****

***“PASSWORD MANAGER”***

**Mini Project Report**

**Submitted in partial fulfillment of the requirements of the subject Object Oriented Programming Method lab**

**by**

**Harsh Katkade (SE-18)**

**Mohil Khare (SE-20)**

**Vaibhav Lakhani (SE-24)**

Supervisor

**Prof.Bhagyashree Madan**



**Department of Computer Engineering**

**K.J. Somaiya Institute of Engineering and Information Technolog**

**Ayurvihar, Sion Mumbai-400022**

**2019**

**

**

*This is to certify that the project entitled* ***“Password Manager”*** *is a bona fide work of Harsh Katkade , Mohil Khare ,Vaibhav Lakhani submitted as mini project in the subject of OOPM*  ***Lab*** *in* ***“Computer Engineering”.***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Prof. Bhagyashree Madan

(Project Guide)

**DECLARATION**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Harsh Katkade \_\_\_\_\_\_\_\_\_\_\_\_

Mohil Khare \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Vaibhav Lakhani\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:

**ACKNOWLEDGEMENT**

Before presenting out our project entitled “Password Manager”, we would like to convey our sincere thanks to many people who guided us throughout the course for this seminar work.

First, we would like to express our sincere thanks to our beloved Principal **Dr. SURESH UKARANDE** for providing various facilities to carry out this report.

We would like to express our sincere thanks to **Prof. BHAGYASHREE MADAN** for her guidance, encouragement, co-operation and suggestions given to us at progressing stages of report.

Finally, we would like to thank our **H.O.D. Prof. SARITA AMBADEKAR** and all teaching, non-teaching staff of the college and friends for their moral support rendered during the course of the report work and for their direct and indirect involvement in the completion of our report work, which made our endeavor fruitful.

Harsh Katkade

Mohil Khare

Vaibhav Lakhani

**INDEX**

|  |  |  |
| --- | --- | --- |
| SERIAL NO | TITLE | PAGE NO |
| 1) | PROBLEM DEFINITION | 1 |
| 2) | THEORY | 1 |
| 3) | ALGORITHM | 10 |
| 4) | OUTPUT | 17 |
| 5) | CONCLUSION | 22 |
| 6) | REFERENCE | 23 |

**PROBLEM DEFINITION**

Create a Graphical User Interface(GUI) using Java Swing to implement and store passwords . First, ask the user to create an account where he/she can save his/her all other passwords Enter the details along with password and stay stress free from remembering your password

**THEORY**

**WHAT IS A GUI?**

A graphical user interface (GUI) is a human-computer interface (i.e., a way for humans to interact with computers) that uses [windows](http://www.linfo.org/window.html), [icons](http://www.linfo.org/icon.html) and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well).

**Advantages of GUIs**

1)A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use. For example, it is much easier for a new user to move a file from one directory to another by dragging its icon with the mouse than by having to remember and type seemingly arcane commands to accomplish the same task.

2)Adding to this intuitiveness of operation is the fact that GUIs generally provide users with immediate, visual feedback about the effect of each action. For example, when a user deletes an icon representing a file, the icon immediately disappears, confirming that the file has been deleted (or at least sent to the trash can). This contrasts with the situation for a CLI, in which the user types a delete command (inclusive of the name of the file to be deleted) but receives no automatic feedback indicating that the file has actually been removed.

3)In addition, GUIs allow users to take full advantage of the powerful [multitasking](http://www.linfo.org/multitasking.html) (the ability for multiple programs and/or multiple instances of single programs to run simultaneously) capabilities of modern [operating systems](http://www.linfo.org/operating_systems_list.html) by allowing such multiple programs and/or instances to be displayed simultaneously. The result is a large increase in the flexibility of computer use and a consequent rise in user productivity.

4)But the GUI has became much more than a mere convenience. It has also become the standard in human-computer interaction, and it has influenced the work of a generation of computer users. Moreover, it has led to the development of new types of applications and entire new industries. An example is desktop publishing, which has revolutionized (and partly wiped out) the traditional printing and typesetting industry.

**SWING AS A GUI**

Swing is a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit) for [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It is part of [Oracle](https://en.wikipedia.org/wiki/Oracle_Corporation)'s [Java Foundation Classes](https://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) – an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for Java programs

Swing was developed to provide a more sophisticated set of GUI [components](https://en.wikipedia.org/wiki/Software_component) than the earlier [Abstract Window Toolkit (AWT)](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit). Swing provides a [look and feel](https://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](https://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

**ADVANTAGES OF SWING**

* single-[threaded](https://en.wikipedia.org/wiki/Thread_(computing)) programming model
* Swing is platform-independent because it is completely written in Java.
* Extensible
* Configurable
* Lightweight UI

### **Relationship to AWT**

In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system.

By contrast, Swing components are often described as lightweight because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as heavyweight components.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing ([JApplet](https://docs.oracle.com/javase/10/docs/api/javax/swing/JApplet.html), [JDialog](https://docs.oracle.com/javase/10/docs/api/javax/swing/JDialog.html), [JFrame](https://docs.oracle.com/javase/10/docs/api/javax/swing/JFrame.html), and [JWindow](https://docs.oracle.com/javase/10/docs/api/javax/swing/JWindow.html)) extend an AWT top-level container.

