



# Government Women's Polytechnic College, Thrissur

[Approved by All India Council for Technical Education]

**NBA Accredited Diploma in Computer Engineering (2019-2022)**

Nedupuzha, Thrissur, Kerala, INDIA – 680 007 web: [gwpctsr.ac.in](http://gwpctsr.ac.in)

Ph:487-2449182, Email: [wpcnedupuzha@gmail.com](mailto:wpcnedupuzha@gmail.com)



## Application Development Lab Using Java

Report

On

## ADMISSION MANAGEMENT FOR COLLEGE

by

**T. S. PARVATHY**

REG.NO: 19138360

Semester 4, Computer Engineering

2020 – 21

# Government Women's Polytechnic College, Thrissur

[Approved by All India Council for Technical Education]

**NBA Accredited Diploma in Computer Engineering (2019-2022)**

Nedupuzha, Thrissur, Kerala, INDIA – 680 007 web: [gwpctsr.ac.in](http://gwpctsr.ac.in)

Ph:487-2449182, Email: [wpcnedupuzha@gmail.com](mailto:wpcnedupuzha@gmail.com)



## **CERTIFICATE**

This is to certify that project report titled **ADMISSION MANAGEMENT FOR COLLEGE** has been successfully completed by **T. S. PARVATHY (Register No: 19138360)** of Semester IV, in partial fulfillment of the requirement for the award of Diploma in Computer Engineering under the Directorate of Technical Education, Kerala State during the academic year 2020- 21.

Mrs. Gowrimol D  
Faculty Advisor, S4 Computer Engineering

Ms. Divya M B  
HOD Computer Engineering

Internal Examiner

External Examiner

Nedupuzha, Thrissur

Date: 29-3-2019

## **ACKNOWLEDGEMENT**

I would like to take this opportunity to express my gratitude towards all the people who have in various ways, helped in the successful completion of my experiment in Application Development Lab. This work is the outcome of the guidance provided by the experienced and dedicated persons of Govt. Women's Polytechnic College, Nedupuzha.

First of all, I would like to convey my hearty gratitude to my faculty advisor **Ms. Gowrimol D**, for giving me the constant source of inspiration and helping in the completion of my experiment, personally correcting my work and providing encouragement throughout the work. I am deeply obliged to **Ms. Divya M B**, Head of Department who guided me a lot for the successful completion of the work.

I would like to thank the valuable guidance provided by the technical staff and all other faculties of Computer Engineering Department for timely suggestion & advice which led to the successful completion of the work.

I thank my parents for their encouragement and support. I especially thank all my classmates who were always there to support me. Above all I thank God almighty for providing with us an opportunity to present my work in the way I did.

Thanks to all once again.....

(signature)  
T. S. PARVATHY

## **ABSTRACT**

The project entitled ADMISSION MANAGEMENT FOR COLLEGE aims at developing an online admission application for a college. This system is an online system that can be accessed throughout the organization and outside as well with proper login provided. Our system has two types of accessing modes, admin login and student. Student management system is managed by an administrator. It is the job of the administrator to admit and monitor the whole process. When a user logs in to the system, he/she can view and edit his/her personal information. The system has three modules. They are

- Admin Register
- Admin Login
- Student

Students have to fill in their details in the application form provided and they can also make changes or delete their application form even after submission. College principal/administrator logging in may also access/search information put up by the students.

## INDEX

CHAPTER No.	CONTENTS	PAGE No.
1	INTRODUCTION	1
2	REQUIREMENT ANALYSIS	7
3	SOFTWARE DESIGN	12
4	CODING AND IMPEMETATION	14
5	TABLES	30
6	TESTING	32
7	RESULT	34
8	CONCLUSION	35
9	APPENDIX	36

## **CHAPTER 1**

### **INTRODUCTION**

“**ADMISSION MANAGEMENT FOR COLLEGE**” is an efficient application for managing admission of eligible students to colleges. It helps in organizing the student information in a digitalized manner complete with the storage of student details in databases than the traditional register-and-paper based system. On a basic level, this application is mainly managed by two kinds of users namely Administrator and Student. Administrator has the main control of the application, whether it is managing students’ details or managing Admin details; while the students can only fill or edit their details through the application form provided and view them.

The project entitled **ADMISSION MANAGEMENT FOR COLLEGE** enables access to two types of Users:

1. Administrator: The administrator must first register with a predefined security key and create their profile. They can access the entire application with the help of this profile through the “Admin login” button. The main functionalities that the Administrator can control are:
  - Creating cut-off list
  - Viewing the final list
2. Student: The students can only upload their details through a registration form. These details can also be edited or deleted by them. The main functionalities include:
  - Filling Student application form
  - Updation and deletion of student details
  - Viewing the final qualified and disqualified list

All the data entered are stored in databases.

Any user with basic knowledge in computer usage will be able to operate this application with ease.

## **FRONT-End**

The front end used for the project is NetBeans 8.2. NetBeans IDE is a modular, standards-based integrated development environment (IDE), written in the Java programming language. The NetBeans project consists of a full-featured open source IDE written in the Java programming language and a rich client application platform, which can be used as a generic framework to build any kind of application.

NetBeans IDE 8.2 supports the following technologies and has been tested with the following application servers and mobile platforms.

### **Netbeans 8.2 Supported technologies:**

Java EE 7, Java EE 6, and Java EE 5	PHP 7, 5.6, 5.5, 5.4, 5.3, 5.2, 5.1
JavaFX 2.2.x and 8	Groovy 2.1
Java ME SDK 8.0	Grails 2.3, 2.2
Java Card 3 SDK	Apache Ant 1.9.7
Struts 1.3.10	Apache Maven 3.0.5 and earlier
Spring 4.0.1, 3.2.7, 2.5	C/C++/Fortran
Hibernate 4.2.6, 3.6.10	VCS
Issue Tracking	Subversion: 1.8, 1.7, 1.6
Bugzilla 4.4 and earlier	Mercurial: 2.8.x and later
Jira 3.13+ and 5.0+	Git 1.8.x an
Node.js 4.0+	

### **New Features in Version 8.2**

ECMAScript 6 and Experimental ECMAScript 7 Support  
HTML5/JavaScript Enhancements  
PHP 7 support  
Docker support  
Java Editor and Profiler Enhancements  
Debugger Enhancements  
C/C++ Enhancements

### **Hardware and Software required for running NetBeans 8.2**

NetBeans IDE runs on operating systems that support the Java VM (Virtual Machine) and has been tested on the platforms listed below. The IDE's minimum screen resolution is 1024x768 pixels.

### **Supported Operating Systems**

Ubuntu 9.10 or above

or

Microsoft Windows Vista SP1/Windows 7 Professional:

### **Minimum Hardware Configurations**

- **Processor:** 800MHz Intel Pentium III or equivalent
- **Memory:** 512 MB
- **Disk space:** 750 MB of free disk space

### **Recommended Hardware Configurations**

- **Processor:** Intel Core i5 or equivalent
- **Memory:** 2 GB (32-bit), 4 GB (64-bit)
- **Disk space:** 1.5 GB of free disk space

NetBeans IDE runs on the Java SE Development Kit (JDK) which consists of the Java Runtime Environment and developer tools for compiling, debugging, and running applications written in the Java language. The tested JDK for this release is JDK 8u101 for Windows, Linux, and OS X. The 8.2 version of the IDE cannot be installed or run on the JDK older than JDK 8. The PHP and C/C++ NetBeans bundles only require the Java Runtime Environment (JRE) 8 to be installed and run. Java features in the IDE and JavaFX 8 features require JDK 8.

You can download the JDK for your platform from the [Java SE Downloads](#) page. Also Java SE 8 is required to install and run NetBeans IDE 8.2.

### **SQL Editor and Databases**

The following changes have been made to the database functionality in NetBeans IDE 8.2:

- **Upgraded Drivers.** The MySQL drivers included in NetBeans IDE have been updated to version 6.0.
- **Tested Drivers**

NetBeans IDE 8.2 has been tested with the following databases and drivers.

Driver	Version	Example URL
JavaDB	Derby 10.3.1.4	jdbc:derby://localhost:1527/sample (Network)



Oracle	Oracle Database 11g (11.1.0.7)	jdbc:oracle:thin:@//localhost:1521:ora11i
--------	--------------------------------	---

• **Other Drivers**

The following drivers and databases have not been formally tested, but should work based on experience.

Driver	Version	Example URL
PostgreSQL	8.x	jdbc:postgresql://jsmith.mycompany.com:5432/postgres
MySQL	MySQL Connector/J 6.0	jdbc:mysql://localhost:3306/sample
Microsoft	Microsoft SQL Server 2005 JDBC Driver 1.2.2828.100	jdbc:sqlserver://localhost:1433;databaseName=travel;selectMethod=cursor
IBM	Redistributable DB2 JDBC Type 4 driver v8 fixpack 13	jdbc:db2://localhost:50002/sample
jTDS	jTDS 1.2.1	jdbc:jtds:sqlserver://test-mycompany.com:1433/travel
DataDirect	DataDirect Connect for JDBC - 3.6 DataDirect Connect for JDBC - 3.6.07	

**BACK-End**

The back end used for the project is MYSQL. MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by

Oracle Corporation.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production

environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

SQL supports many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, BINARY, VARBINARY, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and OpenGIS spatial types.

