[10];

        buttons[0] = jButton15;

        buttons[1] = jButton16;

        buttons[2] = jButton17;

        buttons[3] = jButton18;

        buttons[4] = jButton19;

        buttons[5] = jButton20;

```

```

    buttons[6] = jButton21;

    buttons[7] = jButton22;

    buttons[8] = jButton23;

    buttons[9] = jButton24;

    String[] solutionArray = {"1","2","3","4","5","6","7","8","9","0"};

    shuffleArray(solutionArray);

    jButton15.setText(solutionArray[0]);

    jButton16.setText(solutionArray[1]);

    jButton17.setText(solutionArray[2]);

    jButton18.setText(solutionArray[3]);

    jButton19.setText(solutionArray[4]);

    jButton20.setText(solutionArray[5]);

    jButton21.setText(solutionArray[6]);

    jButton22.setText(solutionArray[7]);

    jButton23.setText(solutionArray[8]);

    jButton24.setText(solutionArray[9]);

}

private static void shuffleArray(String[] ar) {

    Random rnd = new Random();

    for (int i = ar.length-1 ; i > 0; i--)

```

```

{

    int index = rnd.nextInt(i + 1);

    String a = ar[index];

    ar[index] = ar[i];

    ar[i] = a;

}

}

public void shuffleButtons() {

    if (buttons != null) {

        Collections.shuffle(Arrays.asList(buttons));

    }

}

@SuppressWarnings("unchecked")

private void jPanel3AncestorAdded(javax.swing.event.AncestorEvent evt) {

    String nm=DB.database.getname();

    jTextField1.setText(nm);

}

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {

    jTextField2.getText();

}

```



```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    String clear=jTextField2.getText();

    int clr=clear.length();

    String newtxt=clear.substring(0, clr-1);

    jTextField2.setText(newtxt);

}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jButton15ActionPerformed(java.awt.event.ActionEvent evt) {

    String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

    else

    {

        String one= jButton15.getText();

        String val1=val+one;

        jTextField2.setText(val1);
    }
}

```

```
} }
```

```
private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {  
  
String val=jTextField2.getText();  
  
    int l=val.length();  
  
    if(l>=4){  
  
        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four  
digit","error", JOptionPane.ERROR_MESSAGE);  
  
    }  
  
    else  
  
    {  
  
        String one= jButton16.getText();  
  
        String val1=val+one;  
  
        jTextField2.setText(val1);  
  
    } }
```

```
private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {  
  
String val=jTextField2.getText();  
  
    int l=val.length();  
  
    if(l>=4){  
  
        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four  
digit","error", JOptionPane.ERROR_MESSAGE);  
  
    } }
```

```

    }

else

{

    String one= jButton17.getText();

    String val1=val+one;

    jTextField2.setText(val1);

}      }

private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

else

{

    String one= jButton18.getText();

    String val1=val+one;

    jTextField2.setText(val1);

}      }

private void jButton19ActionPerformed(java.awt.event.ActionEvent evt) {

```

```

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

    else

    {

        String one= jButton19.getText();

        String val1=val+one;

        jTextField2.setText(val1);

    }    }

private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

    else

    {

```

```

        String one= jButton20.getText();

        String val1=val+one;

        jTextField2.setText(val1);
    }
}

private void jButton21ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

    else

    {

        String one= jButton21.getText();

        String val1=val+one;

        jTextField2.setText(val1);

    }
}

private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

```

```
JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four  
digit","error", JOptionPane.ERROR_MESSAGE);
```

```
}
```

```
else
```

```
{
```

```
String one= jButton22.getText();
```

```
String val1=val+one;
```

```
jTextField2.setText(val1);
```

```
}      }
```

```
private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
String val=jTextField2.getText();
```

```
int l=val.length();
```

```
if(l>=4){
```

```
JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four  
digit","error", JOptionPane.ERROR_MESSAGE);
```

