

(ISO 9001:2015 Certified), Accredited with ‘A’ Grade by NAAC

: 08258 - 281039 – 281263, Fax: 08258 – 281265

### Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

Report on Mini Project

Student Database Management System

#### Course Code : 18CS506 Course Name : Database Management Systems Lab

##### Semester: V SEM Section: A

**Submitted To:**

## Dr. Sudeepa K.B

## Associate Professor

## Department of Computer Science and Engineering

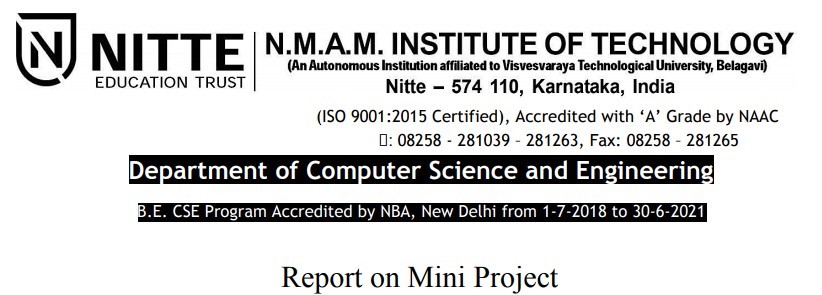
**Submitted By:**

Aditya Rao Chokkadi(4NM18CS007)

Arvind Kamath(4NM18CS024)

**Date of submission:** 31-12-2020

#### Signature of Course Instructor



**CERTIFICATE**

Certified that the mini project entitled

‘Student Database Management Systems’

Is a bona fide work carried on by

**Aditya Rao Chokkadi (4NM18CS007)**

**Arvind Kamath (4NM18CS024)**

In partial fulfillment of requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belgaum during the year 2020-

2021

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The mini project has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

#### Name and Sign of Guide Name & Sign of HOD Dr. Sudeepa K.B Dr. K R Udaya Kumar Reddy

#### Associate Professor, CSE HOD, CSE

**ACKNOWLEDGEMENT**

We believe that our project will be complete only after we thank the people who have contributed to make this project successful.

We thank Dr. N Niranjan Chiplunkar, Principal of NMAMIT, Dr. Sudeepa K.B, Associate Professor, Department of CSE, Dr. K R Udaya Kumar Reddy for the support given through us throughout the entire duration of program.

Finally, we thank the staff of CSE and all our friends for all the honest opinions and feedback.

**Aditya Rao Chokkadi**

**Arvind Kamath**

# 

# ABSTRACT

To design a Java Swing application(GUI) using WindowBuilder in Eclipse editor that is connected to a database (via appropriate drivers) which in turn has a table consisting of rows of student data. This database will be present in Microsoft SQL Server.

The access to the ‘Student Database Management Systems’ application will be allowed only if correct username and password is entered in login frame.

The contents of the student data table will be displayed by the GUI and it will support the following operations:-

1. Add New Button (to add new rows to the table)
2. Print Button (in order to print the table or save the table as pdf)
3. Update Button (in order to make modifications to a row)
4. Reset Button (in order set textboxes and combo boxes to default state)
5. Delete Button (in order to delete a row from the table)
6. Exit Button (in order to Exit the application)

**Table of Contents**

**Contents Page No.**

Title Page 1

Certificate 2

Acknowledgement 3

Abstract 4

Table of Contents 5

Introduction 6

Design 7-9

Implementation 9-13

Result 14-18

Conclusion 19

References 20

# INTRODUCTION

In educational institutes a record has to be made about the students studying in the institution. This record contains StudentID, name, address, gender, phone no etc. This data record can be kept as a separate file(such as excel file). But a problem arises as duplicates of these files may be created which would lead to inaccurate information about the students.

So, instead of creating multiple files, a single database is created in which the student table containing student details is created. The access to this database is generally given to the administration staff of educational institutes so that they can add information, delete info, print info, update info etc. Only a single copy of the table is available in the database. So any changes will be made to that table itself. So databases overcome the problems of using files to store data.