**The Delegation Event Model**The modern approach to handling events is based on the delegation event model, which defines standard and consistent mechanisms to generate and process events. Its concept is quite simple: a source generates an event and sends it to one or more listeners. In this scheme, the listener simply waits until it receives an event. Once an event is received, the listener processes the event and then returns. Listeners must register with a source in order to receive an event notification. This provides an important benefit: notifications are sent only to listeners that want to receive them. The advantage of this design is that the application logic that processes events is cleanly separated from the user interface logic that generates those events.

Events

In the delegation model, an event is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a graphical user interface. Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list,etc. You are free to define events that are appropriate for your application.

Event SourcesA source is an object that generates an event. This occurs when the internal state of that object changes in some way. A source must register listeners in order for the listeners to receive notifications about a specific type of event. The general form is:

public void addTypeListener(TypeListener el)

Here, Type is the name of the event, and el is a reference to the event listener.

When an event occurs, all registered listeners are notified and receive a copy of the event object.

Event ListenersA listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications. The methods that receive and process events are defined in a set of interfaces found in java.awt.event.

Event Classes

The classes that represent events are at the core of Java’s event handling mechanism. The root of the Java event class hierarchy is EventObject, which is in java.util. The class AWTEvent, defined within the java.awt package, is a subclass of EventObject.

• EventObject is a superclass of all events.

• AWTEvent is a superclass of all AWT events that are handled by the delegation event model.

**ACTION EVENTS**

The Action Event ClassActionEvent is generated when a button is pressed, a list item is double-clicked, or a menu item is selected. There is an integer constant, ACTION\_PERFORMED, which can be used to identify action events.The type of the event is specified by type, and its command string is cmd.The parameter specifies when the event occurred. You can obtain the command name for the invoking ActionEvent object by using the getActionCommand( ) method, shown here:String getActionCommand( )

For example, when a button is pressed, an action event is generated that has a command name equal to the label on that button.

The ActionListener InterfaceThis interface defines the actionPerformed( ) method that is invoked when an action event occurs. Its general form is shown here:void actionPerformed(ActionEvent ae)

**Sources of Events**

|  |  |
| --- | --- |
| Event Source | Description |
| Button | Generates action events when the button is pressed |
| Check Box | Generates item events when the check box is selected or deselected |
| Choice | Generates item events when the choice is changed |
| List | Generates action events when an item is double-clicked; generates item events when an item is selected or deselected |
| Menu Item | Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected. |
| Scroll bar | Generates adjustment events when the scroll bar is manipulated. |
| Text components | Generates text events when the user enters a character. |
| Window | Generates window events when a window is activated, closed, deactivated,deiconified, iconified, opened, or quit. |

**DATABASE**

A database is an organized collection of data. A database-management system (DBMS) is a computer-software application that interacts with end-users, other applications, and the database itself to capture and analyze data. A general-purpose DBMS allows the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MS-Access,MySQL, PostgreSQL, EnterpriseDB, MongoDB, MariaDB, Microsoft SQL Server, Oracle, Sybase, SAP HANA, MemSQL, SQLite and IBM DB2.

The DBMS provides various functions that allow entry, storage and retrieval of large quantities of information and provides ways to manage how that information is organized.

Existing DBMSs provide various functions that allow management of a database and its data which can be classified into four main functional groups:

Data definition –

Creation, modification and removal of definitions that define the organization of the data.

Update –

Insertion, modification, and deletion of the actual data.

Retrieval –

Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database.

Administration –

Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure.

**MS-Access**

What is MS-Access?

Microsoft Access is a Database Management System (DBMS) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications, included in the professional and higher editions.

Main Features of MS-Access

* Microsoft Access is just one part of Microsoft’s overall data management product strategy.
* It stores data in its own format based on the Access Jet Database Engine.
* Like relational databases, Microsoft Access also allows you to link related information easily.
* It can also import or link directly to data stored in other applications and databases.
* As its name implies, Access can work directly with data from other sources, including many popular PC database programs, with many SQL (Structured Query Language) databases on the desktop, on servers, on minicomputers, or on mainframes, and with data stored on Internet or intranet web servers.
* Access can also understand and use a wide variety of other data formats, including many other database file structures.
* You can export data to and import data from word processing files, spreadsheets, or database files directly.
* Access can work with most popular databases that support the Open Database Connectivity (ODBC) standard, including SQL Server, Oracle, and DB2.
* Software developers can use Microsoft Access to develop application software.

Creating a database

1)create database user\_defined\_name;

2)use database\_name;

3)create table table\_name(variable\_name data\_type)

Inserting into database

insert into table\_name values(parameter\_list);

Displaying

Select \* from table\_name;

**UCanAccess**

What is UCanAccess?

Java has included a JDBC/ODBC Bridge driver as a transitional solution for accessing ODBC data sources, but it has always been considered a very limited driver and the recommendation has always been to use a pure JDBC driver instead. Starting with Java 8, the bridge driver is no longer provided.

UCanAccess is a pure Java JDBC Driver implementation which allows Java developers and JDBC client programs to read/write Microsoft Access database (.mdb and .accdb) files. No ODBC needed.

Main Features of UCanAccess

* Supports Access Formats from 2000
* SELECT, INSERT, UPDATE, DELETE statements. Transactions and savepoints.
* Core built-in functions for Access SQL are supported (e.g., Date(), Now(), IIf(), ...). Read and write support to complex types (i.e., array of Version, Attachment, SingleValue).
* Access data types: YESNO, BYTE, INTEGER, LONG, SINGLE, DOUBLE, NUMERIC, CURRENCY, COUNTER, TEXT, OLE, MEMO, GUID, DATETIME.
* Both double quote " and single quote ' as SQL string delimiters.
* You can run SQL commands and display their results. CSV export command included.