```
}
```

```
else
```

```
{
```

```
String one= jButton23.getText();
```

```
String val1=val+one;
```

```
jTextField2.setText(val1);
```

```

    }
}

private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    int l=val.length();

    if(l>=4){

        JOptionPane.showMessageDialog(frame,"PIN No Can't Exceed four
digit","error", JOptionPane.ERROR_MESSAGE);

    }

    else

    {

        String one= jButton24.getText();

        String val1=val+one;

        jTextField2.setText(val1);

    }
}

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);
}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {

String val=jTextField2.getText();

    String nm=DB.database.getname();

    try {

Class.forName("com.mysql.jdbc.Driver");

```

```

con=DriverManager.getConnection("jdbc:mysql://localhost:3308/prj","root","root");

st = con.createStatement();

rs = st.executeQuery("select name,pin,email,location from tbl_registration");

while(rs.next())

    {

        name=rs.getString("name");

        if(nm.equals(name))

        {

            loc=rs.getString("pin");

            location=rs.getString("location");

        }

        if(nm.equals(name))

        {

            email=rs.getString("email");

            location=rs.getString("location");

            String[] solutionArray = {"R", "G", "B", "Y", "P", "W",
"V", "O", "A", "S", "L", "M"};

            String[] locationArray={"L", "M", "R"};

            DB.database.shuffleArray(solutionArray);

            DB.database.locationArray(locationArray);

```



```

        for (int i = 0; i < 4; i++) {

            tmp = solutionArray[i];

            otp1 = otp1 + tmp;

        }

Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3308/prj","root","root");

st = con.createStatement();

String sql=" Update tbl_registration set otp="+otp1+" where
name='"+name+"'";

st.executeUpdate(sql);

}catch(Exception e){

    System.out.println(e);

}

finally{

    try {

        st.close();

        con.close();

    } catch (SQLException ex) {

```

```
        //Logger.getLogger(PinEnter.class.getName()).log(Level.SEVERE,
null, ex);

    }
}
```

TRANSACTION.java

```
package atm;

import java.awt.*;

import static java.*;

import java.sql.*;

import java.util.*;

public class Transaction extends javax.swing.JFrame {

    Connection con=null;

    Statement st=null;

    ResultSet rs=null;

    Connection con1=null;

    Statement st1=null;

    ResultSet rs1=null;

    private Component frame;

    String val1;

    int val2;

    String s;

    public Transaction() {
```

```

        initComponents();
    }

    @SuppressWarnings("unchecked")

    private void jPanel3AncestorAdded(javax.swing.event.AncestorEvent evt) {

        String nm=DB.database.getname();

        jTextField1.setText(nm);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        val1=jTextField2.getText();

        val2=parseInt(jTextField5.getText());

        try {

            if(val1.isEmpty())

            {

                JOptionPane.showMessageDialog(null, " Enter proper account number
", "Error", JOptionPane.ERROR_MESSAGE);

                jTextField2.grabFocus();

                jTextField2.selectAll();

                return;

            }

            if(val2<35){

```

```

        JOptionPane.showMessageDialog(null, " Enter proper amount ", "Error",
JOptionPane.ERROR_MESSAGE);

        jTextField5.grabFocus();

        jTextField5.selectAll();

        return;

    }

    else{

        Class.forName("com.mysql.jdbc.Driver");
con1=DriverManager.getConnection("jdbc:mysql://localhost:3308/prj","root","r
oot");

st1 = con1.createStatement();

rs1 = st1.executeQuery("select account from tbl_registration");

while(rs1.next())

{

s = rs1.getString("account");

}

if(val1.equals(s))

{

JOptionPane.showMessageDialog(frame,"Tranaction Success","Success",
JOptionPane.INFORMATION_MESSAGE);

this.setVisible(false);

}

```

```

else{

OptionPane.showMessageDialog(null, " Account doesn't exist", "Error",
OptionPane.ERROR_MESSAGE);

        }

    }

}

    catch (ClassNotFoundException ex) {

        Logger.getLogger(Transaction.class.getName()).log(Level.SEVERE,
null, ex);