Clients can connect to MySQL Server using several protocols. It has a privilege and password system that is very flexible and secure, and that enables host-based verification. Password security by encryption of all password traffic when you connect to a server is also available.

## **CHAPTER 2**

### **REQUIREMENT ANALYSIS**

Project development must concentrate on the software engineering principles. So this project undergone through the various stages of software engineering. Thus it had gone through the requirement gathering phase. So the requirements gathered are as follows:

#### **Requirement gathering:**

Analysis gathers requirements through observation of existing systems, studying existing procedures, interview, task analysis, scenario analysis, by using internet, by contact etc.

#### **Requirement analysis:**

After gathering all requirements, the main purpose of requirement analysis is clearly understood the user requirements and detect inconsistencies and incompleteness. There are three main types of problems in the requirements that the analyst need to identify and resolve they are ambiguity, inconsistency, incompleteness.

After all these SRS document prepared. SRS is a software requirement specification document. It is useful in various contexts. They are statement of user needs, contract document, reference document and definition for implementation.

#### **Functional requirements:**

### **ADMIN**

#### ***1. Introduction***

The admin can start using the application by pressing ADMIN REGISTER button. Then, the ADMIN must register himself/herself to the application using a predefined security key. After successful registration, ADMIN can log in by ADMIN LOGIN button in the home page.

#### ***2. Creating Cut-off list***

Admin can create cut-off list for different subject each year and this gets saved in the database.

#### ***3. View Final list***

The Admin can view the final list of both qualified and disqualified students based on the cut-off marks provided by the Admin.

#### **4. *Update Student details***

The Admin can update or delete students' details.

### **STUDENT**

#### **1. *Introduction***

The Student can enter the application using the STUDENT APPLICATION button and enter the details of the student. The student's application will be successfully registered and saved in the database if they entered the appropriate details.

#### **2. *Student details updation or deletion***

Students can update and delete their details in case of corrections.

#### **3. *View final list***

The students can view the final qualified and disqualified list based on the cut-off mark provided by the admin.

#### **Non-Functional Requirements:**

It includes:

- Reliability issues
- Performance issues
- Human-computer interface issues
- Interface with other external systems, security, maintainability, etc.

After requirement analysis the SRS document is prepared.

## **SRS DOCUMENTATION**

A **software requirements specification (SRS)** is a description of a software system to be developed. It is modeled after **business requirements specification (CONOPS)**, also known as a stakeholder requirements specification (STRS). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

### **1. Introduction**

The project entitled “ADMISSION MANAGEMENT FOR COLLEGE” objective is to provide a system which manages the admissions of college students. There are mainly two types of users who access this system that is., Admin and Student. Both the types of users consumes less amount of time when they work through automated system. The system will take care of all the process in a quick manner. Data storing is easier too.

#### **a. Purpose**

The main purpose of this project is to admit the eligible students to a college based on the cut-off mark provided by the Admin.

b. Intended Audience

The project is mainly used by Students and Admin. Students use this site to apply for admissions in college while Admins or the college admission faculty use this site to admit eligible students to colleges.

c. Project scope

To admit students to colleges. Make sure that the program is simple and easy to use.

## **2. Overall Description**

The Activity “ADMISSION MANAGEMENT FOR COLLEGE” is based on admission of students by college admission management faculty. As soon as a student applies in this site by entering his/her details in the application form, he/she can view whether he/she is qualified for admissions or not. All data will be stored in the database.

a. Product Perspective

To provide efficient details. To provide accurate details of students.

b. Product features

- Login Interface

Students should enter valid username and password to access their profiles.

c. Design and Implementation Constraints

The adding, updating and deleting options. SQL command for above queries. Implement the database at least using the centralized database management system.

d. Software & Hardware requirements

- Software requirements
  - Front End:  
Advance Java(netbeans)
  - Back End:  
MySQL
- Hardware requirements
  - Windows 7
  - Core i3 and above
  - 2GB RAM

e. User documentation

User document gives customers the information they need to use the product. They are primarily teaching materials which include some technical explanation.

### **3. System Features**

#### **a. Module Description(Admin and User)**

“ADMISSION MANAGEMENT FOR COLLEGE” is an efficient application for maintaining Students’ records in a college. There are two types of users, i.e., ADMIN and USER. ADMIN can view students details, qualified and disqualified list, edit them and also define cut-off marks and also view the final qualified and disqualified list.



## CHAPTER 3

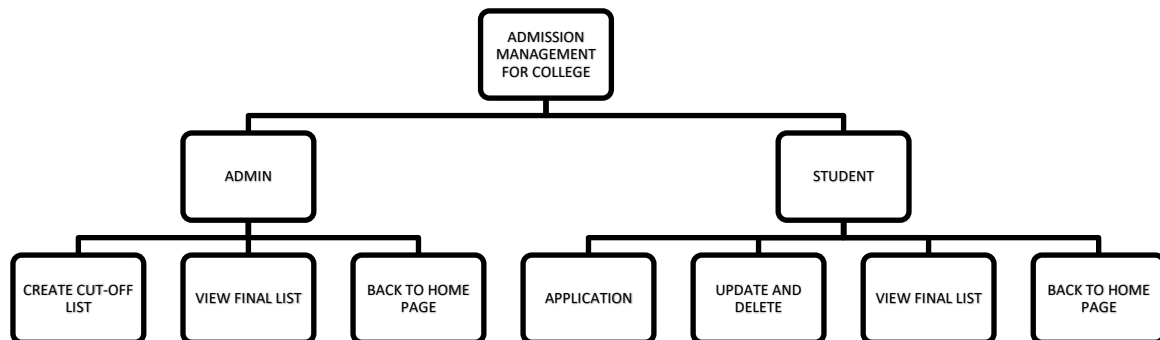
### SOFTWARE DESIGN

After requirement analysis the next phase undergone was software designing.

Designing activities are done through the following types of graphical methods.

#### **Structure Chart**

A Structure Chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name.

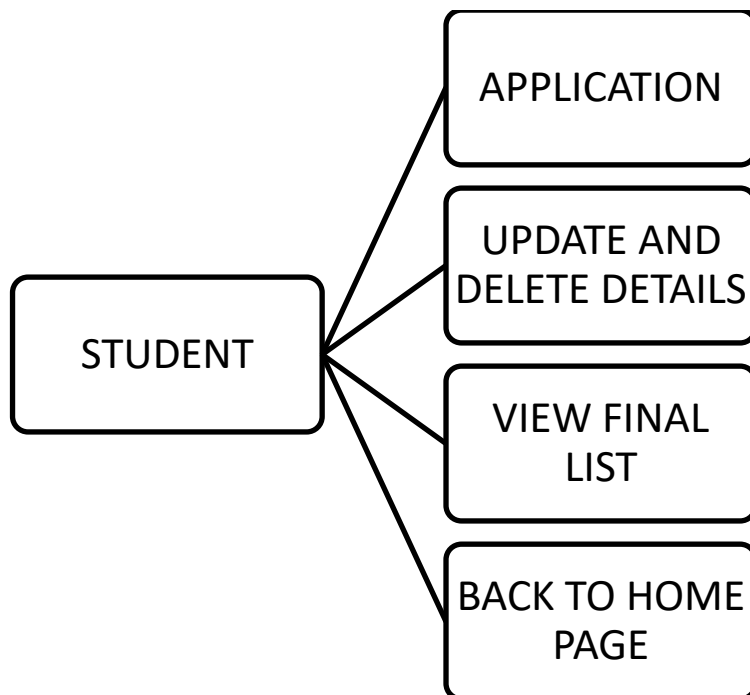
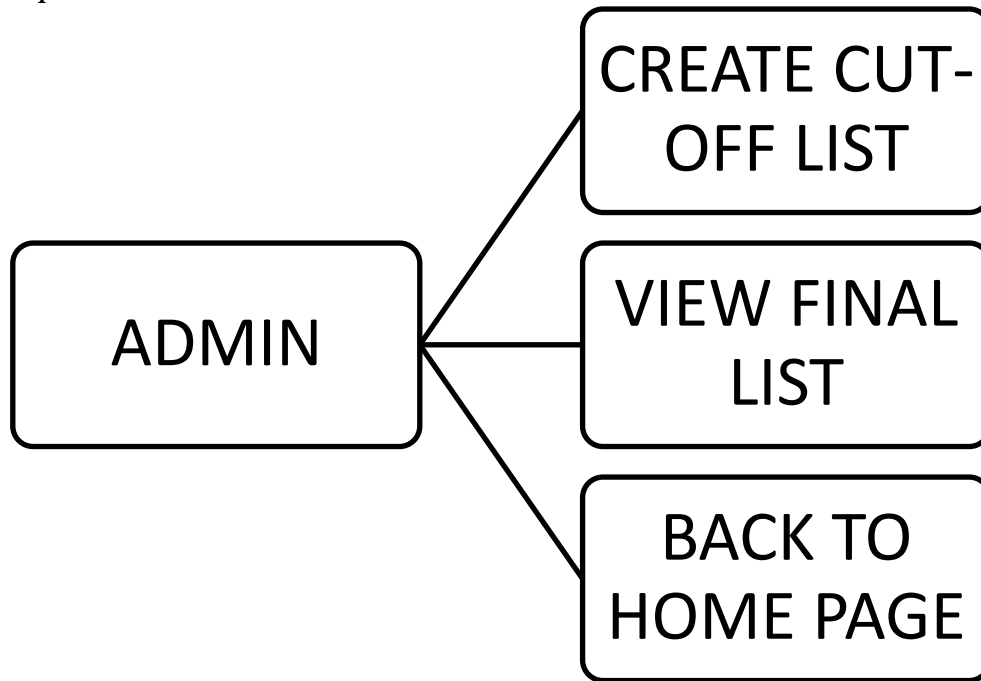


#### **UM Diagram**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

##### **a. Use Case Diagram**

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.



