JAVA PROJECT – GATE PASS APPLICATION

KMIT

PROJECT NAME: GATE PASS APPLICATION

MEMBERS:

1.S. SHRAVAN (GROUP LEADER)-21BD1A057K

2.G. SANJANA-21BD1A057L

3.P. SAI ABHILESH-21BD1A057A

4.G. NIHAL -21BD1A056P

5.A. SOUMITH-21BD1A057M

6.R. NANDINI-21BD1A057F

TECHNOLOGY USED:

BUSINESS LOGIC – JAVA

FRONT END- HTML, CSS

BACK END- MYSQL

CLASS/SECTION – CSE/D

JAVA PROJECT – GATE PASS APPLICATION

ABSTRACT:

This Java program is a simple Gate Pass Application that allows issuing gate passes to students and employees. It also provides functionality to manage user details (add, update, delete) in a MySQL database.

Here's a brief overview of the classes and their functionalities:

1. `User`: This is a base class representing a general user. It has a name and ID as its attributes.

2. `Admin`: This class represents an admin who can issue gate passes to users. An admin has a name and an ID.

3. `Employee`: This class extends `User` and represents an employee. It inherits the name and ID attributes from the `User` class.

4. `Student`: This class extends `User` and represents a student. In addition to the name and ID, it also has attributes for the student's course and year.

5. `GatePass`: This class represents a gate pass issued to a user. It contains a user (either an employee or a student) and a purpose for which the gate pass is issued.

6. `GatePassApplication`: This is the main class that drives the GatePass Application. It contains methods to connect to a MySQL database, retrieve user details, add new users (students and employees), update user details, and delete users. It also allows issuing gate passes and interacting with the user through a console-based menu.

Here's a high-level explanation of the flow of the program:

1. The program establishes a connection to the MySQL database using JDBC.

2. The user is presented with a menu of options to choose from.

3. Based on the user's choice, the program executes the corresponding functionality (issue gate pass, add/update/delete student/employee).