**TextLocal**

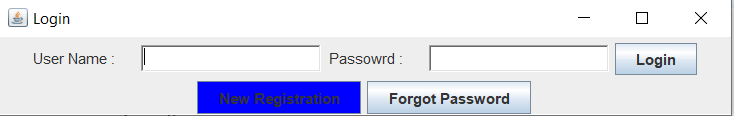
What TextLocal?

TextLocal is a software company which allows users to send Bulk Messages for promotion/Business Growth. It also provides an inbuild API through which Java programs can send message to user at a minimal rate by just using an API Key.

**ALGORITHM**

* Start
* The user enters his Login ID and Password.
* If the user forgets his password he hits “Forget Password Button”.
* An OTP is sent which will be verified and will make the user change his password
* After verifying the user name and password stored in MS-Access Database the user will gain Access of the Password Manager Table where he can add, delete, update his passwords.
* If a user is a New User he clicks on “Registration” which will let him make his account
* Stop.

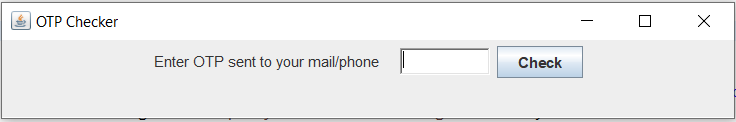
**OUTPUT**



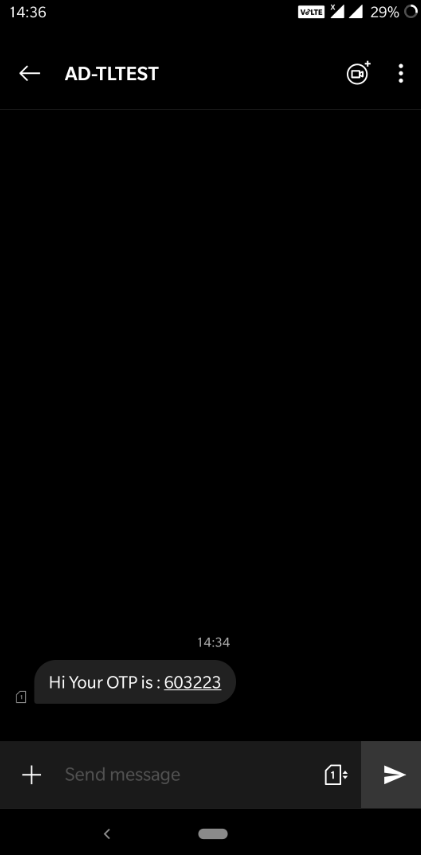
Assuming User has Forgot his Password



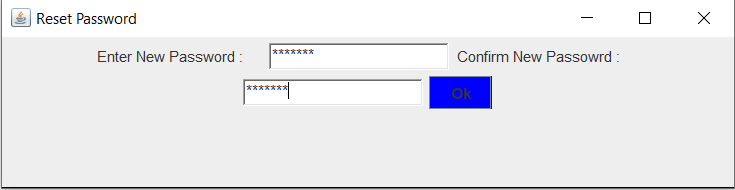
User Name is Checked and OTP is sent to Mail as well as SMS