## **CHAPTER 4**

### **CODING AND IMPLIMENTATION**

In this chapter we are going to discuss about coding. Coding is undertaken once the design phase is complete and the design documents have been successfully reviewed. In the coding phase, every module specified in the design document is coded and unit tested. A module is tested independently as and when its coding is complete. After all the modules of a system have been coded and unit tested, the integration and system testing phase is undertaken. The objective of coding phase is to transform the design of a system into code in a high-level language, and to unit test this code.

Code review for a module is undertaken after the module successfully compiles. That is, all the syntax errors have been eliminated from the module. Code inspection and code walkthrough are the normally followed review types. The main objective of code walkthrough is to discover the algorithmic and logical errors in the code. The principal aim of code inspection is to check for the presence of some common types of errors that usually creep into code due to programmer oversights and to check whether coding standards have been adhered to.

The basic codes that are used in this project are as follows:

#### **JFRAME BUTTON ACTION** (transferring control from one page to another after clicking on a button)

**Eg.** Moving from Home page to security page

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent  
    evt) {  
    Security s=new Security();  
    s.setVisible(true);  
    this.dispose();  
    // TODO add your handling code here:  
}
```

## ADMIN REGISTRATION

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if(admin.getText().trim().isEmpty() || designation.getText().trim().isEmpty()  
    || usertxt.getText().trim().isEmpty() || password.getText().trim().isEmpty())  
    {  
        JOptionPane.showMessageDialog(rootPane, "please enter all the details");  
    }  
  
    try {  
        String name = admin.getText();  
        System.out.println("Name is:" + name);  
        String des = designation.getText();  
        String user = usertxt.getText();  
        int u = Integer.parseInt(user);  
        String pass = password.getText();  
        Connection con = Databaseconnector.getConnection();  
        Statement st = con.createStatement();  
        if (isValidPassword(pass))  
        {  
            String q = "insert into register values('" + name + "','" + des + "','" + u +  
            "','" + pass + "')";  
            st.executeUpdate(q);  
            System.out.println("Inserted successfully");  
            JOptionPane.showMessageDialog(rootPane, "Registration successfull");  
            Adminlogin l=new Adminlogin();  
            l.setVisible(true);  
            this.dispose();  
        }  
        // TODO add your handling code here:  
    } catch (SQLException ex) {  
        Logger.getLogger(Adminregister.class.getName()).log(Level.SEVERE,  
        null, ex);  
    }  
}
```

(After successful registration, log in page appears...)

## **ADMIN SECURITY PASSWORD CHECKING**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
String p=pass.getText();  
if (p.equals("Security123"))  
{  
    JOptionPane.showMessageDialog(rootPane,"Security Login successfull");  
    Adminregister ar=new Adminregister();  
    ar.setVisible(true);  
    this.dispose();  
}  
else{  
    JOptionPane.showMessageDialog(rootPane,"Security Login failed");  
}  
    // TODO add your handling code here:  
}
```

## **ADMIN LOG IN**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
if(userid.getText().trim().isEmpty() && password.getText().trim().isEmpty())  
{  
    userlabel.setText("Username is required");  
    passlabel.setText("Password is required");  
}  
else if(userid.getText().trim().isEmpty()){  
    userlabel.setText("Username is required");  
}  
else if(password.getText().trim().isEmpty()){  
    passlabel.setText("Password is required");  
}
```

```
}  
  
try {  
    String u = userid.getText();  
    String p = password.getText();  
    Connection con = Databaseconnector.getConnection();  
    Statement st = con.createStatement();  
    String s = "select * from register where UserID='" + u + "' &&  
Password='" + p + "'";  
    ResultSet rs=st.executeQuery(s);  
    if(rs.next())  
    {  
        JOptionPane.showMessageDialog(rootPane,"Login successfull");  
        Adminmainpage m=new Adminmainpage();  
        m.setVisible(true);  
        this.dispose();  
    }  
else  
    {  
        JOptionPane.showMessageDialog(rootPane,"Incorrect username or  
password");  
    }  
  
    // TODO add your handling code here:  
} catch (SQLException ex) {  
    Logger.getLogger(Adminlogin.class.getName()).log(Level.SEVERE,  
null, ex);
```

}

(After successful log in, Admin main page appears)

## **CREATE CUT-OFF LIST**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if(maths.getText().trim().isEmpty() || physics.getText().trim().isEmpty() ||  
        chemistry.getText().trim().isEmpty() || year.getText().trim().isEmpty())  
    {  
        JOptionPane.showMessageDialog(rootPane,"Please enter all the details...");  
    }  
  
    try {  
        String math = maths.getText();  
        int m=Integer.parseInt(math);  
        String phy = physics.getText();  
        int p=Integer.parseInt(phy);  
        String chem = chemistry.getText();  
        int c=Integer.parseInt(chem);  
        String y=year.getText();  
        int tm=m+p+c;  
        String convert=Integer.toString(tm);  
        total.setText(convert);  
        Connection con = Databaseconnector.getConnection();  
        Statement st = con.createStatement();  
        String i="insert into cutoff
```

```
values(""+m+"",""+p+"",""+c+"",""+y+"",""+tm+"");  
  
    st.executeUpdate(i);  
  
    System.out.println("Inserted successfully");  
  
    JOptionPane.showMessageDialog(rootPane,"Saved successfully");  
  
    // TODO add your handling code here:  
  
} catch (SQLException ex) {  
  
    Logger.getLogger(cutoff.class.getName()).log(Level.SEVERE, null, ex);  
  
}  
  
}
```

## **CLEAR**

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    admin.setText(null);  
  
    designation.setText(null);  
  
    usertxt.setText(null);  
  
    password.setText(null);  
  
    // TODO add your handling code here:  
  
}
```

## **BACK TO HOME PAGE**

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



}

## **STUDENT APPLICATION FORM**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if(name.getText().trim().isEmpty() || age.getText().trim().isEmpty() ||  
    cont.getText().trim().isEmpty() || user.getText().trim().isEmpty() ||  
    maths.getText().trim().isEmpty() || phy.getText().trim().isEmpty() ||  
    chem.getText().trim().isEmpty())  
    {  
        JOptionPane.showMessageDialog(rootPane, "please enter all the details");  
    }  
  
    try {  
        String n = name.getText();  
        String aget =age.getText();  
        int a=Integer.parseInt(aget);  
        male.setActionCommand("Male");  
        female.setActionCommand("female");  
        other.setActionCommand("other");  
        String g = buttonGroup1.getSelection().getActionCommand();  
        String contact = cont.getText();  
        int c = Integer.parseInt(contact);  
        String use = user.getText();  
        int u = Integer.parseInt(use);  
        String math = maths.getText();  
        int m = Integer.parseInt(math);
```

```
String physics = phy.getText();

int p = Integer.parseInt(physics);

String chemistry = chem.getText();

int ch = Integer.parseInt(chemistry);

int t=m+p+ch;

String conv=Integer.toString(t);

Totalmarks.setText(conv);

Connection con = Databaseconnector.getConnection();

Statement st = con.createStatement();

String s="insert into student
values('"+n+"','"+a+"','"+g+"','"+c+"','"+u+"','"+m+"','"+p+"','"+ch+"','"+t+"')";

st.executeUpdate(s);

System.out.println("Applied successfully");

JOptionPane.showMessageDialog(rootPane,"application registered");

// TODO add your handling code here:


} catch (SQLException ex) {

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

}

}
```

## **DATABASE CONNECTOR CODE**

```
public class Databaseconnector {
```

```
static Connection con;
```

```
public static Connection getConnection() {  
    try  
    {  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        con =  
DriverManager.getConnection("jdbc:mysql://localhost:3308/admission","root",""  
);  
        System.out.println("connected");  
    }  
    catch(Exception e)  
    {  
        System.out.println("class error: "+e.getMessage());  
    }  
    return con;  
}  
  
