Project Report

ON

CONTACTS MANAGEMENT SYSTEM USING JAVA (NetBeansIDE) AND MySQL

By: - Hemant Gupta

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Project Name: Contacts Management System

Project Duration: 15 Days

Project Detail: It is a sample application designed individually in order to simulate major functioning of contacts management system application software in a java desktop application with help of core java programming using Net Beans IDE and back end database connectivity with MySQL. Under this topic we had gone through a practical development of JAVA Desktop Application with MySQL connectivity in backend using JAVA programming language and J2EE Architecture. This Project in JAVA Language of Contacts Management System is a simple application with simple graphics developed with NetBeansIDE and MySQL. In this project you can go through all basic operation which a traditional Contacts Management System go through in its execution cycle just like you feel it elsewhere like in mobile phones, cloud etc..

This Project will aim on understanding the functioning of Contact's Management System to add, delete and update candidate's information in databases.

CHAPTER 2: JAVA

Java is a general-purpose, concurrent, object-oriented, class-based, and the runtime environment (JRE) which consists of **JVM** which is the cornerstone of the Java platform.

Java has been used in different domains. Some of them are listed below:

- Banking: To deal with transaction management.
- Retail: Billing applications that you see in a store/restaurant are completely written in Java.
- Information Technology: Java is designed to solve implementation dependencies.
- Android: Applications are either written in Java or use Java API.
- Financial services: It is used in server-side applications.
- Stock market: To write algorithms as to which company they should invest in.
- Big Data: Hadoop MapReduce framework is written using Java.
- Scientific and Research Community: To deal with huge amount of data.

2.1 HISTORY

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. There are given significant points that describe the history of Java.

2.2 J2EE

This was originally known as Java 2 Platform, Enterprise Edition, which was later changed to Java Platform, Enterprise Edition (Java EE). This is one of the three computing platforms released by Sun Microsystems which was later acquired by Oracle Corporation. The other two platforms are Java Standard Edition (Java SE) and Java Micro Edition (Java ME). J2EE is nothing but an extension of the Java SE based on the Java programming language used for developing and deploying web-based enterprise applications. It consists of a set of APIs, services, and protocols that provide the functionality to develop multi-tiered web-based applications. It includes several technologies that extend the functionality of the Java SE APIs, such as Servlets, Connectors, Enterprise JavaBeans, etc.

It's mainly used for applications which run on servers and accessible through browsers like Chrome, Firefox, etc. It's also used for developing web applications over World Wide Web by creating standardized modular components to handle many aspects of programming. The J2EE architecture provides services to simplify the common challenges faced by developers while developing modern applications, thereby making it easier to implement industry-standard design patterns for greater efficiency and reliability.

The platform was known as *Java 2 Platform, Enterprise Edition* or *J2EE* from version 1.2, until the name was changed to *Java Platform, Enterprise Edition* or *Java EE* in version 1.5.

- J2EE 1.2 (December 12, 1999)
- J2EE 1.3 (September 24, 2001)
- J2EE 1.4 (November 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (December 10, 2009)
- Java EE 7 (May 28, 2013, but April 5, 2013 according to spec document. June 12, 2013 was the planned kickoff date)
- Java EE 8 (August 31, 2017)

2.3 JDBC

JDBC (Java Database Connectivity) is the Java API that manages connecting to a database, issuing queries and commands, and handling result sets obtained from the database. Released as part of JDK 1.1 in 1997, JDBC was one of the first components developed for the Java persistence layer.

JDBC was initially conceived as a client-side API, enabling a Java client to interact with a data source. That changed with JDCB 2.0, which included an optional package supporting server-side JDBC connections. Every new JDBC release since then has featured updates to both the client-side package (java.sql) and the server-side package (javax.sql). JDBC 4.3, the most current version as of this writing, was released as part of Java SE 9 in September 2017.

This article presents an overview of JDBC, followed by a hands-on introduction to using the JDBC API to connect a Java client with SQLite, a lightweight relational database.