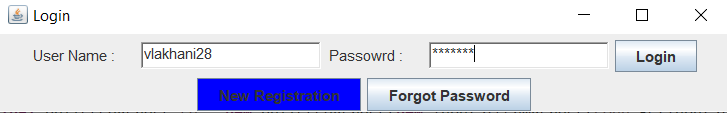




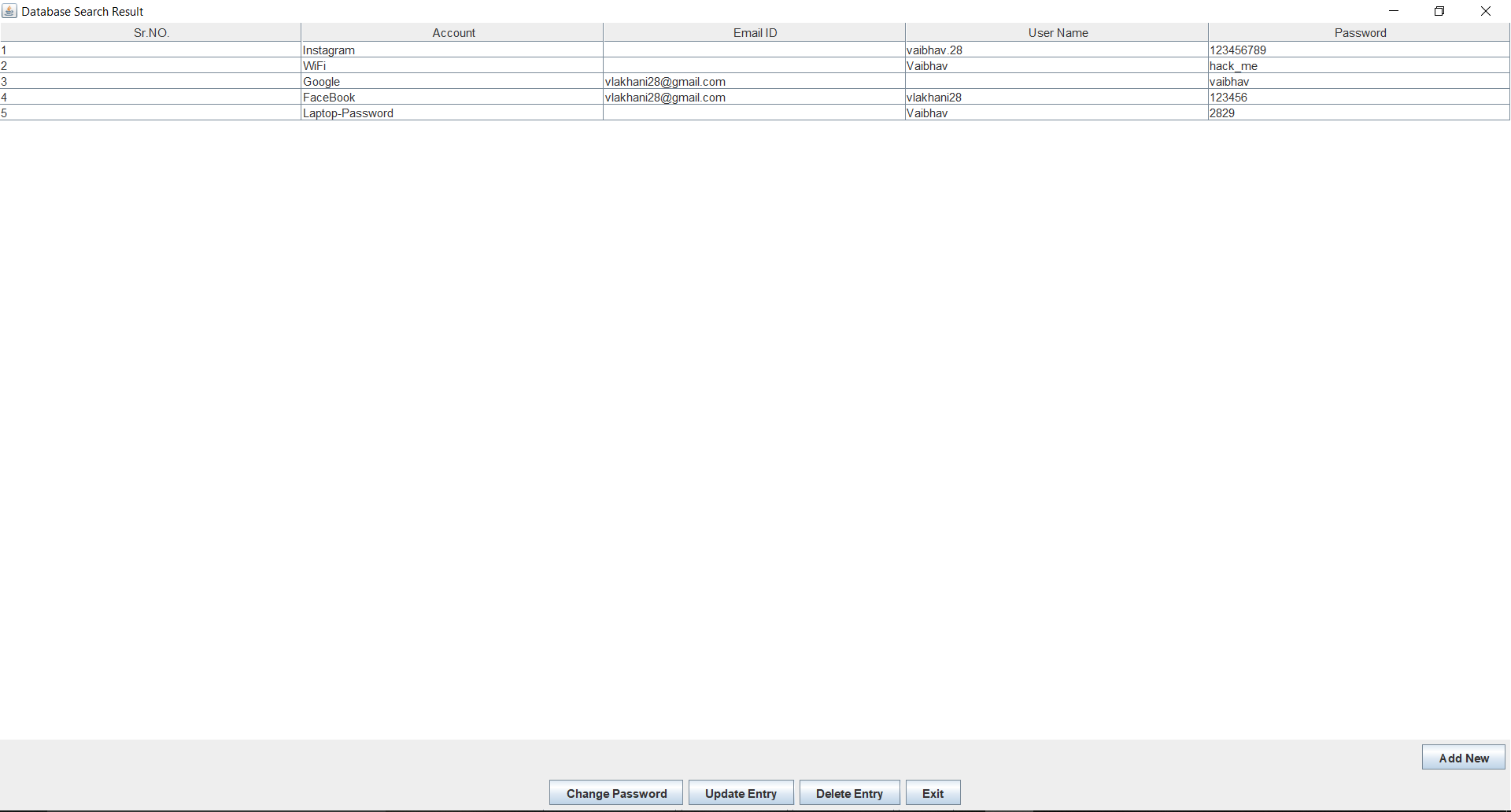
Once the OTP is checked it shows to enter new password



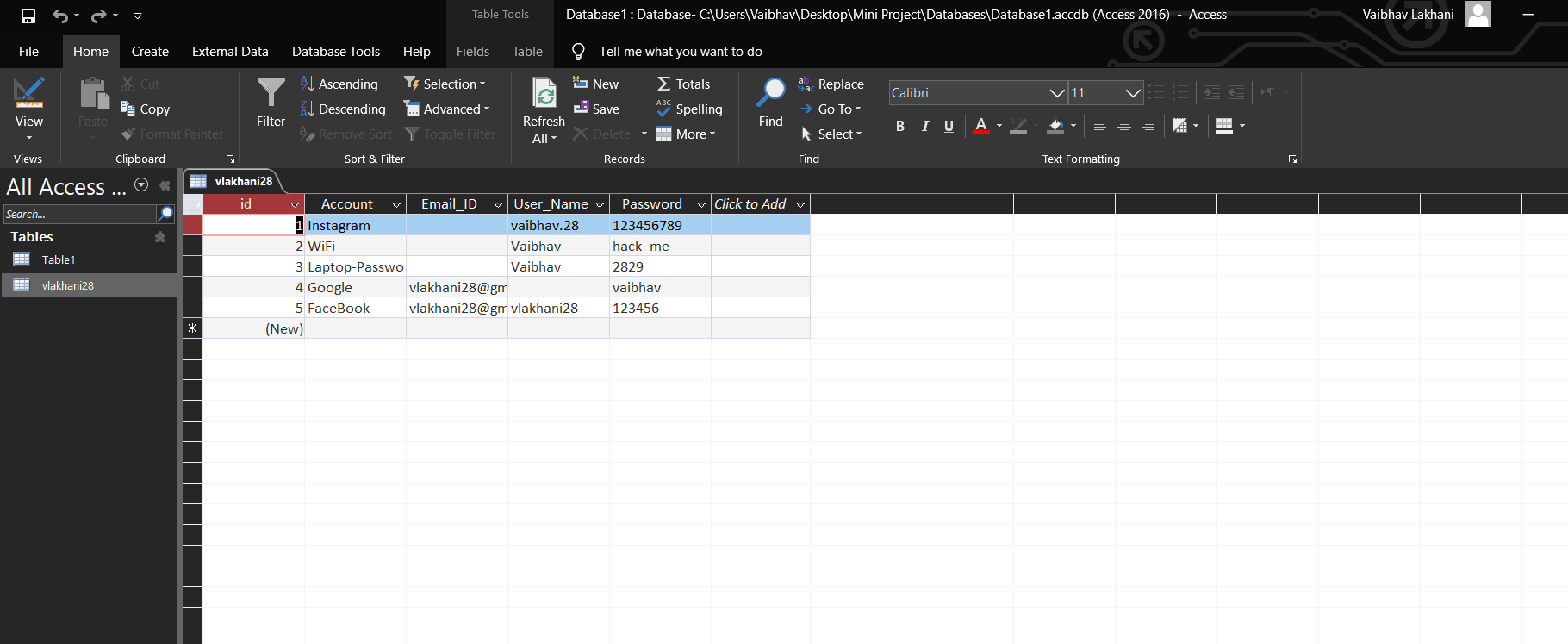
Once the password is reset it shows the login screen again to re-login with new password