}
```

## **SHOW DETAILS OF STUDENTS AFTER ENTERING THEIR USER ID**

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        String userid=user.getText();
```

```
Connection con=Databaseconnector.getConnection();

Statement st=con.createStatement();

String d="select * from student where UserID='"+userid+"'";

ResultSet rs=st.executeQuery(d);

if (rs.next()) {

    String n=rs.getString("Name");

    String a=rs.getString("Age");

    String g=rs.getString("Gender");

    String c=rs.getString("Contact");

    String m=rs.getString("Maths");

    int ma = Integer.parseInt(m);

    String p=rs.getString("Physics");

    int ph = Integer.parseInt(p);

    String ch=rs.getString("Chemistry");

    int che = Integer.parseInt(ch);

    int to=ma+ph+che;

    String conv=Integer.toString(to);

    name.setText(n);

    age.setText(a);

    gender.setText(g);

    contact.setText(c);

    maths.setText(m);

    physics.setText(p);

    chemistry.setText(ch);

    total.setText(conv);

}
```

```
    }  
  
    else  
  
    {  
  
        name.setText(null);  
        age.setText(null);  
        gender.setText(null);  
        contact.setText(null);  
        maths.setText(null);  
        physics.setText(null);  
        chemistry.setText(null);  
        total.setText(null);  
  
        JOptionPane.showMessageDialog(rootPane,"Please enter the correct  
UserID...");  
  
    }  
  
        // TODO add your handling code here:  
  
        } catch (SQLException ex) {  
  
            Logger.getLogger(Updateddelete.class.getName()).log(Level.SEVERE,  
null, ex);  
  
        }  
  
    }
```

## **UPDATE STUDENT DETAILS**

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    if (user.getText().trim().isEmpty())  
  
    {  
  

```

```
JOptionPane.showMessageDialog(rootPane,"Please enter the UserID...");
}

try {
    String u=user.getText().toString();
    String n=name.getText().toString();
    String a=age.getText().toString();
    String g=gender.getText().toString();
    String c=contact.getText().toString();
    String m=maths.getText().toString();
    String p=physics.getText().toString();
    String ch=chemistry.getText().toString();
    String t=total.getText().toString();
    Connection con=Databaseconnector.getConnection();
    Statement s=con.createStatement();
    String up="update student set
Name='"+n+"',Age='"+a+"',Gender='"+g+"',Contact='"+c+"',Maths='"+m+"',Phy
sics='"+p+"',Chemistry='"+ch+"',Totalmarks='"+t+"' where UserID='"+u+"'";
    s.executeUpdate(up);
    JOptionPane.showMessageDialog(rootPane,"Updated successfully");
    user.setText(null);
    name.setText(null);
    age.setText(null);
    gender.setText(null);
    contact.setText(null);
    maths.setText(null);
```

```
physics.setText(null);
```

```
chemistry.setText(null);
```

```
total.setText(null);
```

```
// TODO add your handling code here:
```

```
} catch (SQLException ex) {
```

```
    Logger.getLogger(Updateddelete.class.getName()).log(Level.SEVERE,
```

```
    null, ex);
```

```
}
```

```
}
```

## **DELETE STUDENT DETAILS**

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

```
if (user.getText().trim().isEmpty() || name.getText().trim().isEmpty())
```

```
{
```

```
    JOptionPane.showMessageDialog(rootPane,"Please enter a valid UserID");
```

```
}
```

```
else{
```

```
    try {
```

```
        String uid=user.getText().toString();
```

```
        Connection con=Databaseconnector.getConnection();
```

```
        Statement st=con.createStatement();
```

```
        String d="delete from student where UserID='"+uid+"'";
```

```
        st.executeUpdate(d);
```

```
        JOptionPane.showMessageDialog(rootPane,"Deleted successfully");
```

```
// TODO add your handling code here:

} catch (SQLException ex) {

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

}

}

}
```

## **FINAL LIST**

### **Qualified list**

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

    try {

        Connection con = Databaseconnector.getConnection();

        Statement st = con.createStatement();

        DefaultTableModel model=new DefaultTableModel(new
String[]{"name","UserID","Totalmarks"},0);

        String s="select * from student where Totalmarks>=245";

        ResultSet rs=st.executeQuery(s);

        while(rs.next())

        {

            String n=rs.getString("name");

            String u=rs.getString("UserID");

            String t=rs.getString("Totalmarks");

            model.addRow(new Object[]{n,u,t});

        }

    }

}
```



```
        table.setModel(model);

    }

    // TODO add your handling code here:
} catch (SQLException ex) {

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

}

}
```

### **Disqualified list**

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

    try {

        Connection con = Databaseconnector.getConnection();

        Statement st = con.createStatement();

        DefaultTableModel model = new DefaultTableModel(new
String[]{"name","UserID","Totalmarks"}, 0);

        String dis = "select * from student where Totalmarks<245";

        ResultSet rst = st.executeQuery(dis);

        while (rst.next()) {

            String d = rst.getString("name");

            String u=rst.getString("UserID");

            String t=rst.getString("Totalmarks");

            model.addRow(new Object[]{d,u,t});

            table2.setModel(model);

        }

    }

}
```

```
// TODO add your handling code here:  
// TODO add your handling code here:  
} catch (SQLException ex) {  
    Logger.getLogger(finallist.class.getName()).log(Level.SEVERE, null,  
ex);  
}  
}
```

## CHAPTER 5

### TABLES

#### 1. Add Admin Register

It is used to add details of Admin.

```
c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
mysql>
mysql> use admission;
Database changed
mysql> show tables;
+-----+
| Tables_in_admission |
+-----+
| cutoff               |
| register             |
| student              |
+-----+
3 rows in set (0.43 sec)

mysql> desc register;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| adminname  | varchar(40) | NO   |     | NULL    |       |
| Designation | varchar(40) | NO   |     | NULL    |       |
| UserId     | int(11)    | YES  | UNI | NULL    |       |
| Password   | varchar(40) | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.27 sec)

mysql> select * from register;
+-----+-----+-----+-----+
| adminname  | Designation | UserId | Password |
+-----+-----+-----+-----+
| T S Parvathy | Admin      | 67    | Paruts12 |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> _
```

#### 2. Create cut-off list

Used to create and cut-off list determined by the Admin

```
c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
1 row in set (0.01 sec)

mysql> desc cutoff;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Maths | int(11) | YES | | NULL | |
| Physics | int(11) | YES | | NULL | |
| Chemistry | int(11) | YES | | NULL | |
| Year | int(11) | YES | | NULL | |
| Totalmarks | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select * from cutoff;
+-----+-----+-----+-----+-----+
| Maths | Physics | Chemistry | Year | Totalmarks |
+-----+-----+-----+-----+-----+
| 85 | 80 | 80 | 2021 | 245 |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
```

### 3.Student Table

Used to add Student details which are entered by the students through their application forms.(Updation and Deletion is possible for both Admin and Students.)