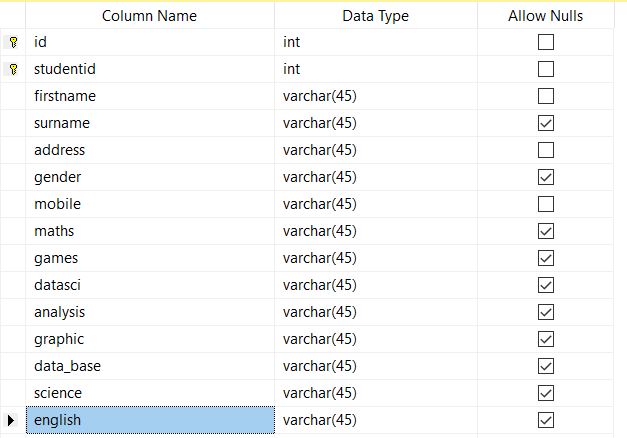
The administrative staff do not have technical knowledge in order to deal with the database using **SQL** and **DBMS** software. So a **GUI** is created by a person who has the technical knowledge about **DBMS** and **SQL**( using a programming language) and this **GUI** can be used by the administrative staff easily in order to deal with student data easily without having to worry about dealing with **DBMS**.

A **connection** is established between **java GUI** and **database** by making use of **JDBC API** and the appropriate **vendor specific driver.**

# DESIGN

# Database design:-

# First a table called ‘studentdata’ is created inside a database called ‘master’ in MS SQL Server Management Studio. It consists of the following columns along with their data types and whether they allow nulls or not:-



# id and studentid attributes together act as a composite primary key. Primary key is represented by the key icon.

# id attribute is set to auto increment mode( +1 each time new row is added, starts from value 1). We cannot manually give value to id attribute.

# firstname, surname, address, gender and mobile attributes together consist of personal information about a student.

# maths, games, datasci, analysis, graphic, data\_base, science and english attributes together consist of the subjects offered by the institute.

# Each of these attributes can have one of the following values from this set: {No,Yes,Core Unit,Complete}.

# No signifies that the student is not enrolled in that particular subject.

# Yes signifies that the student is enrolled in that particular subject.

# Core Unit signifies that the particular subject is compulsory for that student.

# Complete signifies that the student has cleared that particular subject.

# id and studentid attributes are of integer data type whereas the rest of the attributes are of varchar( String) data type.

# Java Swing GUI design (StudentDBMS class):-

# We make use of WindowBuilder in Eclipse editor in order to create GUI. First, an empty is JFrame is created by default by the editor. We use absolute layout in order to put the components inside JFrame at any desired location.

# 6 JPanels are added. These JPanels act as containers for other components to be added. 1Panel will contain JLabel, 3 panels will contain these components: JTextBox, JComboBox , 1 Panel will contain JButtons and the remaining panel will contain JTable.

# Within one of the JPanels, 5 textboxes are added in order to enter the studentid, firstname, surname, address and mobile no. 1 JComboBox is added in order to select gender(Male or Female).

# 2 JPanels contain JComboBoxes for each of the subject columns present in the ‘studentdata’ table. The options offerred are: No, Yes, Core Unit, Complete.

# 1 JPanel contains 6 JButtons: Add New, Print, Update, Reset ,Delete and Exit whose functions have already been specified in the ABSTRACT section of this project report.

# 1 JPanel contains a JTable in order to display all the rows and columns present in the ‘studentdata’ table.

* The remaing JPanel contains JLabel that contains the text “Student Database Management System”.
* Additonally another JFrame called ‘**Student DBMS Login’** is created with 1 JTextBox(for username), 1 JPasswordField(for password),1 JCheckBox( to show characters of password or to show ‘\*’ instead of characters) and 1 JButton whose text is ‘Login’ (in order to submit the details and login).

This JFrame acts as a login screen and the other JFrame( containing JTable and rest of the components) is displayed only if the correct username and password is entered.

**IMPLEMENTATION**

The source code for GUI layout and components is automatically generated by WindowBuilder. Now we have to implement the functionality of the buttons and certain other things:-

1. First a method called ‘**updateDB()**’ is defined. This method is used( called) in order to display all the rows and columns of the ‘**studentdata**’ table in the JTable of the GUI.