        } catch (SQLException ex) {

        Logger.getLogger(Transaction.class.getName()).log(Level.SEVERE,
null, ex);

        }

    }

}

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new Transaction().setVisible(true);

        }

    });

}

```

```
// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField5;

}
```

DB.java

```
package DB;

import java.sql.*;

import java.util.Random;

public class database {

    public static Connection con;

    public static PreparedStatement pst;
```

```

public static ResultSet rs;

public static String name;

private static String email;

public static void getConnection () {

    try{

        Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3308/prj","root","ro
ot");

        }

    catch(Exception e)

        {

            System.out.println("ERROR"+ e);

        }

    }

public static String registration(String nme,String pass,String mail,String
contact,String address,String accno,String pin,String loc) {

    String s="";

    try

    {

        String s1="Insert into tbl_registration
(name,password,email,contactno,address,account,pin,location,status)values(?,?
?,?,?, ?,?,1)";

```

```
PreparedStatement pst1=con.prepareStatement(s1);
```

```
    pst1.setString(1, nme);
```

```
    pst1.setString(2, pass);
```

```
    pst1.setString(3, mail);
```

```
    pst1.setString(4, contact);
```

```
    pst1.setString(5, address);
```

```
    pst1.setString(6, accno);
```

```
    pst1.setString(7, pin);
```

```
    pst1.setString(8, loc);
```

```
    pst1.executeUpdate();
```

```
    pst1.close();
```

```
    con.close();
```

```
    System.out.println("Insertion success");
```

```
        s="OK";
```

```
    }
```

```
catch(SQLException se)
```

```
{
```

```
    s="ERROR"+ se.getMessage();
```

```
}
```

```
return s;
```



```

}

public static String shuffleArray(String[] ar) {

    String p="";

    String tmp="";

    try{

        Random rnd = new Random();

        for (int i = ar.length - 1; i > 0; i--)

        {

            int index = rnd.nextInt(i + 1);

            String a = ar[index];

            ar[index] = ar[i];

            ar[i] = a;

        }

    }

    catch(Exception e)

    {

        p="ERROR"+e;

    }

    return p;

}

```

```

public static String locationArray(String[] ar) {

    String l="";

    String tmp="";

    try{

        Random rnd = new Random();

        for (int i = ar.length - 1; i > 0; i--)

        {

            int index=rnd.nextInt(i+1);

            String a = ar[index];

            ar[index] = ar[i];

            ar[i] = a;

        }

    }

    catch(Exception e)

    {

        l="ERROR"+e;

    }

    return l;

}

public static Boolean checkLogin(String un, String pw) {

```

```

try {

    getConnection();

    Statement st;

    st = con.createStatement();

    ResultSet rs = st.executeQuery("select account,password from
tbl_registration where account='"+un+"' and password='"+pw+"'");

    System.out.println("select account,password from tbl_registration where
account='"+un+"' and password='"+pw+"'");

    if (rs.next()) {

        return true;

    } else {

        return false;

    }

} catch (Exception e) {

    System.out.println("error while validating" + e);

    return false;

}

}

public static String setname(String name)

{

    database.name=name;

```

```

        return name;

    }

    public static String getname(){

        return name;

    }

    public static String setmail(String mail)

    {

        database.email=mail;

        return email;

    }

    public static String getmail(){

        return email;

    }

}

}

}

}

    public static void main(String args[]) {

        SwingUtilities.invokeLater(new Runnable() {

            public void run() {

                new IPINENTER1().setVisible(true);