```
c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
mysql>
mysql>
mysql>
mysql>
mysql> select * from student;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Name | Age | Gender | Contact | UserID | Maths | Physics | Chemistry | Totalmarks |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Amelia | 18 | female | 684264126 | 1001 | 85 | 51 | 56 | 192 |
| Aarav | 18 | Male | 955849651 | 1002 | 84 | 58 | 78 | 220 |
| Benny | 19 | Male | 842812692 | 1003 | 98 | 88 | 92 | 278 |
| Beena | 19 | Male | 245894645 | 1004 | 87 | 68 | 78 | 233 |
| Catherine | 18 | female | 36985247 | 1005 | 88 | 97 | 88 | 273 |
| Emily | 18 | female | 98567481 | 1006 | 79 | 58 | 88 | 225 |
| Emmanuel | 19 | Male | 455641642 | 1007 | 87 | 88 | 89 | 264 |
| Farhan | 19 | Male | 574878541 | 1008 | 98 | 87 | 87 | 272 |
| Gayathri | 19 | female | 985697458 | 1009 | 87 | 95 | 87 | 269 |
| Helen | 18 | female | 789874585 | 1010 | 97 | 87 | 84 | 268 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> _
```

## **CHAPTER 6**

### **TESTING**

The aim of program testing is to identify all defects in a program. Testing a program involves providing the program with a set of test inputs or test cases and observing if the program behaves as expected. If the program fails to behave as expected, then the input data and the conditions are noted for later debugging and error correction. An error is a mistake committed in any of the development phases. A failure is a manifestation of an error. A test case is the triplet [I, S, O], where I is the data input to the system, S is the state of the system at which the data is input, and O is the expected output of the system. A test suite is the set of all test cases with which a software product is tested. Verification is concerned with phase containment of errors, and aim of validation is to make the final product error free. Testing involves four activities such as test suite design, running test cases and checking the results to detect failures, debugging and error correction. A minimal test suite is a carefully designed set of test cases such that each test case helps detect different errors.

A software product is normally tested in three levels or stages:

- Unit testing: It is undertaken after a module has been coded and reviewed. Unit testing is referred to as testing in the small and it makes debugging easier.
- Integration testing: The individually tested modules are integrated and tested after each step of integration is called integration testing. It is known as testing in the large. If a failure is detected when an integrated set of modules is being tested, it would be difficult to determine which module exactly has the error.
- System testing: The fully integrated system is tested in system testing. It is also called testing in large. It is carried out in the testing phase.

There are essentially two main approaches to systematically design test cases:

- Black-box approach: In this approach, test cases are designed using only the functional specification of the software. It does not require any knowledge of the internal structure of

a program. So, black box testing is also known as functional testing. The two main approaches available to design black-box test cases are equivalence class partitioning and boundary value analysis. The main idea behind defining equivalence classes of input data is that, testing the code with any one value belonging to an equivalence class is as good as testing the code with any other value belonging to the same equivalence class. Boundary value analysis-based test suite design involves designing test cases using the values at the boundaries of different equivalence classes.

**White-box or glass-box approach:** It requires a thorough knowledge of the internal structure of a program, so it is also called structural testing. White-box testing is an important type of unit testing.

**Basic concepts:** A white-box testing strategy can either be coverage-based or fault-based. A fault-based testing strategy targets to detect certain types of faults. These faults that a test strategy focuses on constitute the fault model of the strategy. A coverage-based testing strategy attempts to execute or cover certain elements of a program. Popular examples of coverage-based testing strategies are statement coverage, branch coverage, and path coverage-based testing. The set of specific program elements that a testing strategy requires to be executed is called the testing criterion of the strategy. A white-box testing strategy is said to be stronger than another strategy, if all types of program elements covered by the second testing strategy are also covered by the first testing strategy, and the first strategy additionally covers some more types of elements not covered by the second strategy. When none of two testing strategies fully covers the program elements exercised by the other, then the two are called complementary testing strategies. If a stronger testing has been performed, then a weaker testing need not be carried out.