JAVA PROJECT – GATE PASS APPLICATION

4. The user can choose to exit the program, which closes the database connection and terminates the application.

BUSINESS LOGIC (JAVA CODE):

```
import java.sql.*;
import java.util.Scanner;

class User {
    protected String name;
    protected String id;

    public User(String name, String id) {
        this.name = name;
        this.id = id;
    }
}

class Admin {
    protected String name;
    protected String id;

    public Admin(String name, String id) {
        this.name = name;
        this.id = id;
    }

    public GatePass issueGatePass(User user, String purpose) {
        return new GatePass(user, purpose);
```

```
    }
}

class Employee extends User {
    public Employee(String name, String id) {
        super(name, id);
    }
}

class Student extends User {
    private String course;
    private int year;

    public Student(String name, String id, int year, String course) {
        super(name, id);
        this.course = course;
        this.year = year;
    }

    @Override
    public String toString() {
        return "Name: " + name + "\n" +
                "ID: " + id + "\n" +
                "Course: " + course + "\n" +
                "Year: " + year + "\n";
    }
}

class GatePass {
```

```
    private User user;
    private String purpose;

    public GatePass(User user, String purpose) {
        this.user = user;
        this.purpose = purpose;
    }

    @Override
    public String toString() {
        return "Gate Pass Details:\n" +
                "User: " + user.name + " (ID: " + user.id + ")\n" +
                "Purpose: " + purpose + "\n";
    }
}

public class GatePassApplication {
    private static Connection con;

    // Method to establish database connection
    private static void connectDB() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/project";
        String user = "root";
        String password = "Abhilesh13";
        con = DriverManager.getConnection(url, user, password);
    }

    // Method to retrieve user details from the database based on the user ID
    private static User getUserDetails(String userId, String type) throws SQLException {
```

```
        String query;
        if (type.equals("Student")) {
            query = "SELECT * FROM Student WHERE id=?";
        } else {
            query = "SELECT * FROM EMP WHERE id=?";
        }
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, userId);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            String name = resultSet.getString("name");
            String id = resultSet.getString("id");
            if (type.equals("Student")) {
                int year = resultSet.getInt("year");
                String course = resultSet.getString("course");
                return new Student(name, id, year, course);
            }
            return new Employee(name, id);
        } else {
            return null;
        }
    }

    private static void addStudentToDB(String id, String name, int year, String course) throws
SQLException {
        String query = "INSERT INTO Student (id, name, year, course) VALUES (?, ?, ?, ?)";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, id);
        preparedStatement.setString(2, name);
```

```
    preparedStatement.setInt(3, year);
    preparedStatement.setString(4, course);
    preparedStatement.executeUpdate();
    System.out.println("User details added to the database.");
}

private static void addEmpToDB(String id, String name) throws SQLException {
    String query = "INSERT INTO Emp (id, name) VALUES (?, ?)";
    PreparedStatement preparedStatement = con.prepareStatement(query);
    preparedStatement.setString(1, id);
    preparedStatement.setString(2, name);
    preparedStatement.executeUpdate();
    System.out.println("User details added to the database.");
}

private static void deleteStudent(String id) throws SQLException {
    String query = "DELETE FROM Student WHERE id=?";
    PreparedStatement preparedStatement = con.prepareStatement(query);
    preparedStatement.setString(1, id);
    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Student with ID " + id + " deleted from the database.");
    } else {
        System.out.println("Student with ID " + id + " not found in the database.");
    }
}

private static void deleteEmp(String id) throws SQLException {
    String query = "DELETE FROM Emp WHERE id=?";
```

```
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, id);
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Employee with ID " + id + " deleted from the database.");
        } else {
            System.out.println("Employee with ID " + id + " not found in the database.");
        }
    }

    private static void updateStudentCourse(String id, String newCourse) throws
SQLException {
        String query = "UPDATE Student SET course=? WHERE id=?";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, newCourse);
        preparedStatement.setString(2, id);
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Student with ID " + id + " course updated in the database.");
        } else {
            System.out.println("Student with ID " + id + " not found in the database.");
        }
    }

    private static void updateEmpName(String id, String newName) throws SQLException {
        String query = "UPDATE Emp SET name=? WHERE id=?";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, newName);
        preparedStatement.setString(2, id);
        int rowsAffected = preparedStatement.executeUpdate();
```

```
    if (rowsAffected > 0) {
        System.out.println("Employee with ID " + id + " name updated in the database.");
    } else {
        System.out.println("Employee with ID " + id + " not found in the database.");
    }
}

public static void main(String[] args) throws SQLException {
    try {
        connectDB(); // Establish database connection
    } catch (SQLException e) {
        System.out.println("Failed to connect to the database. Exiting...");
        return;
    }

    Scanner scanner = new Scanner(System.in);
    //System.out.println("Welcome to GatePass Application!");
    while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Issue Gate Pass");
        System.out.println("2. Create a new Student");
        System.out.println("3. Update Student Course");
        System.out.println("4. Create a new Employee");
        System.out.println("5. Update Employee Name");
        System.out.println("6. Delete Student");
        System.out.println("7. Delete Employee");
        System.out.println("0. Exit");

        System.out.println("Enter your choice:");
```

```
        int choice = scanner.nextInt();
        scanner.nextLine(); // Clear the input buffer

        switch (choice) {
            case 1:
            System.out.println("\nEnter your ID:");
            String id = scanner.next();
            System.out.println("\nEnter your role:\n1. Student\n2. Employee\n");
            int role = scanner.nextInt();

            User user;
            try {
                user = getUserDetails(id, (role == 1) ? "Student" : "Emp"); // Corrected the table name for Employee
            } catch (SQLException e) {
                System.out.println("Error fetching user details. Please try again.");
                e.printStackTrace();
                continue;
            }

            if (user == null) {
                System.out.println("Invalid user ID. Try again.");
                continue;
            }

            scanner.nextLine(); // Clear the input buffer
            System.out.println("Enter the purpose of the gate pass:");
            String purpose = scanner.nextLine();

            Admin admin = new Admin("Admin Name", "1");
```

```
GatePass gatePass = admin.issueGatePass(user, purpose);
System.out.println("\nGate pass issued successfully:");
System.out.println(gatePass);

if (user instanceof Student) {
    System.out.println("Student Details:");
    System.out.println(user);
}
break;



case 2:
    System.out.println("\nEnter Student ID:");
    String studentId = scanner.next();
    System.out.println("Enter Student Name:");
    String studentName = scanner.next();
    System.out.println("Enter Student Year:");
    int studentYear = scanner.nextInt();
    scanner.nextLine(); // Clear the input buffer
    System.out.println("Enter Student Course:");
    String studentCourse = scanner.nextLine();

    addStudentToDB(studentId, studentName, studentYear, studentCourse);
    break;

case 3:
    System.out.println("\nEnter Student ID:");
    String studentIdToUpdate = scanner.next();
    scanner.nextLine(); // Clear the input buffer
```