This is done by first obtaining a ‘**DefaultTableModel** ‘ object from the JTable object by using the method ‘**getModel()**’ . Next the query “**SELECT \*** **FROM studentdata**” is executed via **executeQuery()** method of **PreparedStatement** object and a **ResultSet** object is obtained( which contains the results of query). By making use of this object, all the rows are traversed and in each iteration of loop , column values are added to a Vector ‘v’. Lastly by making use of ‘**addRow()**’ method of ‘**DefaultTableModel**’ object, the Vector ‘v’ (which contains all column values of a row) is added to the JTable in each iteration of while loop thereby making it visible to the user in the GUI via JTable.

1. ‘**mouseClicked**’ event is added to the JTable so that when a row on it is selected then the respective JTextBoxes and JComboBoxes corresponding to each column of that row are filled automatically.(in order to make it easier to perform update operation).

This is achieved by using ‘**DefaultTableModel**’ object and its method ‘**getValueAt(int row,int column)**’ in order to fetch the values from the specified row and column and then displaying them on the JTextBoxes and JComboBoxes by the methods ‘**setText(String text)**’ and ‘**setSelectedItem(Object obj)**’ respectively.

Now we just have to implement the buttons:-

1. ‘**ActionEvent**’ is added to the JButton ‘**Add New’** so that when the button is pressed all the data entered in the JTextBoxes and JComboBoxes are added as new row to the database. (Code is written inside ‘**actionPerformed()**’ method)

This is done by making use of ‘**PreparedStatment**’ object and executing the

SQL INSERT statement ‘**INSERT INTO studentdata(studentid,firstname,**

**lastname,---------) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)**’ where

----------- signifies the rest of the column names.‘**PreparedStatment’** method

‘**setString(int pos,string value)**’ can be used to set the values of those

Question marks(**?**) after having obtained the values from the respective

JTextBoxes and JComboBoxes. Finally the update can be executed by

making use of the ‘**PreparedStatement**’ method **‘executeUpdate()**’ after

which a new row is added to the table. At the end ‘**updateDB()**’ method is

called inorder to display the updated contents of ‘**studentdata**’ table in JTable.

# ‘ActionEvent’ is added to JButton ‘Print’ in order to display a printing window in which we can choose to either print the contents of JTable or save its contents as a pdf file.

# 

# This is done by making use of JTable class method

# ‘print(int PRINT\_MODE, MessageFormat header, MessageFormat footer)’

# Header specified the text that will appear on each page above the table.

# Footer is generally used to make page numbers appear at the bottom of each

# page.

# PRINT\_MODE specifies the printing mode. There are two printing modes,

# NORMAL AND FIT\_WIDTH print modes.

# 

# In NORMAL mode, if many columns are there then they will be printed on different pages whereas in FIT\_WIDTH mode, column sizes will be reduced so that all of them fit on a single page.

# ‘ActionEvent’ is added to JButton ‘Update’ in order to get the updated values from the JTextBoxes and JComboBoxes and change the column values of that row to those values( Code is written inside ‘actionPerformed()’ method)

This is done by making use of ‘**PreparedStatment**’ object and executing the SQL UPDATE statement ‘**UPDATE studentdata SET studentid=?,firstname=?,surname=?,address=?,-------- WHERE id=?’** where ------- signifies:similarly for the rest of the columns.

As discussed earlier the ‘**setString(int pos,string value)**’ method of

‘**PreparedStatement**’ is used inorder to set the values of the respective question marks(?). The String values(updated) are obtained from the JTextBoxes and JCheckBoxes . After this the’ **executeUpdate()**’ is invoked

by ‘**PreparedStatement’** object which causes the column values of the selected row to be updated. At the end ‘**updateDB()**’ method is called inorder to display the updated contents of ‘**studentdata’** table in JTable.

6) ‘**ActionEvent**’ is added to JButton ‘**Reset**’ in order to clear or set the

JTextBoxes and JComboBoxes to their default state(empty for text box and

‘No’ for combo box) (Code is written inside ‘**actionPerformed()**’ method).

This is done by making use of ‘**setText(String text)**’ method for JTextBoxes and ‘**setSelectedIndex(int index)**’ for JComboBoxes.