## **CHAPTER 7**

### **RESULT**

The project entitled “Admission management for College” after going through a series of phase like requirement analysis, design phase, coding and testing. The various limitations of current system were diagnosed and are improved in this work. After considering various feasible solutions, the most feasible one was selected for designing by considering time and efficiency as constraints. The performance of the system was evaluated to determine whether system achieved the results that were expected and whether predicted benefits of the system were being realized. The results are obtained in a timely and constraints manner. Since each process is implemented as simple modules, the system is liable to further modification and also provides easy maintenance. The project contains a lot of improvement in it. But the overall look and feel of the project gives the rough picture of an automated version of existing systems. The performance of the system was evaluated to determine whether system achieved the results that were expected and whether predicted benefits of the system were being realized.

## **CHAPTER 8**

### **CONCLUSION**

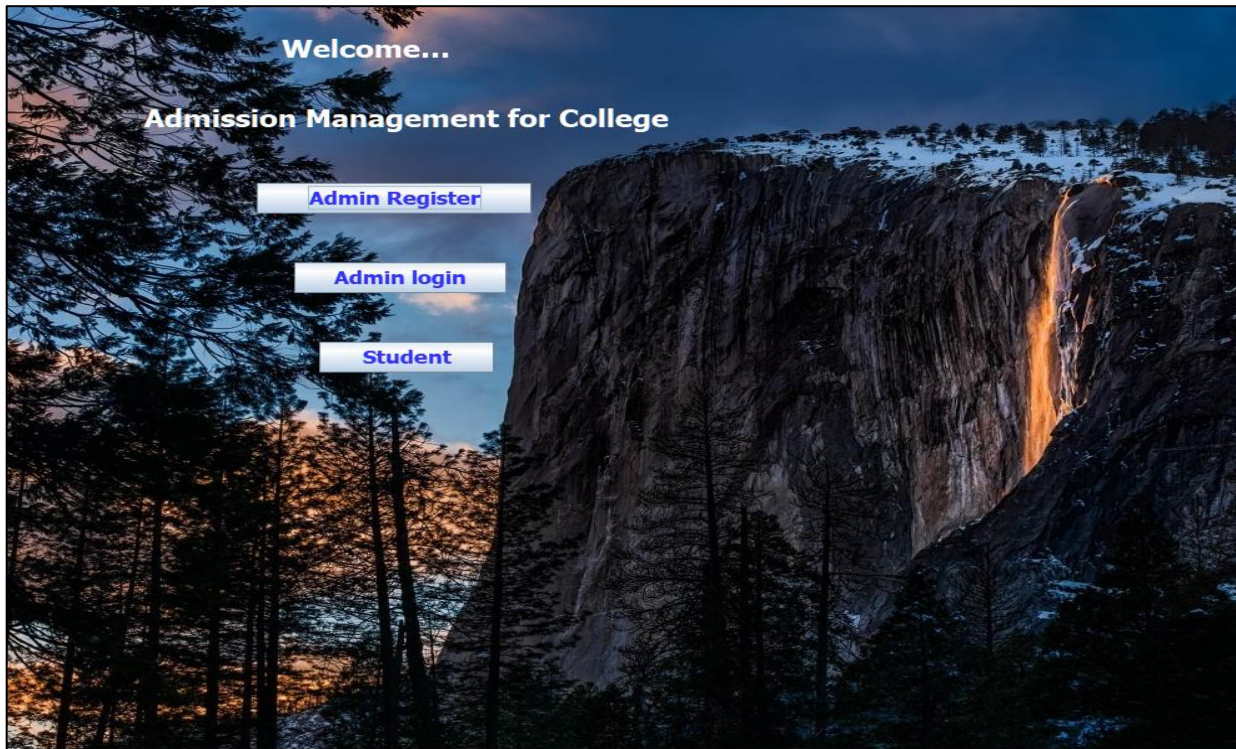
The program on Application Development lab using java was completed successfully. Each module in this project was successfully run. The project entitled “Admission management for College” was made operational after going through a series of phases like requirement analysis, design phase, coding and testing. The various limitations of current system were diagnosed and are improved in this project. After considering various feasible solutions, the most feasible one was selected for designing by considering time and efficiency as constraints. The performance of the system was evaluated to determine whether system achieved the results that were expected and whether the predicted benefits of the system were realized.

The results are obtained in a timely and constrained manner. Since each process is implemented as simple modules, the system is liable to further modification and also provides easy maintenance. The project contains a lot of scope of improvement in it. But the overall look and feel of the project gives the rough picture of an automated version of existing system. The screenshots of basic frames are included in the appendix.



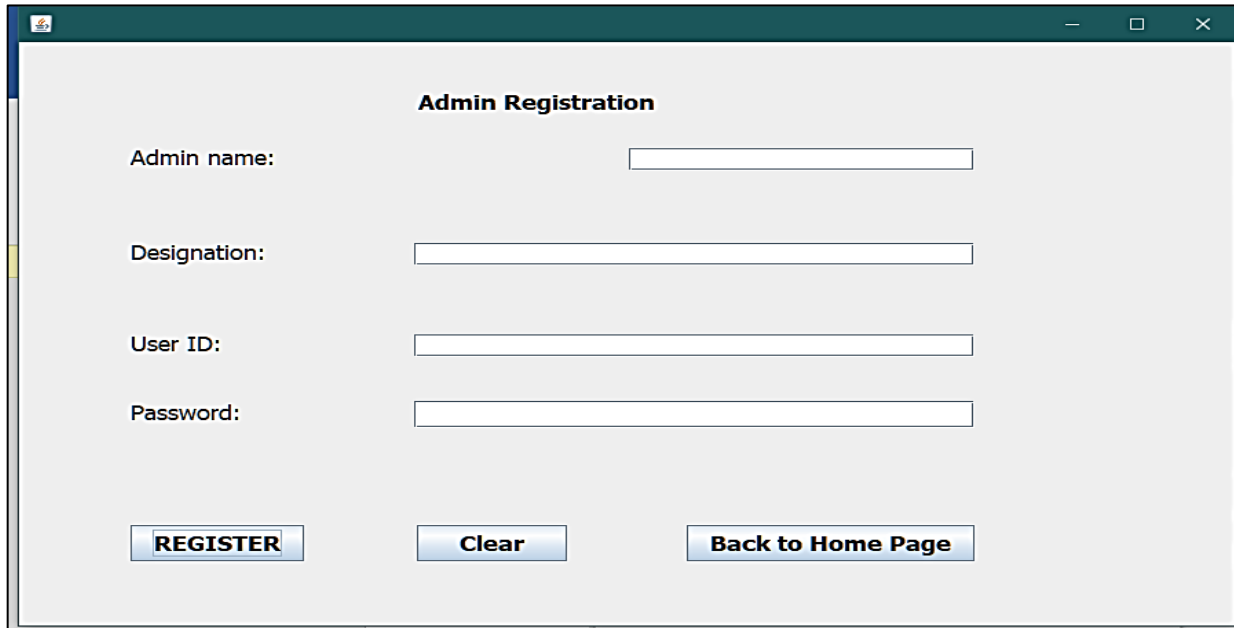
## APPENDIX

### HOME PAGE



### SECURITY PASSWORD

The image shows a small, standard Windows-style dialog box titled 'Enter the Security Password'. It has a light gray background and a dark green title bar with minimize, maximize, and close buttons. Inside the dialog, the text 'Password:' is followed by a single-line text input field. Below the input field are two buttons: 'Enter' and 'Clear', both with a blue gradient and white text.

**ADMIN REGISTRATION**

A screenshot of a web application window titled "Admin Registration". The window has a light gray background and a dark green title bar. It contains four input fields: "Admin name:", "Designation:", "User ID:", and "Password:". Below the fields are three buttons: "REGISTER", "Clear", and "Back to Home Page".

**Admin Registration**

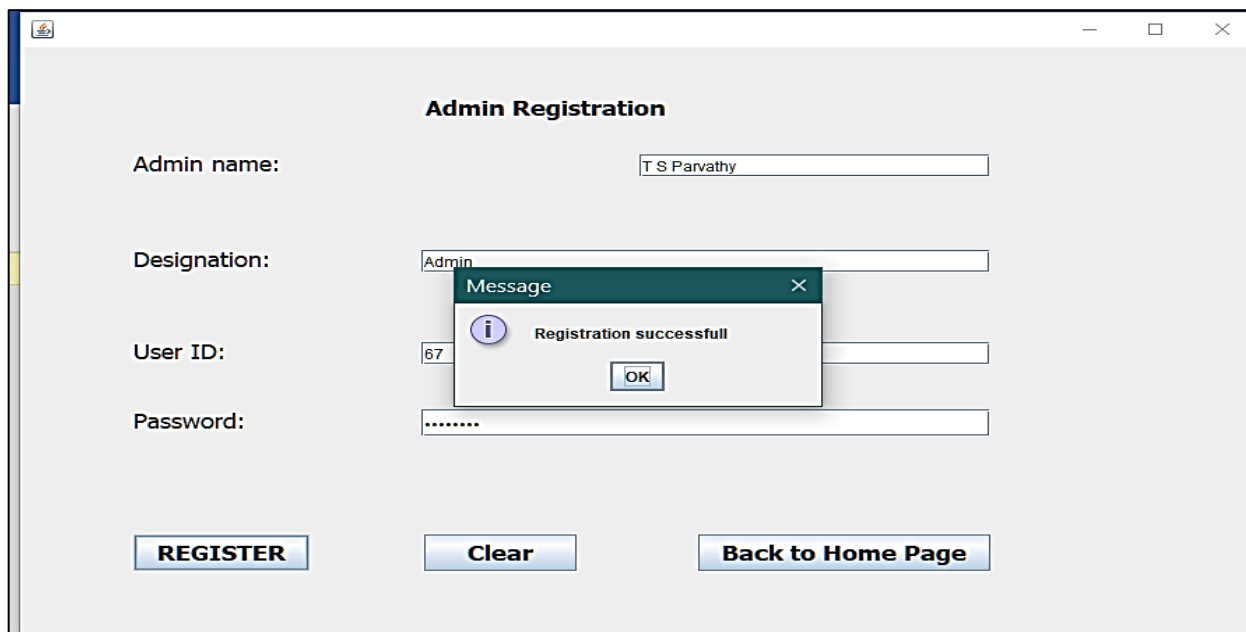
Admin name:

Designation:

User ID:

Password:

**REGISTER** **Clear** **Back to Home Page**



A screenshot of the same "Admin Registration" form, but with data entered in the fields and a success message displayed. The "Admin name" field contains "T S Parvathy", "Designation" contains "Admin", "User ID" contains "67", and "Password" contains ".....". A modal message box titled "Message" is overlaid on the form, displaying an information icon, the text "Registration successfull", and an "OK" button. The "REGISTER", "Clear", and "Back to Home Page" buttons are still visible at the bottom.

**Admin Registration**

Admin name:

Designation:

User ID:

Password:

**REGISTER** **Clear** **Back to Home Page**

**Message**

Registration successfull

**OK**

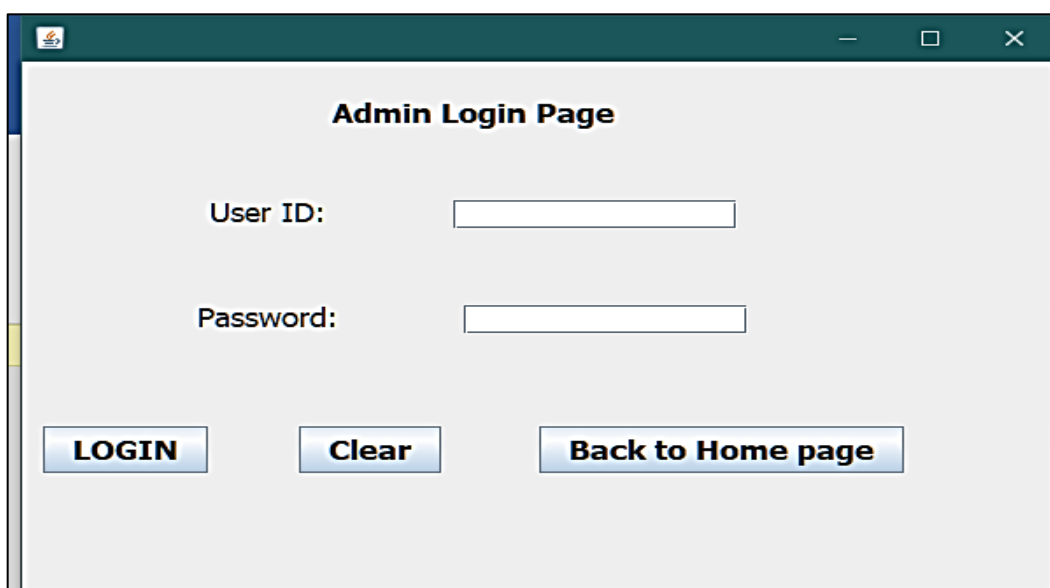
```
c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
mysql> use admission;
Database changed
mysql> show tables;
+-----+
| Tables_in_admission |
+-----+
| cutoff               |
| register             |
| student              |
+-----+
3 rows in set (0.43 sec)