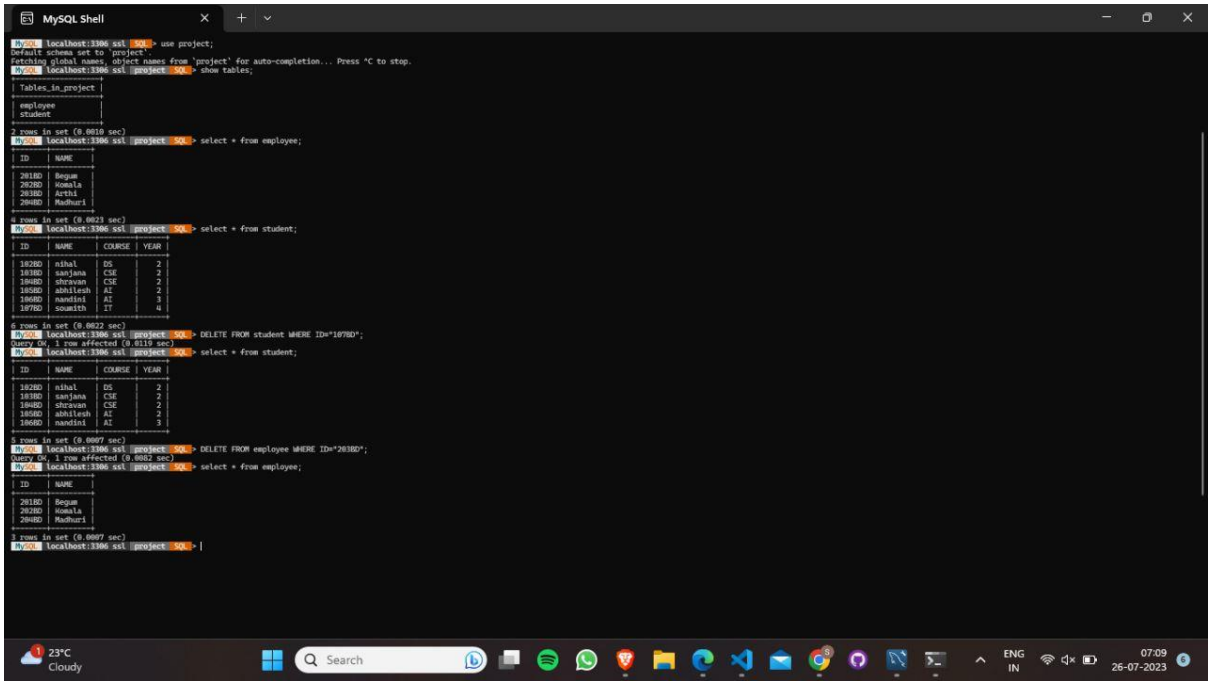
```
        System.out.println("Enter New Student Course:");
        String newStudentCourse = scanner.nextLine();

        updateStudentCourse(studentIdToUpdate, newStudentCourse);
        break;

    case 4:
        System.out.println("\nEnter Employee ID:");
        String empId = scanner.next();
        scanner.nextLine(); // Clear the input buffer
        System.out.println("Enter Employee Name:");
        String empName = scanner.nextLine();

        addEmpToDB(empId, empName);
        break;

    case 5:
        System.out.println("\nEnter Employee ID:");
        String empIdToUpdate = scanner.next();
        scanner.nextLine(); // Clear the input buffer
        System.out.println("Enter New Employee Name:");
        String newEmpName = scanner.nextLine();

        updateEmpName(empIdToUpdate, newEmpName);
        break;

    case 6:
        System.out.println("\nEnter Student ID to delete:");
        String studentIdToDelete = scanner.next();
```

```
            deleteStudent(studentIdToDelete);
            break;

        case 7:
            System.out.println("\nEnter Employee ID to delete:");
            String empIdToDelete = scanner.next();

            deleteEmp(empIdToDelete);
            break;

        case 0:
            // Exit the program
            System.out.println("Exiting...");
            try {
                if (con != null) {
                    con.close(); // Close the database connection
                }
            } catch (SQLException e) {
                System.out.println("Error closing the database connection.");
            }
            scanner.close();
            return;

        default:
            System.out.println("Invalid choice. Please try again.");
      }
   }
}
```

JAVA PROJECT – GATE PASS APPLICATION

}

BACK END:



JAVA PROJECT – GATE PASS APPLICATION

OUTPUT:

Menu:

1. Issue Gate Pass

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

1

Enter your ID:

101BD

Enter your role:

1. Student

2. Employee

1

Invalid user ID. Try again.

Menu:

1. Issue Gate Pass

JAVA PROJECT – GATE PASS APPLICATION

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

2

Enter Student ID:

101BD

Enter Student Name:

Sanjana

Enter Student Year:

2

Enter Student Course:

CSe

User details added to the database.