How JDBC works

Developed as an alternative to the C-based ODBC (Open Database Connectivity) API, JDBC offers a programming-level interface that handles the mechanics of Java applications communicating with a database or RDBMS. The JDBC interface consists of two layers:

- The JDBC API supports communication between the Java application and the JDBC manager.
- The JDBC driver supports communication between the JDBC manager and the database driver.

JDBC is the common API that your application code interacts with. Beneath that is the JDBC-compliant driver for the database you are using.

CHAPTER 3: MySQL HISTORY

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 License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, or a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services, including MariaDB and Percona.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (proprietarily licensed) mSQL antecedent.

CHAPTER 4: OBJECTIVE

To learn JAVA application development and database connectivity with MySQL with the application of computer graphics, developing a desktop application can always be a better and smarter option. To have an idea of what we can achieve with relatively simple OOPs concepts and JAVA(j2ee) and perhaps the basis by which to extend the principles and create more interesting applications of our own.

CHAPTER 5: SYSTEM REQUIREMENT ANALYSIS

5.1 SOFTWARE REQUIREMENTS:-

Operating system: WINDOWS

Application software: NetBeansIDE & MySQL Command Line

Language: JAVA and Sql

5.2 HARDWARE REQUIREMENTS:-

Hard Disk: - 32 GB

RAM: - 128 MB

Processor: - Any Pentium version

CHAPTER 6: MySQL DATABASE

Database Name:- contacts

```
mysql> create database contacts;
Query OK, 1 row affected (0.00 sec)

mysql> use contacts;
Database changed
mysql>
```

Figure 6.1 DATABASE CREATION

CHAPTER 7: MySQL TABLES

Table Name:-

- Mycontact

```
mysql> create table mycontact(
    -> id int(11) primary key,
    -> name varchar(30),
    -> email varchar(70),
    -> city varchar(20),
    -> phone_no varchar(30),
    -> address varchar(300));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> desc mycontacts;
ERROR 1146 (42S02): Table 'contacts.mycontacts' doesn't exist
mysql> desc mycontact;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| name     | varchar(30)  | YES  |     | NULL    |       |
| email    | varchar(70)  | YES  |     | NULL    |       |
| city     | varchar(20)  | YES  |     | NULL    |       |
| phone_no | varchar(30)  | YES  |     | NULL    |       |
| address  | varchar(300) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
6 rows in set (0.02 sec)
```

Figure 7.1 TABLE CREATION

CHAPTER 8: NET BEANS IDE CODING

8.1 home.java

Figure 8.1.1 WELCOME PAGE

i) Let's Enter Button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.setVisible(false);
    new login().setVisible(true);

}
```

ii) Exit Button:-

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

    // TODO add your handling code here:

    System.exit(0);
}
```

8.2 login.java

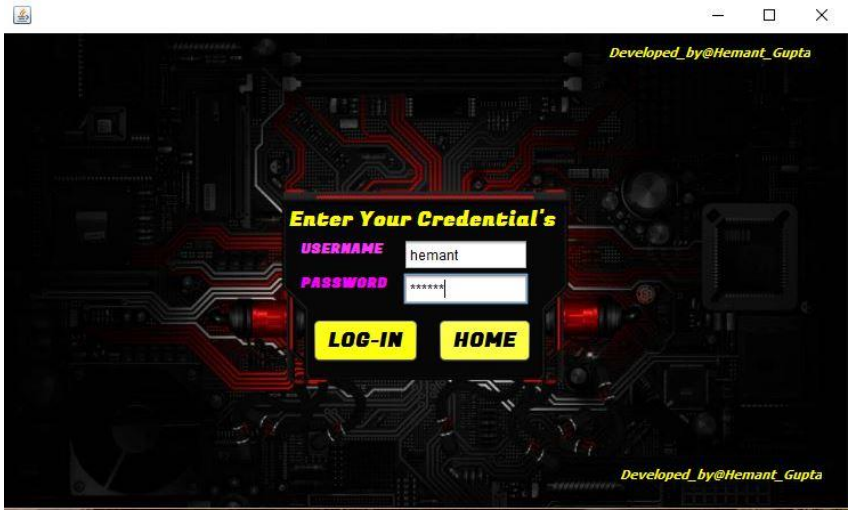

Figure 8.2.1 LOGIN PAGE

i) Log-In button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:

       String username= jTextField1.getText();
       String password= new String(jPasswordField1.getPassword());

       if(username.equals("hemant") && password.equals("hemant"))
       {
          this.setVisible(false);
          new options().setVisible(true);
       }
      else
          JOptionPane.showMessageDialog(null,"Password or username is Incorrect.\nPlease Enter
          Correct Credentials");
}
```

ii) Home Button:-

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

     // TODO add your handling code here:
     this.setVisible(false);
     new home().setVisible(true);

  }
```

8.3 options.java

Figure 8.3.1 Options Page

i) Add Contact Button:-

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

        // TODO add your handling code here:
        this.setVisible(false);
        new add_update_delete_search_contact().setVisible(true);
    }
```

ii)Update Contact Button:-

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

        // TODO add your handling code here:
        this.setVisible(false);
        new add_update_delete_search_contact().setVisible(true);
       }
```

iii)Delete Contact Button:-

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        this.setVisible(false);
        new add_update_delete_search_contact().setVisible(true);
       }
```

iv)Search Contact Button:-

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
        this.setVisible(false);
        new add_update_delete_search_contact().setVisible(true);

       }
```

v)Log-Out Button

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

        // TODO add your handling code here:
        this.setVisible(false);
        new login().setVisible(true);

  }
```

8.4 add_update_delete_search_contact.java

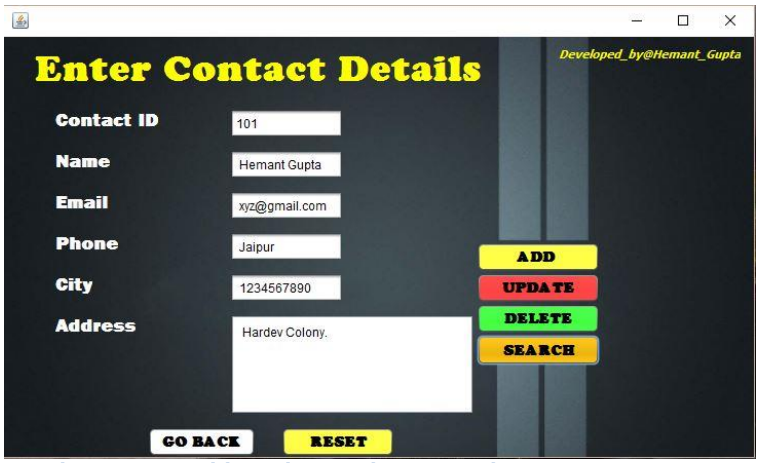

Figure 8.4.1 Add_Update_Delete_Search Contact Page

i) Add button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
       try
       {
        Class.forName("java.sql.Driver");
        con=DriverManager.getConnection("jdbc:mysql://localhost/contacts","root","root");
        stmt=con.createStatement();

        id=Integer.parseInt(jTextField1.getText());
        name=jTextField2.getText();
        email=jTextField3.getText();
        city=jTextField5.getText();
        phone_no=jTextField4.getText();
        address=jTextArea1.getText();

        query = "insert into mycontact values
        ("+id+",'"+name+"','"+email+"','"+city+"','"+phone_no+"','"+address+"');";
        System.out.println(query);
        stmt.executeUpdate(query);

        JOptionPane.showMessageDialog(null, "RECORD ADDED SUCCESSFULLY");
        }
        catch (Exception e1)
        {
         JOptionPane.showMessageDialog(null, "RECORD CANT BE ADDED");
        }
    }
```

ii)Update Button:-

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
   try
   {
   Class.forName("java.sql.Driver");
   con=DriverManager.getConnection("jdbc:mysql://localhost/contacts","root","root");
   stmt=con.createStatement();

   id=Integer.parseInt(jTextField1.getText());
   name=jTextField2.getText();
   email=jTextField3.getText();
   city=jTextField5.getText();
   phone_no=jTextField4.getText();
   address=jTextArea1.getText();

  query = "update mycontact set
  name='"+name+"',email='"+email+"',city='"+city+"',phone_no='"+phone_no+"',address='"+address+"'
  where id="+id+";";