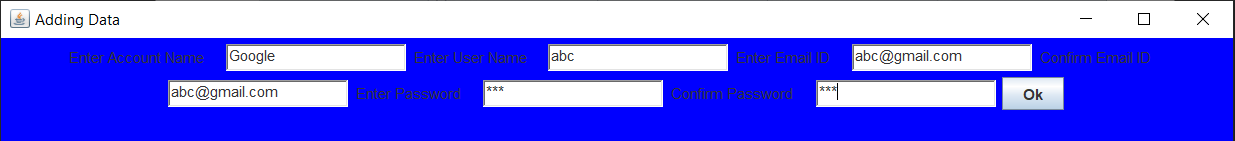


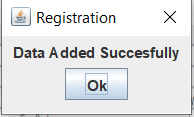






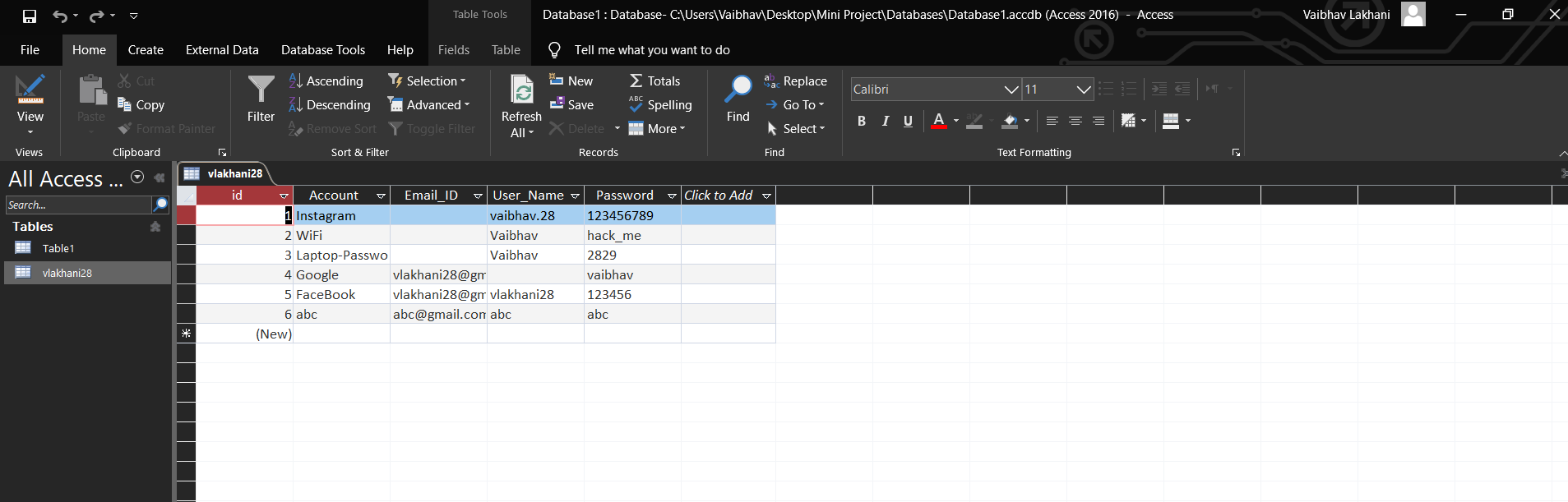
Now Lets us add Some data



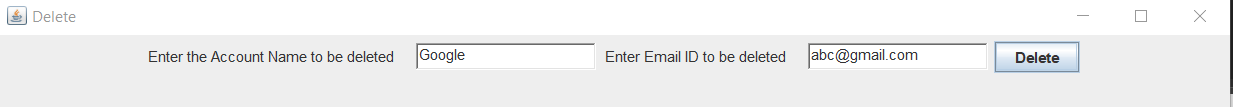


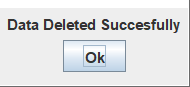
The Password Manager Table and MS Access Table gets updated