For JTextBox: **setText(“”)** (in order to clear text box)

For JComboBox: **setSelectedIndex(0)** (in order to set JComboBoxes to display ‘No’)

Even the JTable is cleared by invoking the method ‘setRowCount(int count)’ of

‘**DefaultTableModel**’ class. The argument passed is 0.

7) ‘**ActionEvent**’ is added to JButton ‘**Delete**’ in order to delete a selected row from the database (Code is written inside ‘**actionPerformed()**’ method).

This is done by making use of ‘**PreparedStatement’** object and the SQL DELETE statement ‘**DELETE FROM studentdata WHERE id=?’**

The selected row number is obtained by using ‘**getSelectedRow()**’ method of JTable class.

This value of ‘**id’** column of the selected row is obtained by using

**‘getValueAt(int row,int column)**’ of ‘**DefaultTableModel**’ class. Arguments passed are selectedRow and 0.

The value for question mark(?) is set by using ‘**setInt(int pos,int value)**’ method of ‘**PreparedStatement**’ and finally the ‘**executeUpdate()’** method is called, after which, the row selected in the JTable gets deleted. The JTextBoxes and JComboboxes are also set to default state. After this **‘updateDB()**’ method is invoked inorder to display the updated **‘studentdata’** table on JTable.

# 

# 8) ‘ActionEvent’ is added to JButton ‘Exit’ in order to exit the application( code is written inside actionPerformed() method).

# A confirmation dialog box is created and called by using the static method ‘showConfirmDialog(Component comp, String message, String title, int optionMode)’ of ‘JOptionPane’ class.

# The dialog box will display two options yes and no. If yes is pressed then the method returns a value of 0 due to which the condition inside if clause becomes true due to which ‘System.exit(1)’ is invoked which will cause the application to terminate.

# If no is pressed then the method returns a value 1 but nothing would happen and we can continue to operate the application normally.

# All the button have thus been implemented. The only thing remaining is to implement the JCheckBox and Login button of the ‘Student DBMS Login’ JFrame. In order to create Login JFrame we create a separate class called LoginScreen.

* ‘**ActionEvent’** is added to JCheckBox whose text is ‘**Show password’** so that when the checkbox is selected, the password characters are visible and when the checkbox is not selected then instead of the actual characters of password, asterisk(\*) is visible. (code is written inside ‘actionPerformed()’ method).

This is done by using the method ‘**isSelected()’** of JCheckBox class and method ‘**setEchoChar(char c)**’ of ‘**JPasswordField**’ class.

* ‘**ActionEvent**’ is added to JButton ‘Login’ in order to invoke the constructor of the **StudentDBMS** class (which defines the GUI and its components in order to work with the database). After the constructor is invoked, the JFrame defined by **StudentDBMS** class pops up on the screen.

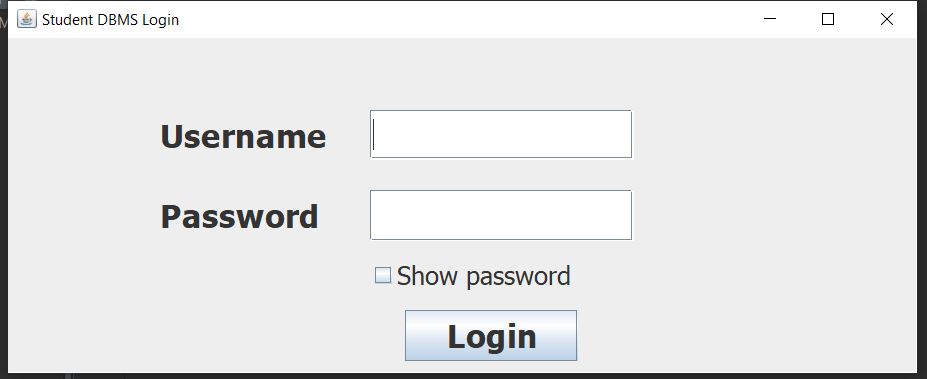
**StudentDBMS** constructor will be invoked only if the username and password entered In the JTextBox and JPasswordField match up with the username and password stored in the class variables ‘**username**’ and ‘**password**’ respectively.