mysql> desc register;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| adminname      | varchar(40)   | NO   |     | NULL    |       |
| Designation    | varchar(40)   | NO   |     | NULL    |       |
| UserId         | int(11)       | YES  | UNI | NULL    |       |
| Password       | varchar(40)   | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.27 sec)

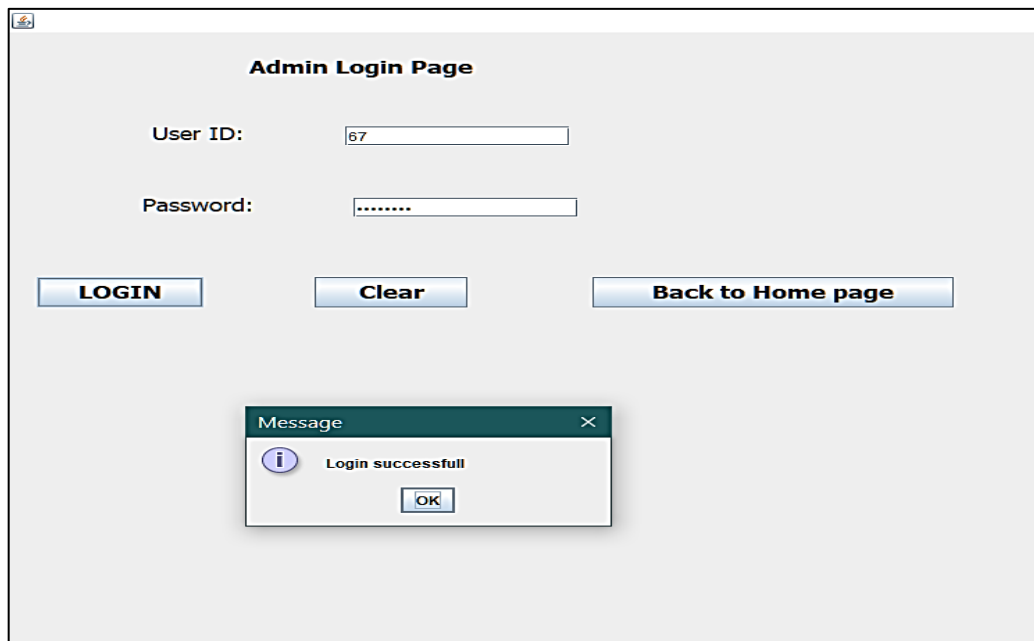
mysql> select * from register;
+-----+-----+-----+-----+
| adminname | Designation | UserId | Password |
+-----+-----+-----+-----+
| T S Parvathy | Admin      | 67    | Paruts12 |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> _
```

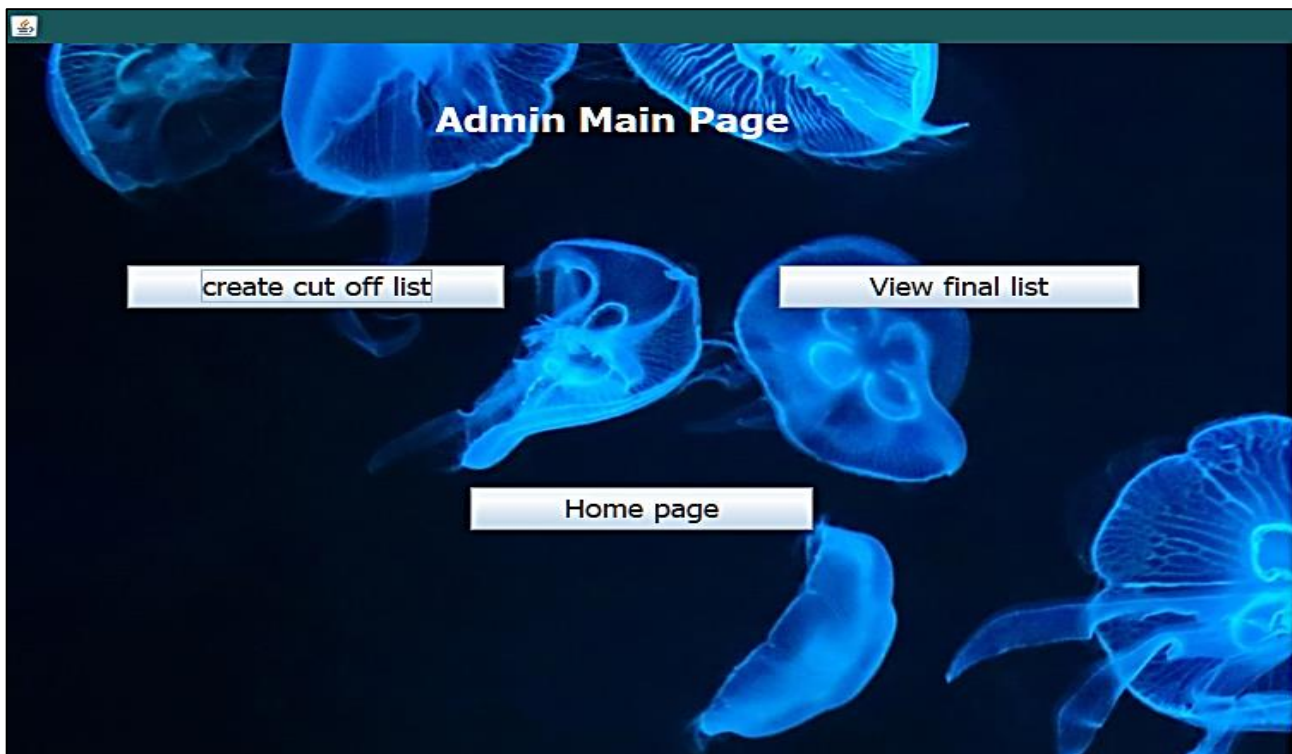
## ADMIN LOGIN PAGE



The screenshot shows a web browser window titled "Admin Login Page". The page has a light gray background. At the top, the title "Admin Login Page" is centered. Below the title, there are two input fields: "User ID:" and "Password:". Each input field is a simple white rectangle with a thin border. Below the input fields, there are three buttons: "LOGIN", "Clear", and "Back to Home page". The "LOGIN" button is on the left, "Clear" is in the middle, and "Back to Home page" is on the right. All buttons have a blue gradient and a 3D effect.



### ADMIN MAIN PAGE



**CREATE CUT-OFF LIST**

Create Cut-off list

Maths:

Physics:

Chemistry:

Year:

Total marks:

Save

Clear

Back to Admin Mainpage

Create Cut-off list

Maths:

85

Physics:

80

Chemistry:

80

Year:

2021

Total marks:

245

Save

Clear

Back to Admin Mainpage

Message

Saved successfully

OK



```

c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
1 row in set (0.01 sec)

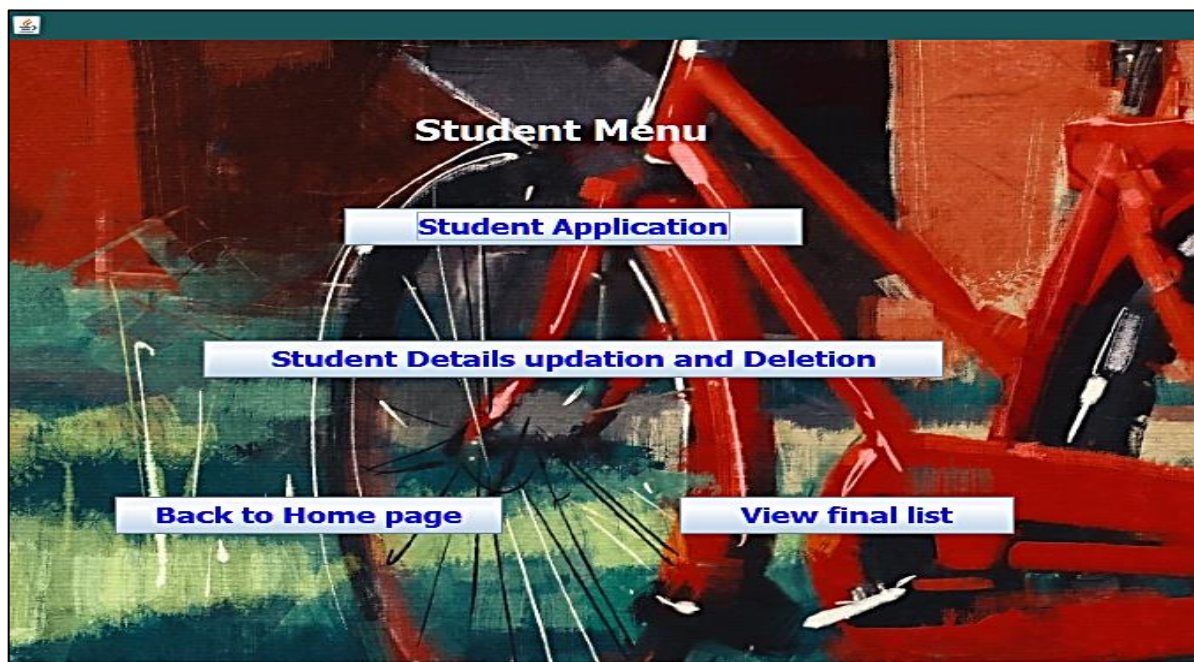
mysql> desc cutoff;
+-----+-----+-----+-----+-----+-----+
| Field      | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Maths      | int(11) | YES  |     | NULL    |       |
| Physics    | int(11) | YES  |     | NULL    |       |
| Chemistry  | int(11) | YES  |     | NULL    |       |
| Year       | int(11) | YES  |     | NULL    |       |
| Totalmarks | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