Menu:

1. Issue Gate Pass

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

3

JAVA PROJECT – GATE PASS APPLICATION

Enter Student ID:

101BD

Enter New Student Course:

ds

Student with ID 101BD course updated in the database.

Menu:

1. Issue Gate Pass

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

4

Enter Employee ID:

205BD

Enter Employee Name:

happy

User details added to the database.

Menu:

1. Issue Gate Pass

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

5

Enter Employee ID:

205BD

Enter New Employee Name:

begum

Employee with ID 205BD name updated in the database.

Menu:

1. Issue Gate Pass

2. Create a new Student

3. Update Student Course

4. Create a new Employee

5. Update Employee Name

6. Delete Student

7. Delete Employee

0. Exit

Enter your choice:

6

Enter Student ID to delete:

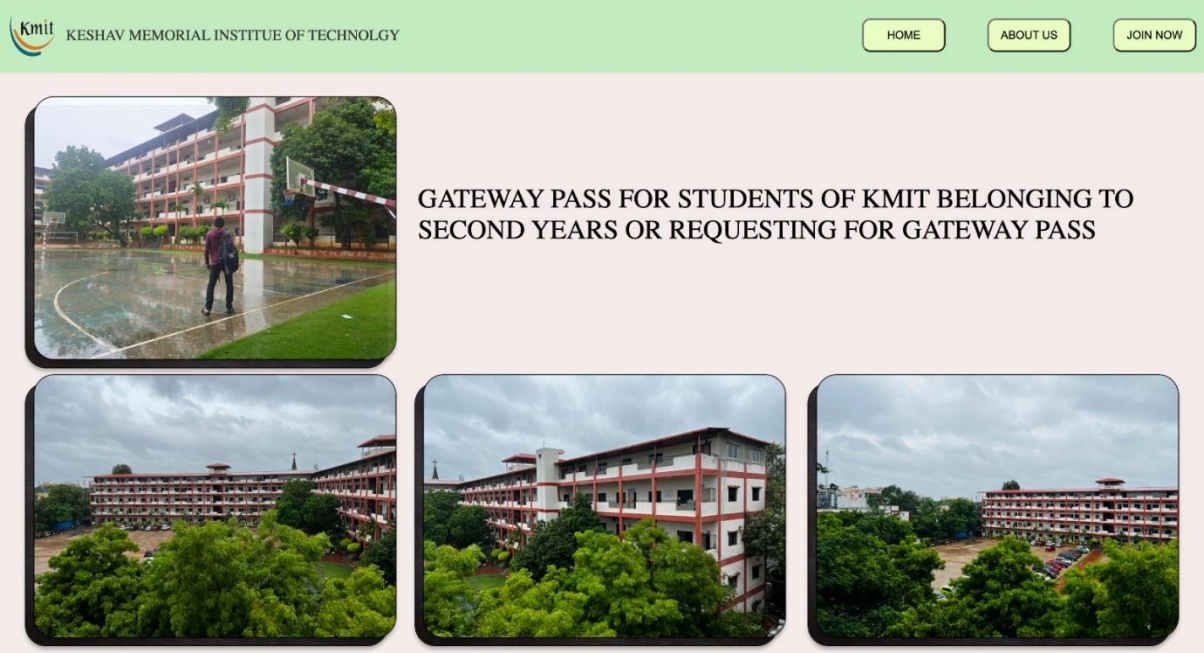
103BD

Student with ID 103BD deleted from the database.