Both of them are String variables. If they match up, a message dialog is displayed with message ‘**Login successful’** else if they don’t match up then a message dialog is displayed with message ‘**Invalid username or password!**’.

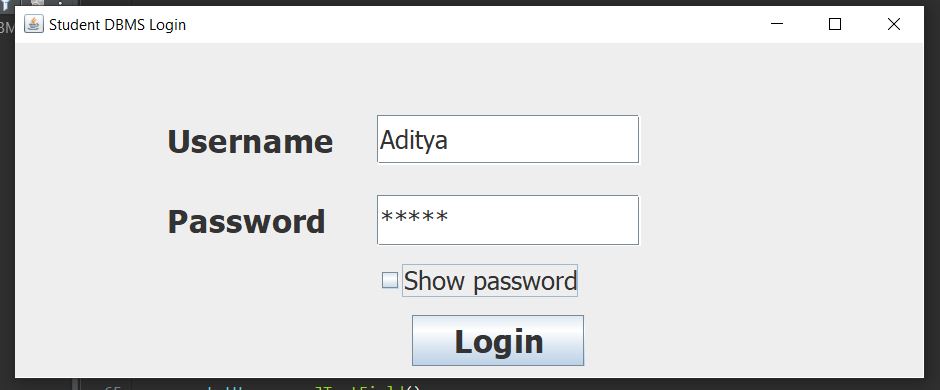
If the login is successful then only the JFrame of **StudentDBMS** class is displayed.

**RESULT**

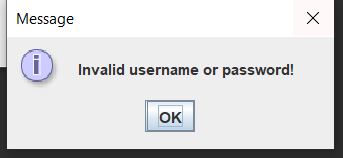
The login JFrame appears as shown:-



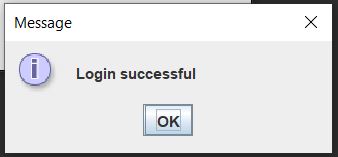
The username and password is typed in:-



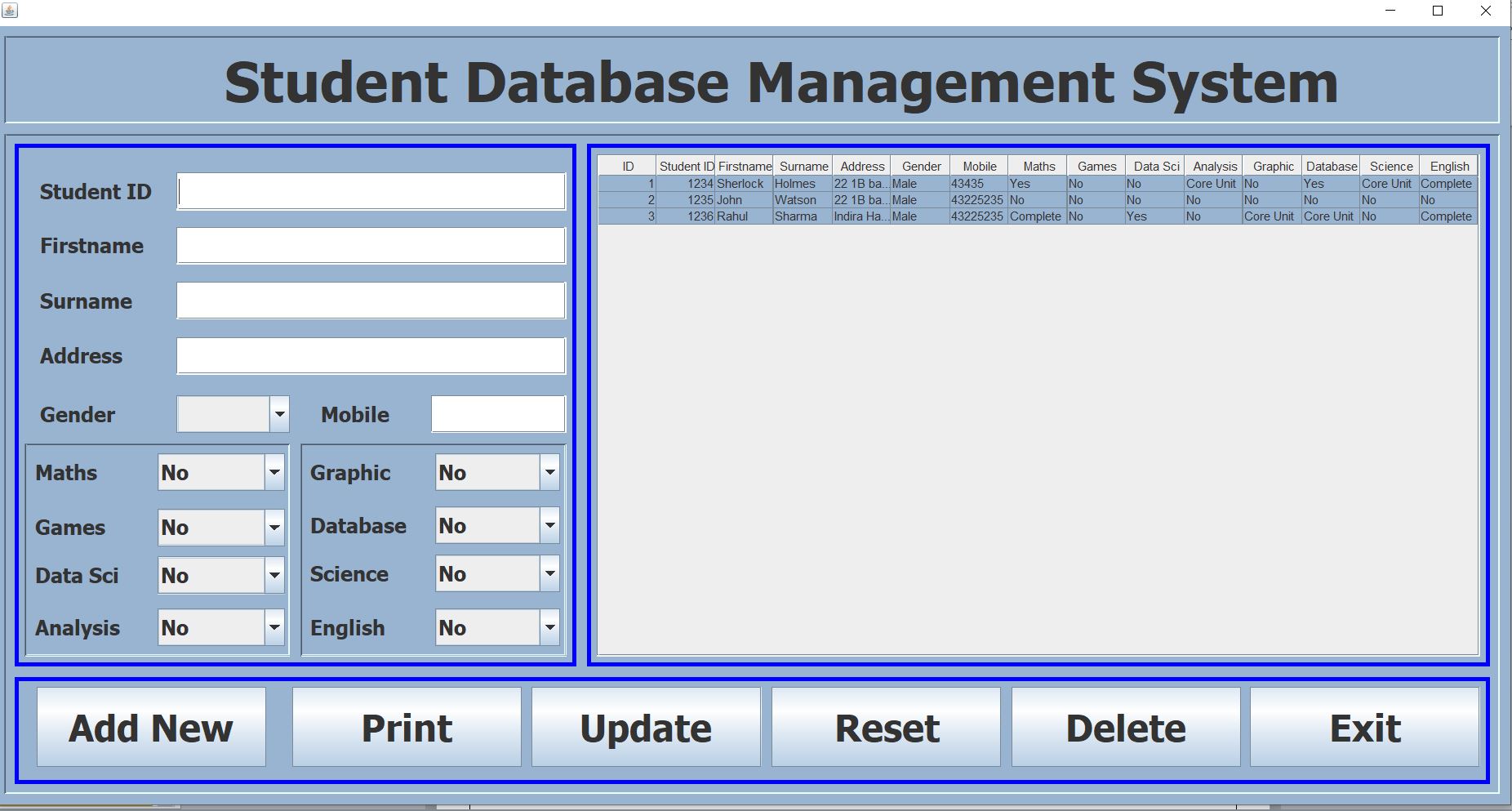
If wrong username or password is entered:-