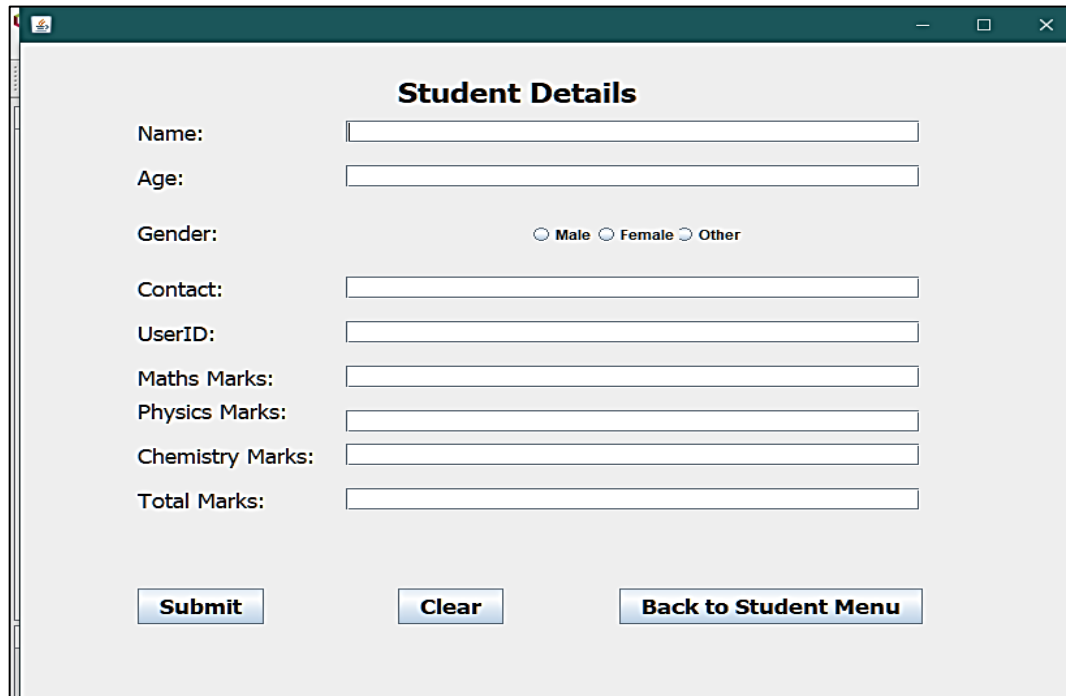
mysql> select * from cutoff;
+-----+-----+-----+-----+-----+
| Maths | Physics | Chemistry | Year | Totalmarks |
+-----+-----+-----+-----+-----+
| 85    | 80     | 80        | 2021 | 245        |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>

```

## STUDENT MENU

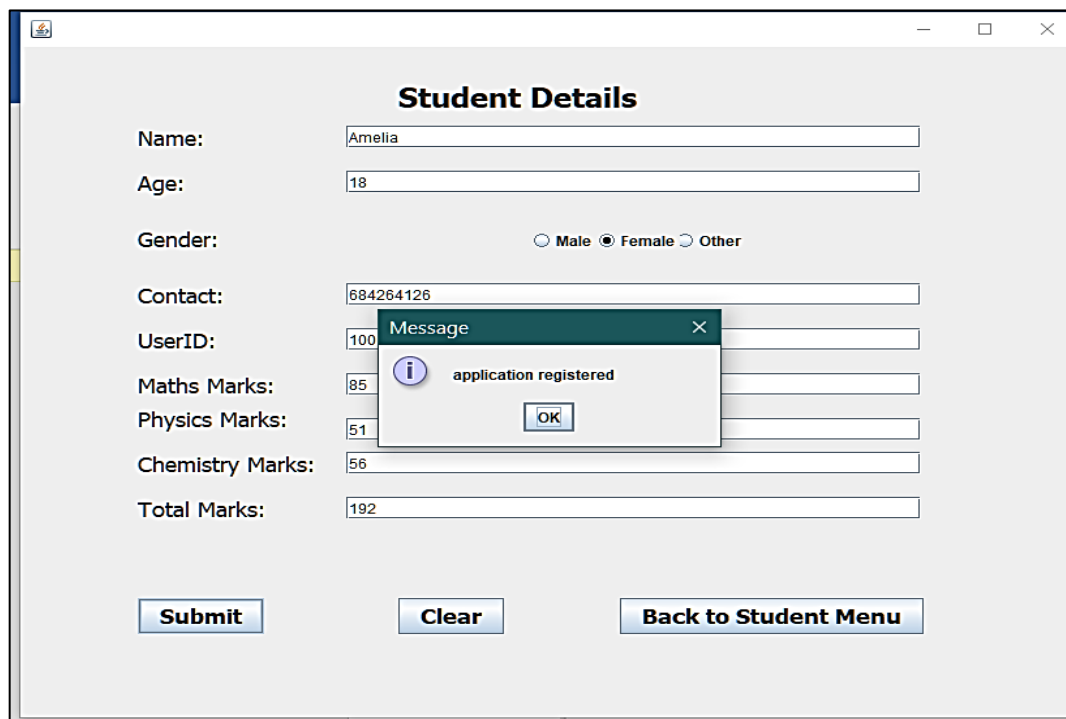


**STUDENT APPLICATION**

A screenshot of a web application window titled "Student Details". The form contains the following fields and controls:

- Name:
- Age:
- Gender: ☐ Male ☐ Female ☐ Other
- Contact:
- UserID:
- Maths Marks:
- Physics Marks:
- Chemistry Marks:
- Total Marks:

At the bottom of the form are three buttons: "Submit", "Clear", and "Back to Student Menu".



A screenshot of the same "Student Details" form, now with data entered in the input fields. A message dialog box is overlaid on the form, displaying the text "application registered" with an "OK" button.

Student Details

Name:

Age:

Gender: ☐ Male ☒ Female ☐ Other

Contact:

UserID:

Maths Marks:

Physics Marks:

Chemistry Marks:

Total Marks:

Buttons: Submit, Clear, Back to Student Menu

Message dialog box: application registered (OK)

```

c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
mysql>
mysql>
mysql>
mysql>
mysql> select * from student;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Name   | Age  | Gender | Contact | UserID | Maths | Physics | Chemistry | Totalmarks |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Amelia | 18   | female | 684264126 | 1001   | 85    | 51     | 56        | 192        |
| Aarav  | 18   | Male   | 955849651 | 1002   | 84    | 58     | 78        | 220        |
| Benny  | 19   | Male   | 842812692 | 1003   | 98    | 88     | 92        | 278        |
| Beena  | 19   | Male   | 245894645 | 1004   | 87    | 68     | 78        | 233        |
| Catherine | 18   | female | 36985247  | 1005   | 88    | 97     | 88        | 273        |
| Emily  | 18   | female | 98567481  | 1006   | 79    | 58     | 88        | 225        |
| Emmanuel | 19   | Male   | 455641642 | 1007   | 87    | 88     | 89        | 264        |
| Farhan | 19   | Male   | 574878541 | 1008   | 98    | 87     | 87        | 272        |
| Gayathri | 19   | female | 985697458 | 1009   | 87    | 95     | 87        | 269        |
| Helen  | 18   | female | 789874585 | 1010   | 97    | 87     | 84        | 268        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>

```

## STUDENT DETAILS UPDATION AND DELETION

### Student Details Updation and Deletion

Enter User ID:

Name:

Age:

Gender:

Contact:

Maths marks:

Physics marks:

Chemistry marks:

Total Marks:



**Student Details Updation and Deletion**

Enter User ID:  [Show Details](#)

Name:

Age:

Gender:

Contact:

Maths marks:

Physics marks:

Chemistry marks:

Total Marks:

[Update](#) [Clear](#) [Delete](#)

[Back to Student's Main Page](#)

**Student Details Updation and Deletion**

Enter User ID:  [Show Details](#)

Name:

Age:

Gender:

Contact:

Maths marks:

Physics marks:

Chemistry marks:

Total Marks:

[Update](#) [Clear](#) [Delete](#)

[Back to Student's Main Page](#)

### Student Details Updation and Deletion

Enter User ID:

Name:

Age:

Gender:

Contact:

Maths marks:

Physics marks:

Chemistry marks:

Total Marks:

**Message** ×

Updated successfully

c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe

```
mysql>
mysql>
mysql>
mysql>
mysql> select * from student;
```

Name	Age	Gender	Contact	UserID	Maths	Physics	Chemistry	Totalmarks
Amelia	18	Female	684264126	1001	85	51	56	192
Aarav	18	Male	955849651	1002	84	58	78	220
Benny	19	Male	842812692	1003	98	88	92	278
Beena	19	Female	245894645	1004	87	68	78	233
Catherine	18	Female	36985247	1005	88	97	88	233
Emily	18	Female	98567481	1006	79	58	88	233
Emmanuel	19	Male	455641642	1007	87	88	89	233
Farhan	19	Male	574878541	1008	98	87	87	272
Gayathri	19	Female	985697458	1009	87	95	87	269
Helen	18	Female	789874585	1010	97	87	84	268

10 rows in set (0.02 sec)

```
mysql>
```

**Student Details Updation and Deletion**

Enter User ID:

Name:

Age:

Gender:

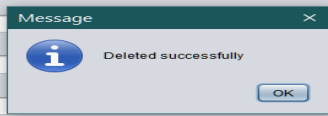
Contact:

Maths marks:

Physics marks:

Chemistry marks:

Total Marks:



```
c:\wamp64\bin\mysql\mysql8.0.18\bin\mysql.exe
```

```
mysql>
mysql>
mysql> select * from student;
```

Name	Age	Gender	Contact	UserID	Maths	Physics	Chemistry	Totalmarks
Amelia	18	Female	684264126	1001	85	51	56	192
Aarav	18	Male	955849651	1002	84	58	78	220
Benny	19	Male	842812692	1003	98	88	92	278
Beena	19	Female	245894645	1004	87	68	78	233
Catherine	18	Female	36985247	1005	88	97	88	273
Emily	18	Female	98567481	1006	79	58	88	225
Emmanuel	19	Male	455641642	1007	87	88	89	264
Farhan	19	Male	574878541	1008	98	87	87	272
Gayathri	19	Female	985697458	1009	87	95	87	269

```
9 rows in set (0.00 sec)
```

```
mysql>
```

## FINAL LIST

## Final list

[Click here to view the final list](#)

[Click here to view disqualified list](#)

name	UserID	Totalmarks
Benny	1003	278
Catherine	1005	273
Emmanuel	1007	264
Farhan	1008	272
Gayathri	1009	269

name	UserID	Totalmarks
Amelia	1001	192
Aarav	1002	220
Beena	1004	233
Emily	1006	225

[Back to Admin Main Page](#)

[Back to Student main page](#)