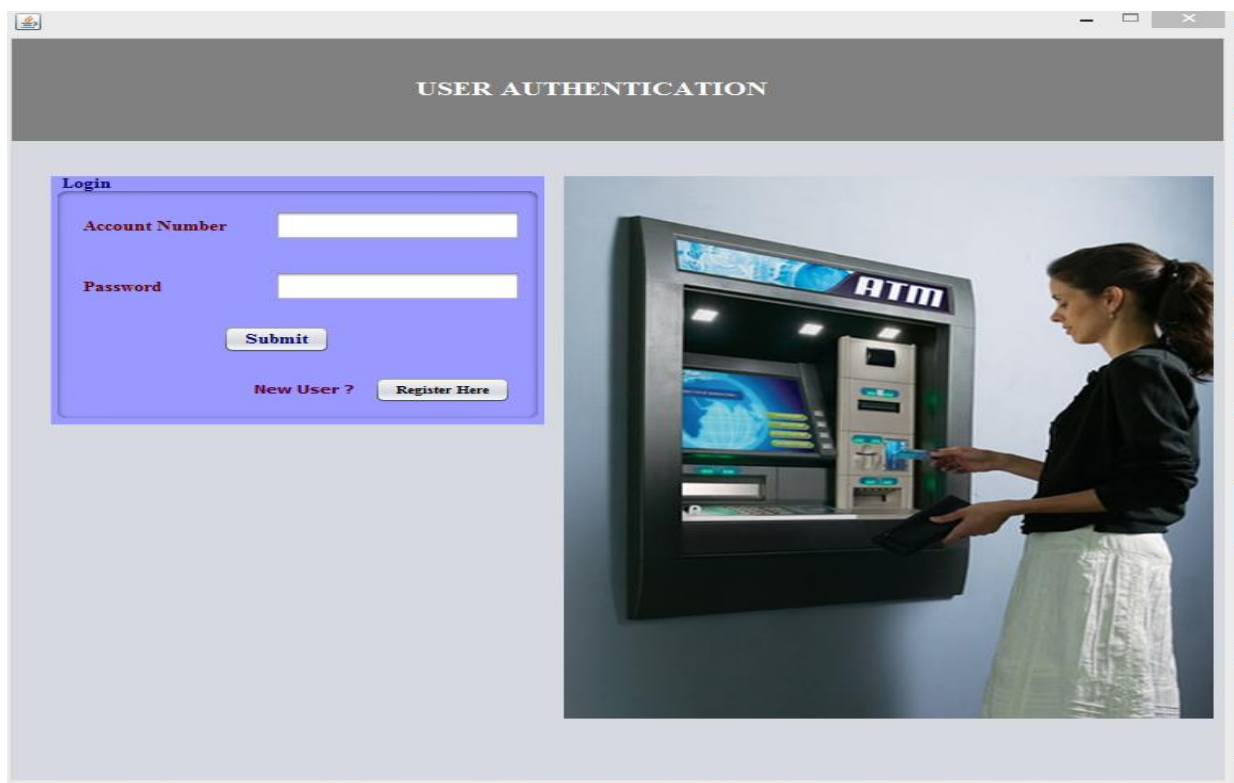
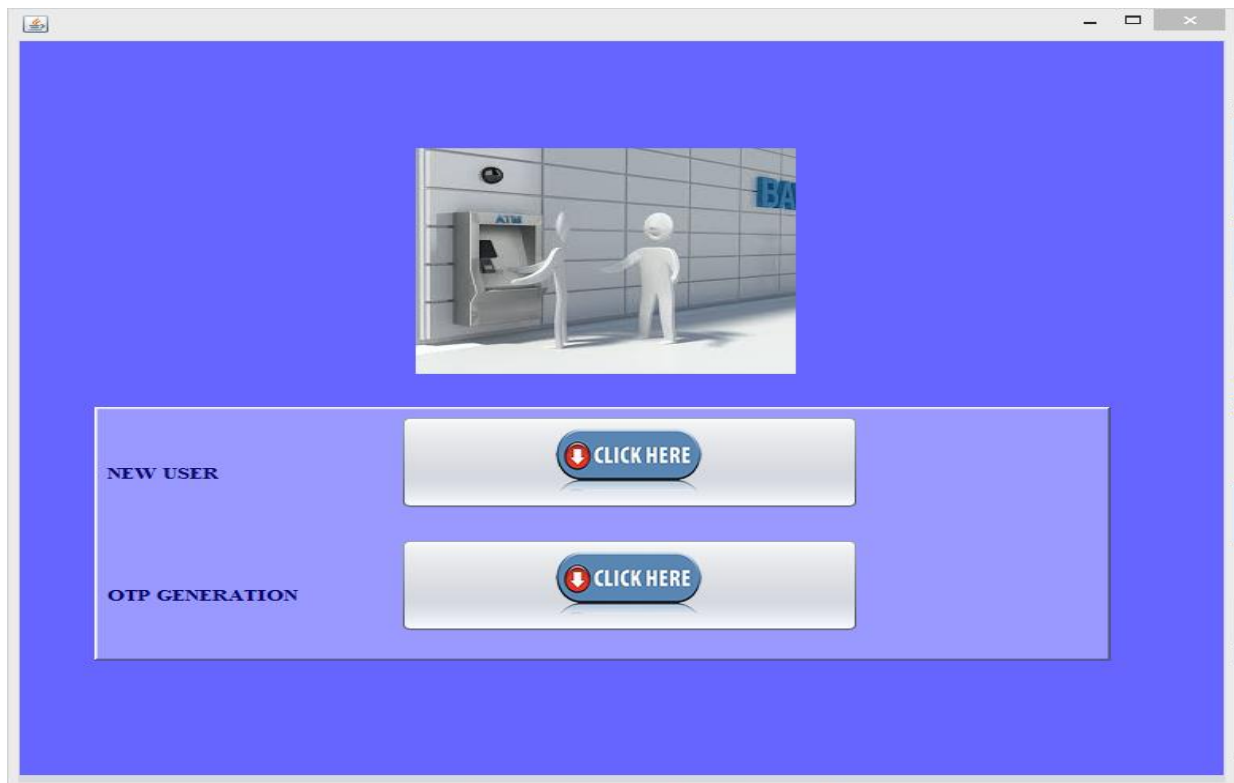
```

}

});

}

APPENDIX-II SCREENSHOTS



Registration

Name

:

Indhu

Password

:

E-Mail

:

indhu@gmail.com

Contact No

:

9080823748

Bank Name

:

CANARA

Account No

:

2323232323

Pin

:

Pin Location

:

Gender

:

☐ MALE ☒ FEMALE


Age

:

21

Register

Reset



USER AUTHENTICATION

Login

Account Number

2323232323

Password

Submit

New User ?

Register Here



USER PIN GENERATION

Please Note Below No.

User Name

PIN Number

VIRTUAL KEYBOARD FOR PIN ENTRY

Details

User Name **PIN No.**

2	4	3
8	1	5
6	9	7
0		

Success - PIN Entry

User Details

Name

:

Indhu

Password

:

indhu

E-Mail

:

indhu@gmail.com

Contact No

:

9876564345

Address

:

Trichy

Account No

:

2323232323

Pin

:


6987

Pin Location

:

L

Continue

 ...

Secure OTP Generation

Check your mail

Mail ID

indhu@gmail.com

Continue

SQLyog Community - [New Connection/prj - root@localhost]

File Edit Favorites Database Table Others Tools Powertools Window Help

root@localhost

- atm
- information_schema
- mysql
- prj
 - Tables
 - tbl_registration
 - Views
 - Stored Procs
 - Functions
 - Triggers
 - Events
 - test

Reasons for upgrading to Professional/Enterprise/Ulimate: No nag screens on startup and shutdown

Query

1 Result 2 Profiler 3 Messages 4 Table Data 5 Info 6 History

First row: 0 100 rows Refresh

	name	password	email	contactno	address	account	pin	location	status	otp			
<input checked="" type="checkbox"/>	Indhu	5B	indhu	5B	indhu@gmail...	15B	9876564345	Trichy	6B	2323232323	6987 L	1	BPRL