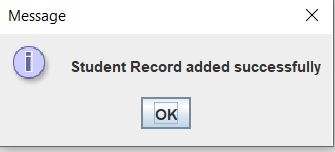
if correct username and password is entered:-



After pressing **OK** button, the **StudentDBMS** class’s JFrame appears:-

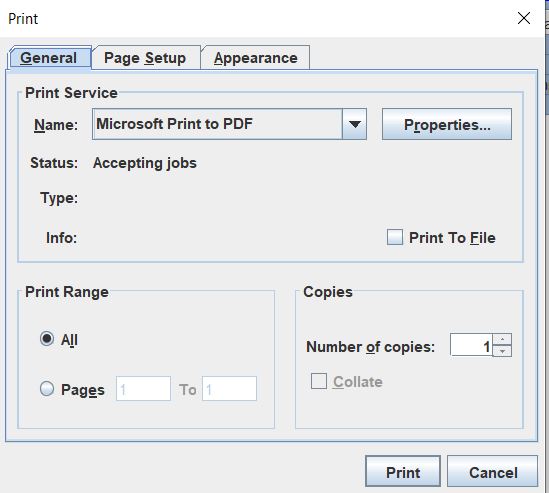


After having input the student details, the Add New button is pressed to add new row to the ‘**studentdata’** table. The following appears:-