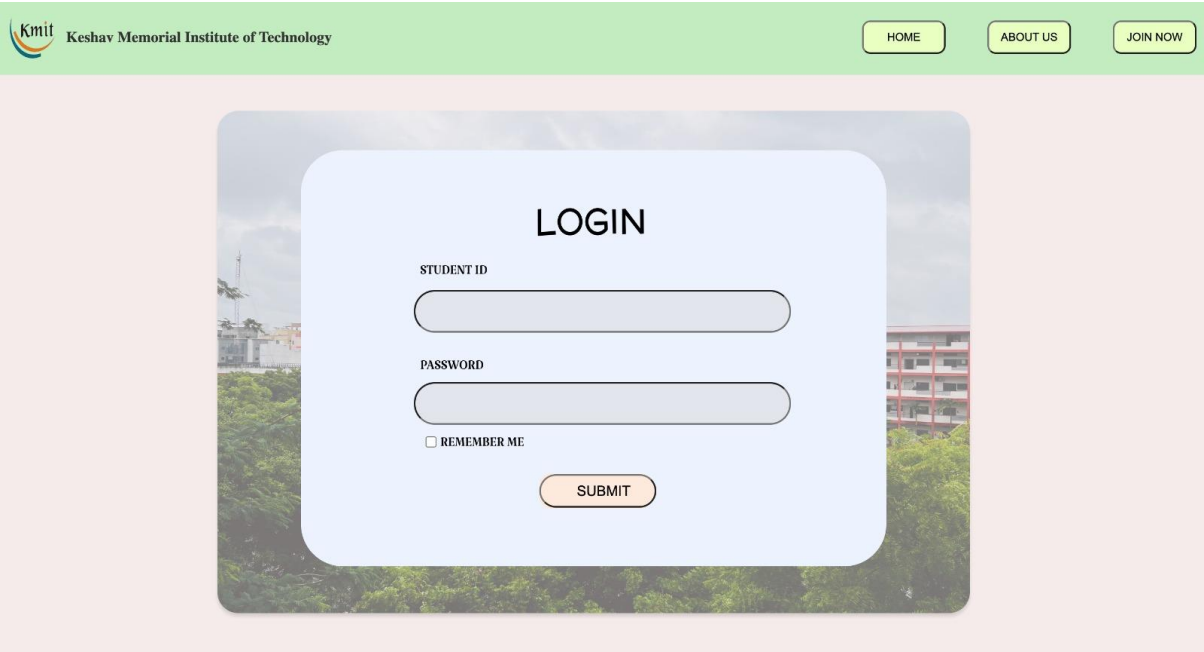
JAVA PROJECT – GATE PASS APPLICATION

FRONT END:

1.Home Page

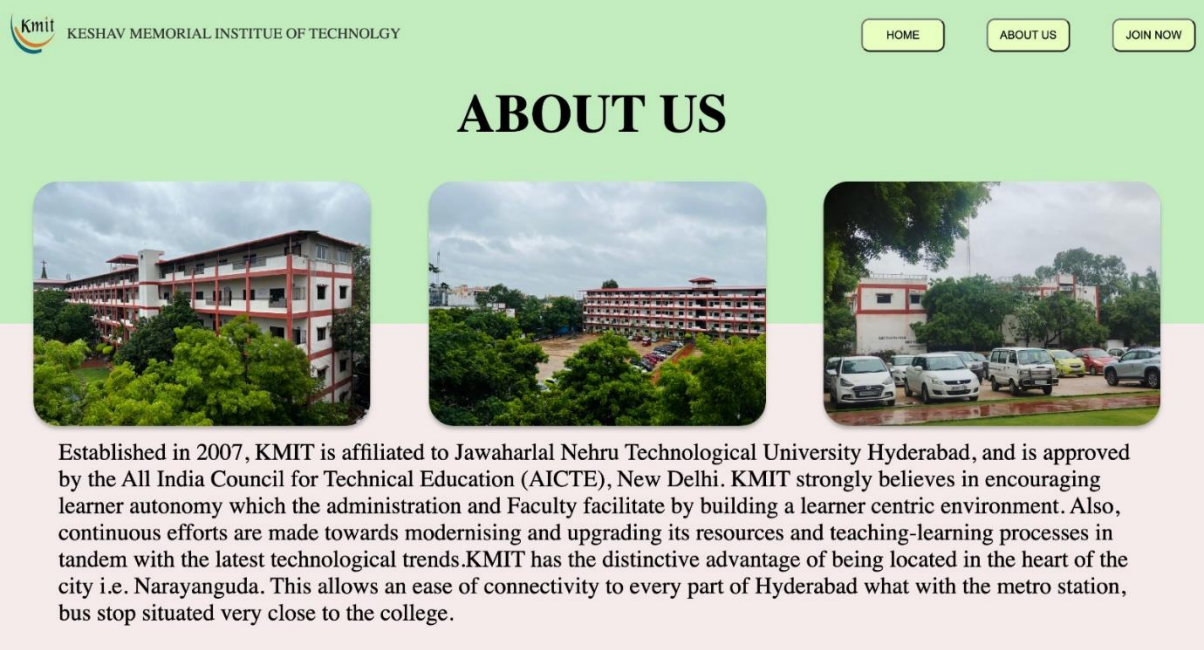


2.Login



JAVA PROJECT – GATE PASS APPLICATION

3.About us:



CONCLUSION-

The main purpose of our project is to provide a facility to both students and employees in KMIT to issue a gate pass through the admin, it is a well implemented gate pass application and helps in maintain an organized environment within our organisation. It is user friendly and is easier to use.

- THANK YOU

JAVA PROJECT – GATE PASS APPLICATION