   System.out.println(query);
   stmt.executeUpdate(query);

   JOptionPane.showMessageDialog(null, "RECORD Updated SUCCESSFULLY");
   }
   catch (Exception e1)
   {
    JOptionPane.showMessageDialog(null, "RECORD CANT BE Updated");
   }
}
```

iii) Delete button:-

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
   try
   {
   Class.forName("java.sql.Driver");
   con=DriverManager.getConnection("jdbc:mysql://localhost/contacts","root","root");
   stmt=con.createStatement();

   id=Integer.parseInt(jTextField1.getText());

   query = "delete from mycontact where id="+id+";";
   System.out.println(query);
```

```
    stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(null, "RECORD Deleted SUCCESSFULLY");
    }
    catch (Exception e1)
    {
     JOptionPane.showMessageDialog(null, "RECORD Doesn't Exist");
    }
}
```

iv) <u>Search Button:-</u>

```
    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
     // TODO add your handling code here:
    try
    {
     Class.forName("java.sql.Driver");
     con=DriverManager.getConnection("jdbc:mysql://localhost/contacts","root","root");
     stmt=con.createStatement();

     id=Integer.parseInt(jTextField1.getText());


     query = "select * from mycontact where id="+id+";";
     System.out.println(query);
     rs= stmt.executeQuery(query);
     if(rs.next())
     {
     name=rs.getString("name");
     email=rs.getString("email");
     city=rs.getString("city");
     phone_no=rs.getString("phone_no");
     address=rs.getString("address");

     jTextField2.setText(" "+name);
     jTextField3.setText(" "+email);
     jTextField4.setText(" "+city);
     jTextField5.setText(" "+phone_no);
     jTextArea1.setText(" "+address);
     }
     else
    {
    JOptionPane.showMessageDialog(null, "Record doesnt exist");
     }
     }
   catch(Exception e1)
```

```
    {
      JOptionPane.showMessageDialog(null, "Error in Connectivity");
    }
  }
```

v)GO-BACK Button:-

```
  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      this.setVisible(false);
      new options().setVisible(true);
  }
```

vi) Reset Button:-

```
  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      jTextField1.setText("");
      jTextField2.setText("");
      jTextField3.setText("");
      jTextField4.setText("");
      jTextField5.setText("");
      jTextArea1.setText("");
  }
```

CHAPTER 9: TESTING AND SCREENSHOTS OF PROJECT

1. Show tables

```
mysql> use contacts;
Database changed
mysql> show tables;
+--------------------+
| Tables_in_contacts |
+--------------------+
| mycontact          |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

Figure 9.1 Show Table

2. Desc Table

```
mysql> desc mycontact;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| name     | varchar(30)  | YES  |     | NULL    |       |
| email    | varchar(70)  | YES  |     | NULL    |       |
| city     | varchar(20)  | YES  |     | NULL    |       |
| phone_no | varchar(30)  | YES  |     | NULL    |       |
| address  | varchar(300) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

Figure 9.2 Desc. Table

3. Show table content

```
mysql> select * from mycontact;
+-----+---------------+----------------+---------+------------+-----------------+
| id  | name          | email          | city    | phone_no   | address         |
+-----+---------------+----------------+---------+------------+-----------------+
| 101 | Hemant Gupta  | xyz@gmail.com  | Jaipur  | 1234567890 | Hardev Colony.  |
| 102 | Rashika Sharma | xabc@gmail.com | Gwalior | 0123456789 | Subhash Stambh. |
+-----+---------------+----------------+---------+------------+-----------------+
2 rows in set (0.00 sec)
```