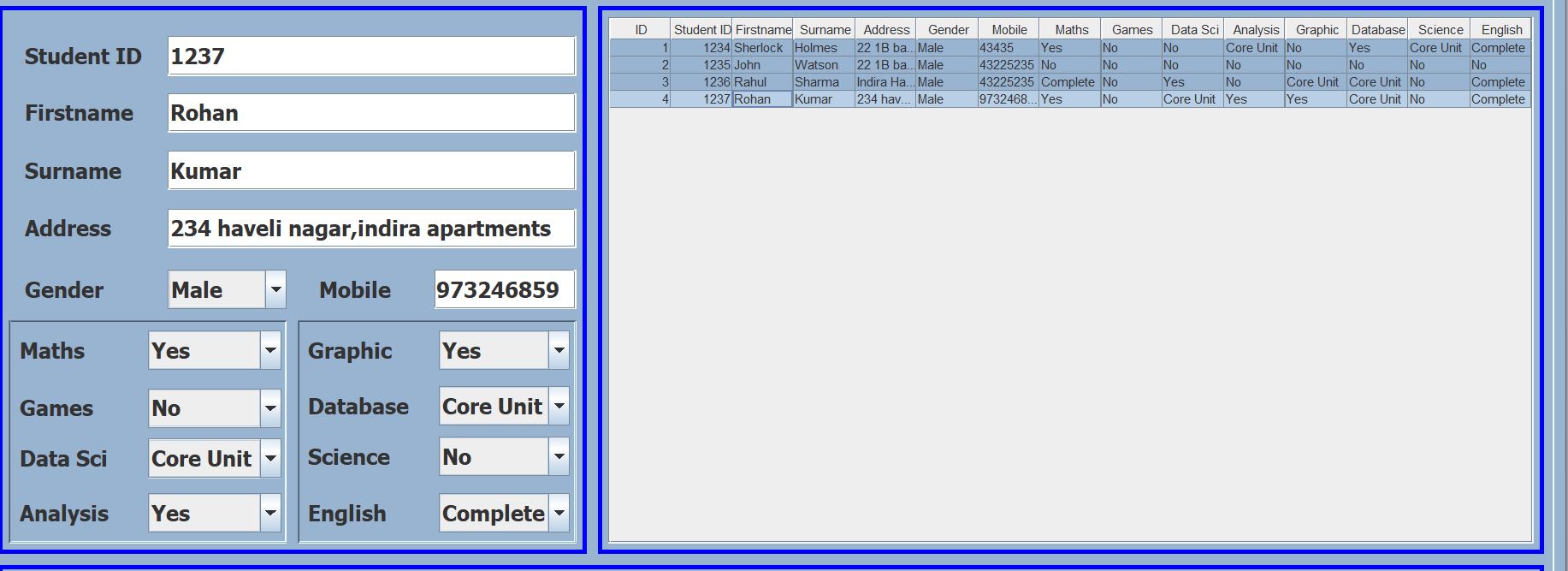
After pressing **OK** button, new row shows up in the JTable(it has been added to actual **‘studentdata’** table as well).

# On pressing Print Button, the following appears:-



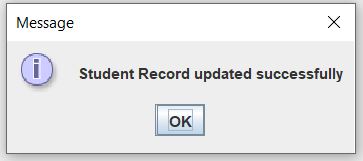
From here we can either print the contents of JTable or save it as pdf file.

When one of the rows from JTable is clicked upon then the corresponding details appear in the JTextBoxes and JComboBoxes as follows:-

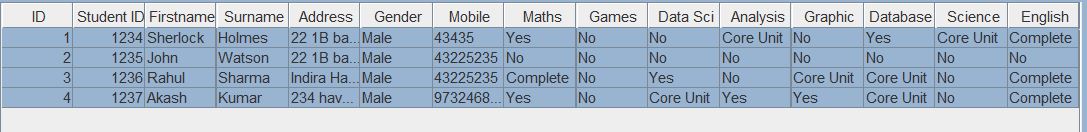


Here the 4th row has been selected.

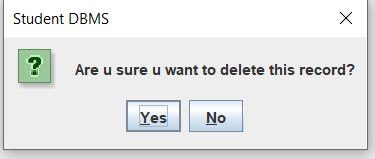
After making changes if needed, the **Update** button is pressed. Then the following appears:-



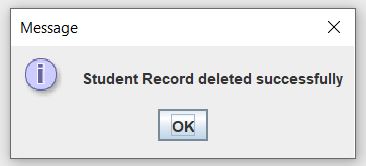
In the 4th row, the firstname was changed from ‘Rohan’ to ‘Akash’. The JTable is as follows:-



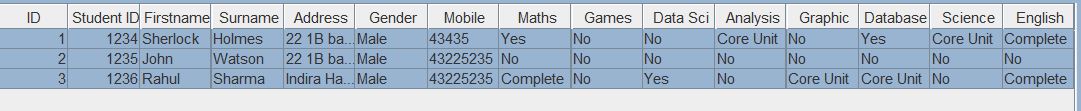
If the same 4th row is selected( clicked upon) and the **Delete** button is pressed then the following appears:-



If **Yes** button is pressed:-



After pressing **OK** button, the 4th row disappears from JTable.( it has been deleted from **‘studentdata’** table as well). JTable is as follows:-