<input type="checkbox"/>	Monisha	7B	moni	4B	moni@gmail.com	14B	9080978655	Trichy	6B	4564564564	1530 M	1	MPRA
<input type="checkbox"/>	Ranjani	7B	rani	4B	ranjani@ma...	17B	9789978674	Chennai	7B	6667778889	1038 L	1	LVER
<input type="checkbox"/>	Kaviya	6B	kavi	4B	kavi@gmail.com	14B	9887765563	chennai	7B	6786786789	4925 L	1	PAIM
<input type="checkbox"/>	Amutha	6B	ammu	4B	ammu@gmail.com	14B	8934647846	Trichy	6B	23452345	843 L	1	(NULL)
<input type="checkbox"/>	pattu	5B	pattu	5B	pattu@gmail...	15B	9088767565	Trichy	6B	1231231231	6317 M	1	(NULL)
<input type="checkbox"/>	Sara	4B	sara	4B	sara@gmail.com	14B	9874654745	Chennai	7B	345634563456	6573 M	1	(NULL)
<input type="checkbox"/>	Harini	6B	Harini	6B	harini@mai...	16B	9874657853	Gova	4B	543254325432	3198 R	1	(NULL)
<input type="checkbox"/>	Suriya	6B	suriya	6B	suri@gmail.com	14B	9088767655	Harur	5B	987698769876	5129 R	1	(NULL)
<input type="checkbox"/>	Jaya	4B	jaya	4B	jaya@gmail.com	14B	9876598765	Chennai	7B	9876543210	5914 M	1	(NULL)
*	(NULL)	OK	(NULL)	OK	(NULL)	OK	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Database: prj Table: tbl_registration

http://www.webyog.com 10 row(s) Connections: 1 [Upgrade to SQLyog Professional/Enterprise/Ulimate](#)

REFERENCES

- [1] Mun Kyu Lee, "Security Notions and Advanced Method for Human Shoulder Surfing Resistant PIN Entry", IEEE Transactions on Information Forensics And Security , Vol .9, No.4, Apr i l 2014.