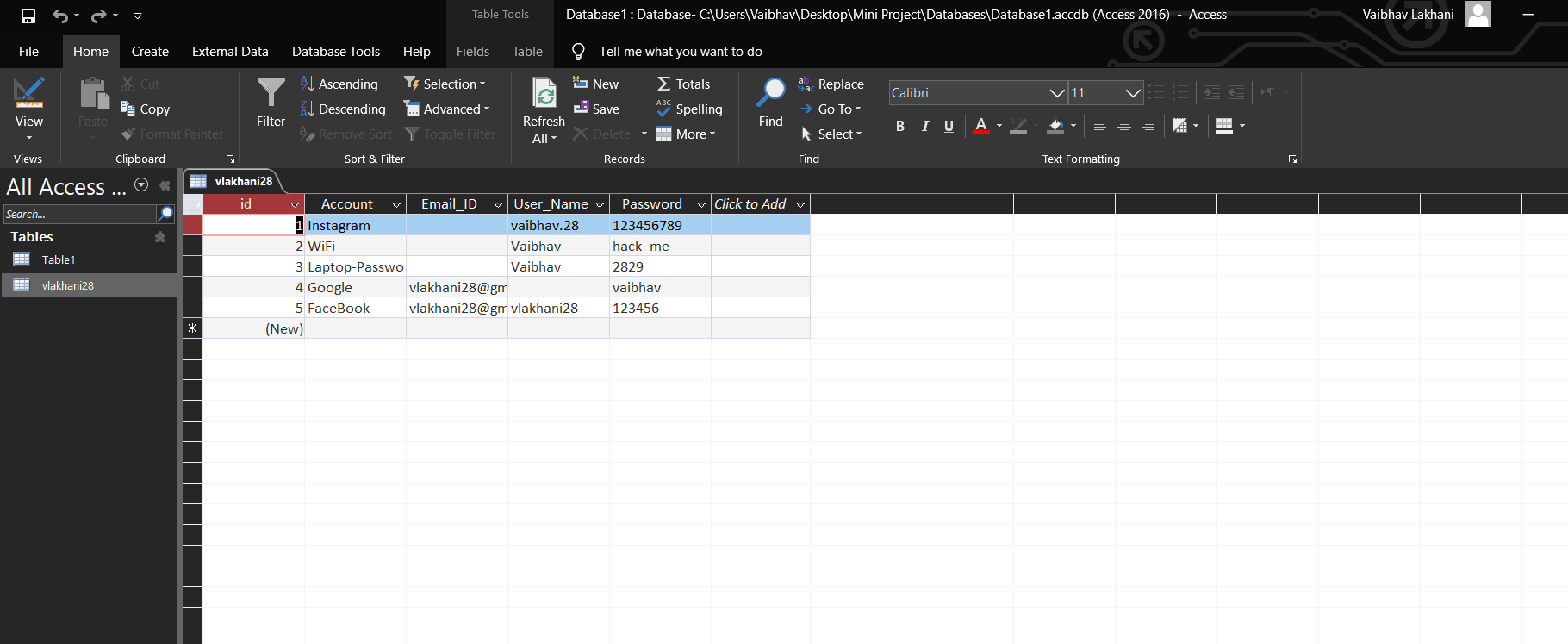


Now Lets us delete Some data

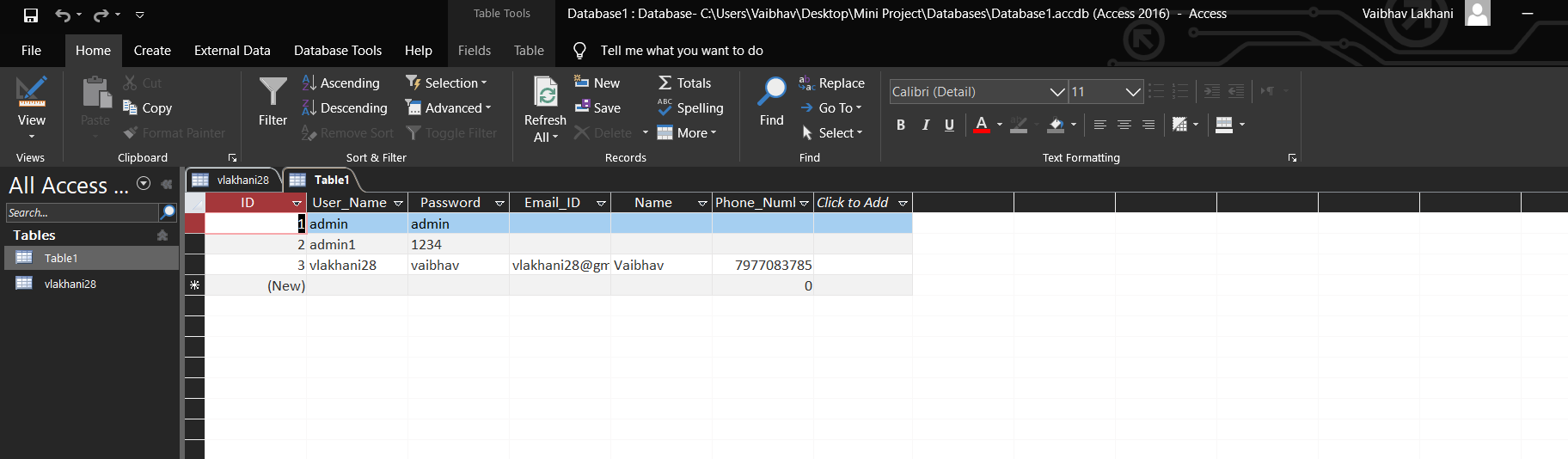




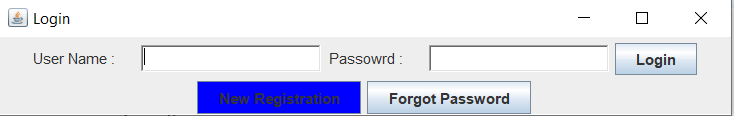


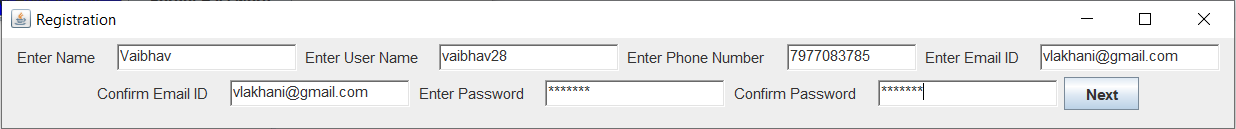


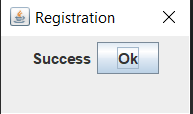
**Registered Users Table Database**



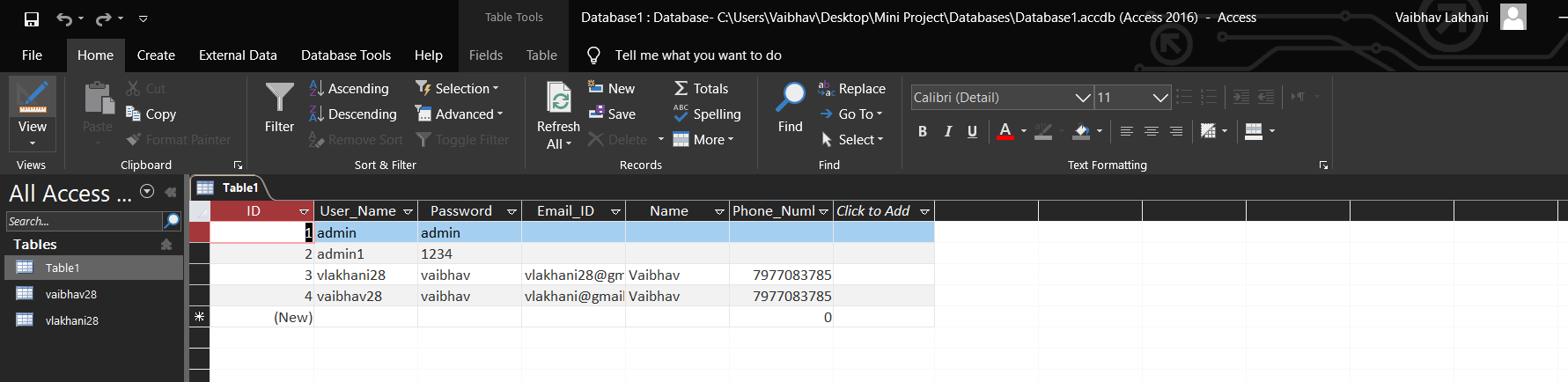
**New Registration Process**

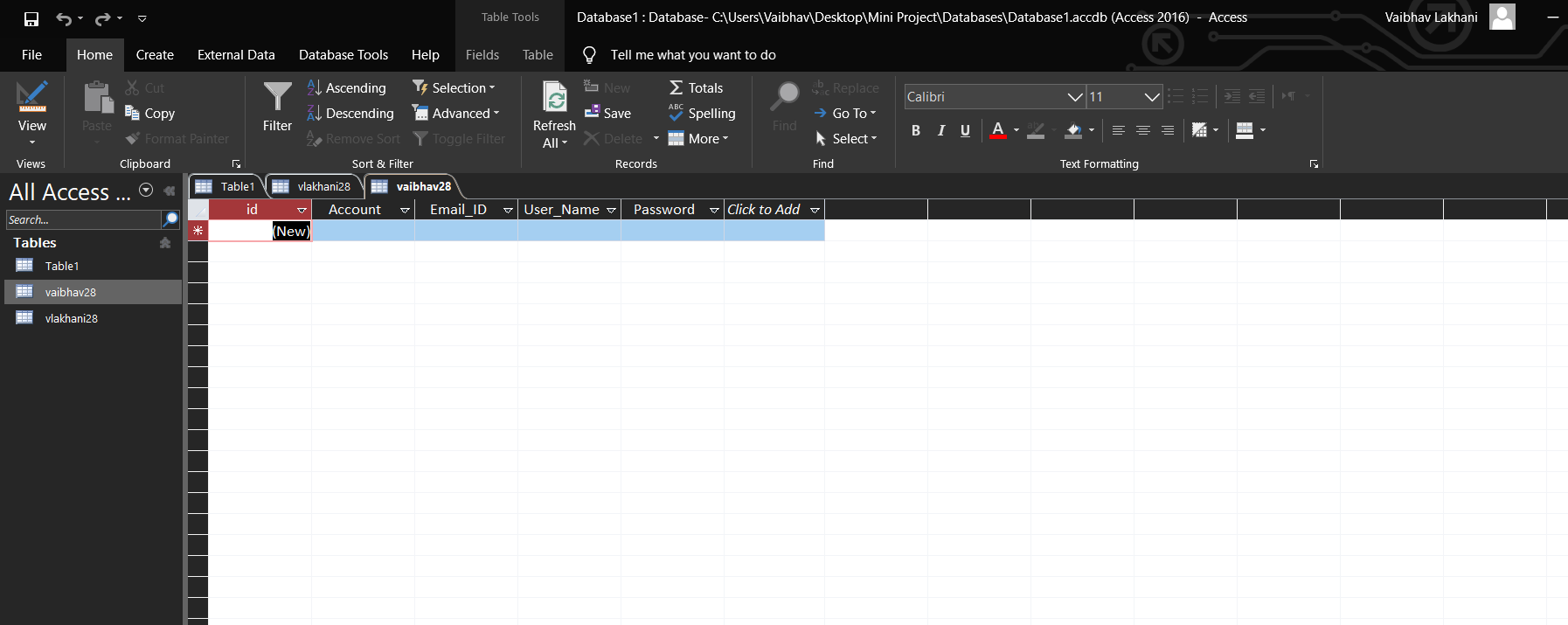




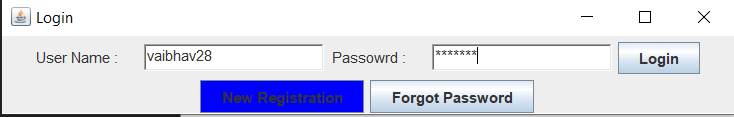


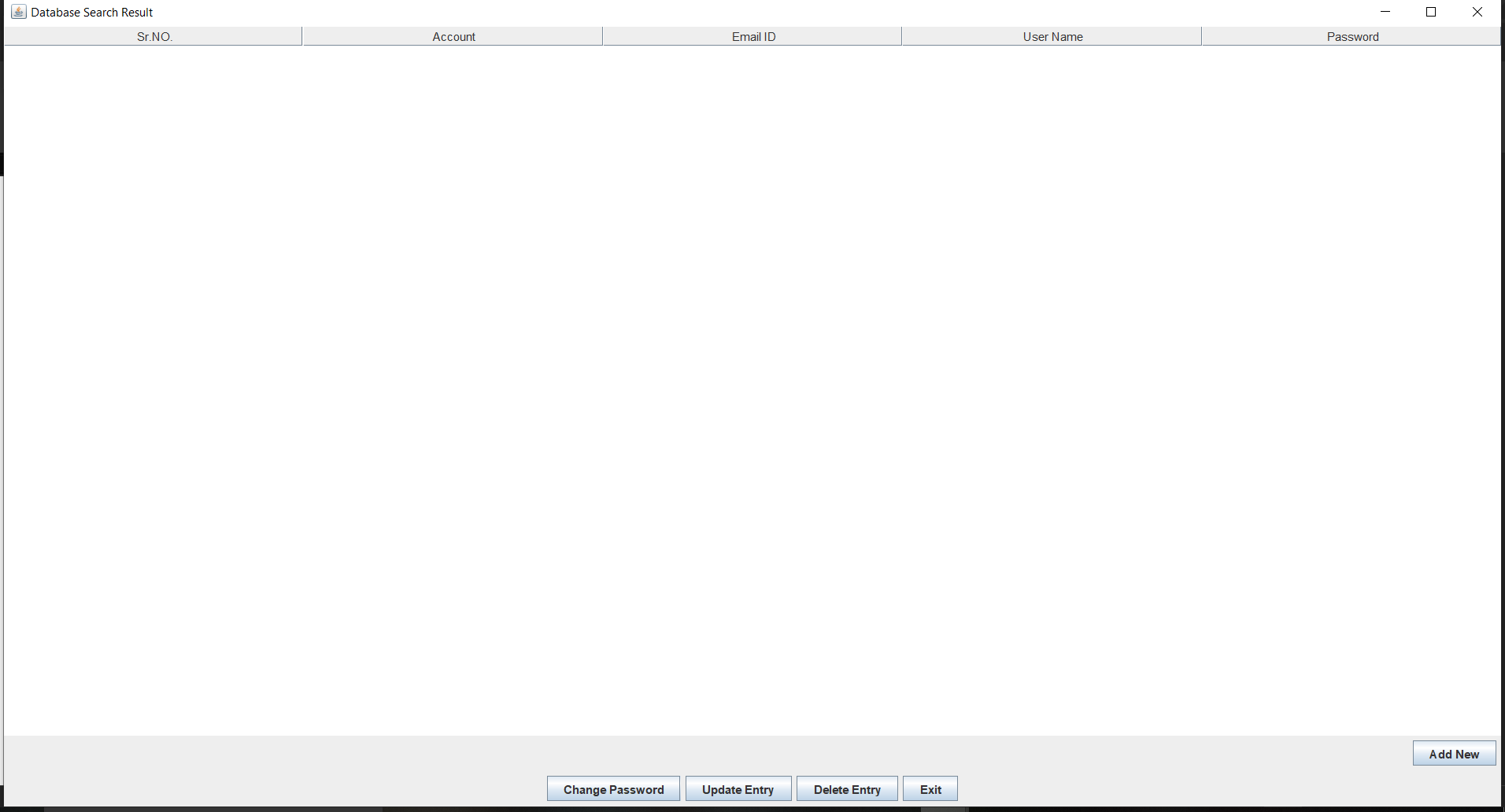
Now the backend Ms-Access Database adds entry in Registered Database Table of the new user. And one table of that username is created to store password





The new User will need to login again and then he will be directed to his Password Manager Page