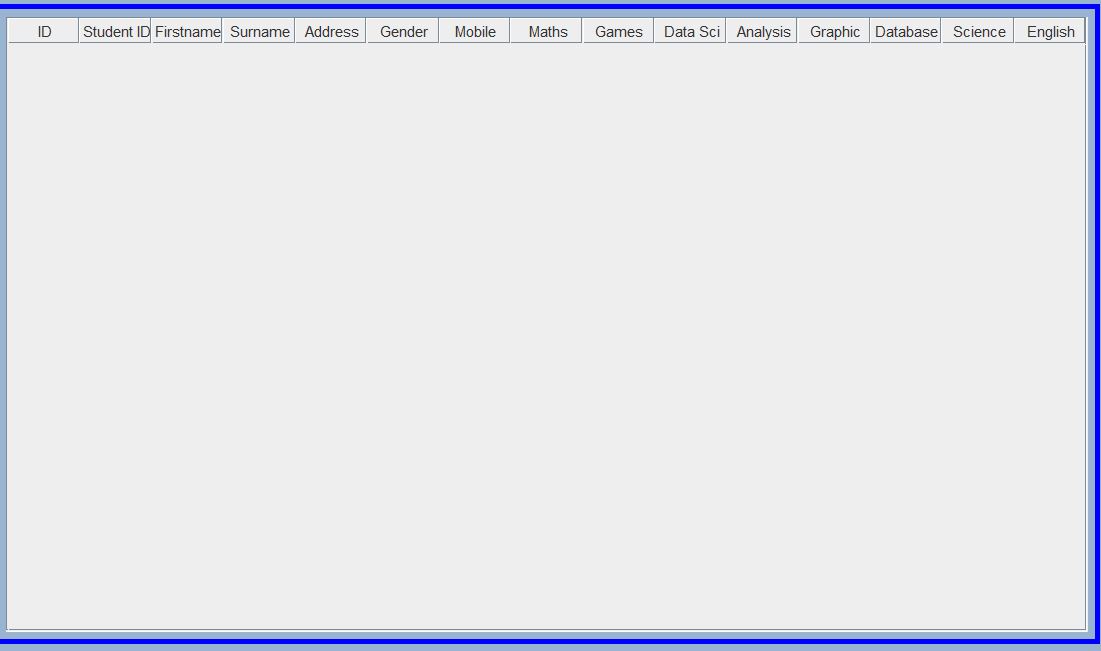


# When the Exit button is pressed, the following appears:-



If **Yes** button is pressed then the application closes. Whereas if **No** button is pressed, the dialog box disappears and we can continue to use the application as normal.

If the **Reset** button is pressed then along with the JTextBoxes and JComboBoxes getting set to their default state, the JTable is made empty and it appears as follows:-



Only the JTable is cleared not the actual ‘**studentdata**’ table present in ‘**master**’ database in **MS SQL Server**. If we restart the application or if we add a new row then the rows and columns of ‘**studentdata**’ table are again displayed on JTable as ‘**updateDB()**’ method is called.

**CONCLUSION**

**Java swing application(GUI)** using **WindowBuilder** tool has been developed in order to work with ‘**studentdata**’ table present in ‘**master**’ database in **MS SQL Server** and the operations supported are exit, reset components, add a new row, delete an existing row, update an existing row and to either print or save contents of JTable as pdf document.

The ‘**studentdata**’ table contains information about the students. The responsibility to maintain this information is usually assigned to the Administrative staff of educational institutions.They can make use of such kind of Graphical User Interfaces in order to deal with the data easily.

**REFERENCES**

* The Complete Reference-Java-Seventh Edition-‘Herbert Schildt’
* Fundamentals of Database Systems-Seventh Edition-‘Ramez Elmasri’ and ‘Shamkant B. Navathe’
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/JTable.html>
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/JPasswordField.html>
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/table/DefaultTableModel.html>
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html#:~:text=JOptionPane%20makes%20it%20easy%20to,section%20in%20The%20Java%20Tutorial>.
* <https://examples.javacodegeeks.com/desktop-java/ide/eclipse/eclipse-windowbuilder-tutorial/>