Figure 9.3 Show table content

4. Home Page

Figure 9.4 Home page snapshot

5. Login Page

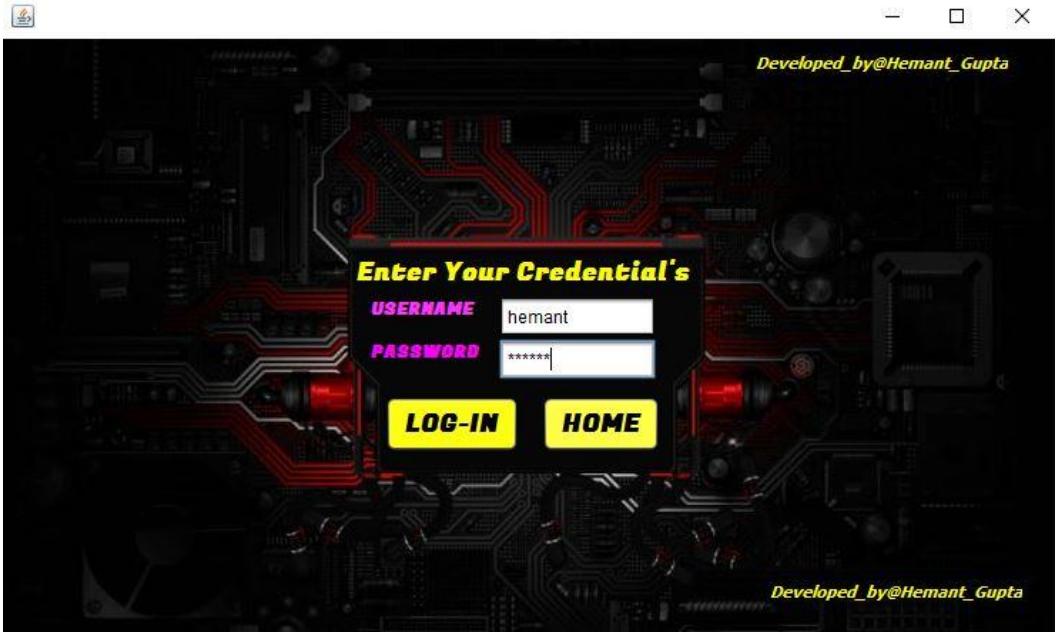

Figure 9.5 Login Page snapshot

6. Options Page

Figure 9.6 Options Page snapshot

7. Add_Update_Delete_Search_Contact Page

Figure 9.7 Add_Update_Delete_Search Page snapshot

CHAPTER 10: FUTURE OF THE PROJECT

This Project will aim on understanding the functioning of software used by Contacts Management System Application / software to add, delete and update customer and products information in databases and how it can be more improved using advanced technologies like artificial intelligence and this project will help new programmers and students to understand the use of J2EE after learning basic java and to deal with graphics, databases and connectivity and how a desktop application can be useful and fruitful in multiple dynamic aspects.

CHAPTER 11: CONCLUSION

This is to conclude that this project simulates the basic fundamental functioning of how Contacts Management System Application Works.

CHAPTER 12: BIBLIOGRAPHY

Reference Books:

- JAVA The Complete Reference 8th Edition
- Head First JAVA

Reference Websites

- https://www.tutorialspoint.com/awt/awt_graphics_class.htm
- https://www.webopedia.com/TERM/J/J2EE.html
- https://www.oracle.com/java/technologies/appmodel.html
- http://zetcode.com/db/mysqljava/
- https://en.wikipedia.org/wiki/MySQL
- https://dev.mysql.com/doc/connectors/en/connector-j-reference-type-conversions.html
- https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-connect-drivermanager.html
- https://www.mysqltutorial.org/mysql-jdbc-tutorial/