**CONCLUSION**

Implementing this project helped us learn a lot about java applet and its uses. We learnt how to create a java applet, use various types of layouts, make use of textfields, labels, buttons, etc., how to create a event and making use of action listener for event handling to use the applet as a GUI, make use of UCanAccess driver to link applet to the MS-Access database, how to send an Email using Java and sending a Mobile Message using Java-API. This concept of Password Manager has a wide use to store and save all passwords to which access can be provided using with help of a master password. Thus this mini project by implementing java applet as front end and MS-Access(database) as back end was achieved and tested accordingly.

**REFERENCE**

**Text books:**

* Herbert Schildt, ‘JAVA: The Complete Reference’, Ninth Edition, Oracle Press.
* Sachin Malhotra and Saurabh Chaudhary, “Programming in Java”, Oxford University Press, 2010

**Reference Books:**

* Ivor Horton, ‘Beginning JAVA’, Wiley India.
* DietalandDietal, ‘Java: How to Program’, 8/e,PHI
* ‘JAVA Programming’, Black Book, Dreamtech Press.
* ‘Learn to Master Java Programming’, Stardusolutions

**Digital Material :**

* [www.geekforgeeks.com](http://www.geekforgeeks.com)
* [www.javatpoint.com](http://www.javatpoint.com)
* [www.textlocal.in](http://www.textlocal.in)
* [www.tutorialsfield.com](http://www.tutorialsfield.com)
* [www.sourceforge.net](http://www.sourceforge.net)