- [2] D. S. Tan, P. Keyani, and M. Czerwinski, "Spy-resistant keyboard: More secure password entry on public touch screen displays," in Proc. 17th Austral. Conf. Comput. Human Interaction OZCHI, 2005, pp. 1–10.
- [3] Gerson De Souza Faria, HaeYong Kim, "Identification of pressed keys from mechanical vibrations", 2013.
- [4] C. S. Kim and M.-K. Lee, "Secure and user friendly PIN entry method," in Proc. 28th Int. Conf. Consum. Electron., 2010, p. 5.1–1.
- [5] A. D. Luca, E. von Zezschwitz, L. Pichler, and H. Hussmann, "Using fake cursors to secure on-screen password entry," in Proc. CHI, 2013, pp. 2399–2402.
- [6] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd, "Reducing shoulder-surfing by using gaze-based password entry," in Proc. SOUPS, 2007, pp. 13–19.
- [7] Q. Yan, J. Han, Y. Li, J. Zhou, and R. H. Deng, "Designing leakage resilient password entry on touchscreen mobile devices," in Proc. ASIACCS, 2013, pp. 37–48.
- [8] Jong-Hoon Kim, Gokarna Sharma, Irrin Steve Cardenar, Do Yeon Kim, Nagarajan Prabakar, S.S.Iyengar, "DynamicPIN: A Novel Approach Towards Secure Atm Authentication", 2017.
- [9] Hao-Jun Xu, Yu-Xuan Dan, Wei-Chi Ku, "An Observation Attacks Resistant PIN-Entry Scheme using localised Haptic Feedback", 2016.

[10] Geetika Kovelamudi, Jun Zheng, Srinivas Mukkamala, “A Study of The Security And Usability of Scramble Keypad For Pin Unlock on Smartphones”, 2017.

[11] A.D.Luca, K.Hertzschuch, and H.Husmann, “ ColorPIN: Securing PIN entry through indirect input” , in Proc. CHI , 2010.

[12] Ojaswi K. Kasat, Umesh S. Bhadade, “Revolving Flywheel Pin Entry Method to Prevent Shoulder Surfing Attacks”, 2018 3rd Int. Conf.