**TASK-1**

**CODE**

import java.util.ArrayList;

import java.util.Scanner;

class Employee {

private String id;

private String name;

private String position;

public Employee(String id, String name, String position) {

this.id = id;

this.name = name;

this.position = position;

}

public String getId() {

return id;

}

public String getName() {

return name;

}

public String getPosition() {

return position;

}

@Override

public String toString() {

return "Employee{" +

"id='" + id + '\'' +

", name='" + name + '\'' +

", position='" + position + '\'' +

'}';

}

}

class EmployeeManagementSystem {

private ArrayList<Employee> employees;

public EmployeeManagementSystem() {

this.employees = new ArrayList<>();

}

public void addEmployee(Employee employee) {

employees.add(employee);

System.out.println("Employee added successfully.");

}

public void viewAllEmployees() {

if (employees.isEmpty()) {

System.out.println("No employees in the system.");

} else {

System.out.println("List of Employees:");

for (Employee employee : employees) {

System.out.println(employee);

}

}

}

public void deleteEmployee(String id) {

for (Employee employee : employees) {

if (employee.getId().equals(id)) {

employees.remove(employee);

System.out.println("Employee with ID " + id + " deleted successfully.");

return;

}

}

System.out.println("Employee with ID " + id + " not found.");

}

}

public class Main {

public static void main(String[] args) {

EmployeeManagementSystem system = new EmployeeManagementSystem();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Add Employee");

System.out.println("2. View All Employees");

System.out.println("3. Delete Employee");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter employee ID: ");

String id = scanner.nextLine();

System.out.print("Enter employee name: ");

String name = scanner.nextLine();

System.out.print("Enter employee position: ");

String position = scanner.nextLine();

Employee newEmployee = new Employee(id, name, position);

system.addEmployee(newEmployee);

break;

case 2:

system.viewAllEmployees();

break;

case 3:

System.out.print("Enter employee ID to delete: ");

String deleteId = scanner.nextLine();

system.deleteEmployee(deleteId);

break;

case 4:

System.out.println("Exiting the Employee Management Application. Goodbye!");

System.exit(0);

default:

System.out.println("Invalid choice. Please enter a valid option.");

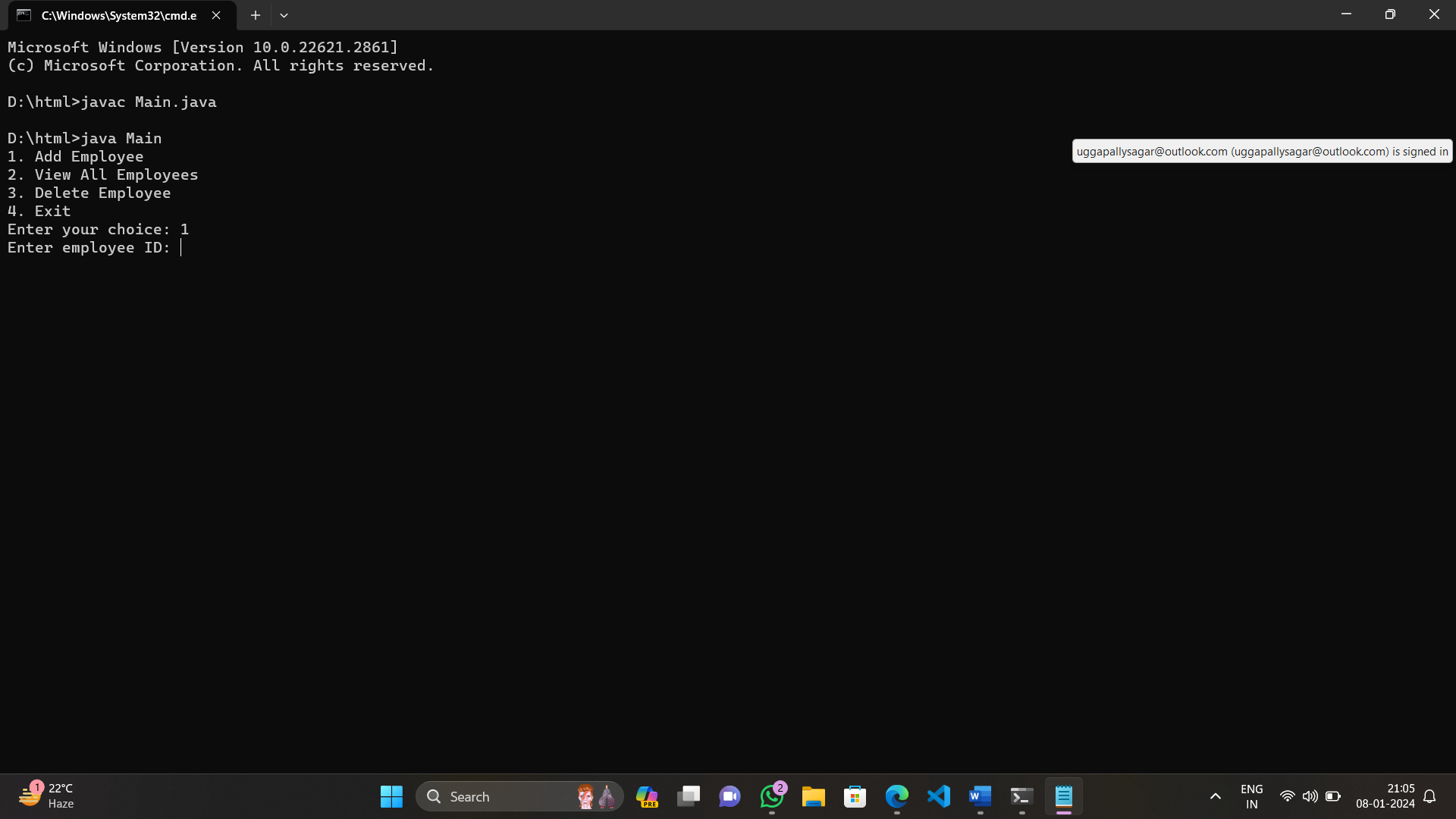
}

}

}

}

**OUT PUT:**



**EXPLANATION:**

The Employee Management Application is a Java-based software designed to streamline the process of managing employee information within an organization. This application serves as a centralized platform for HR departments and managers to perform various tasks related to employee records efficiently. The primary functionalities of the application include adding new employee records, viewing existing records, updating information, and deleting records when necessary.

Key features of the Employee Management Application:

1. **User-Friendly Interface:**
   * The application provides a user-friendly interface, making it easy for HR personnel and managers to navigate through the system.
2. **Employee Record Creation:**
   * HR administrators can easily add new employee records to the system. This includes entering essential details such as personal information, contact details, job title, department, and any other relevant data.
3. **Record Viewing:**
   * The application allows authorized users to view comprehensive details of existing employee records. This feature is essential for quickly accessing information about specific employees.
4. **Information Update:**
   * HR managers can update employee information as needed. This includes modifying personal details, job-related information, or any other relevant data. This ensures that the system always reflects the latest and most accurate employee information.
5. **Record Deletion:**
   * In cases where an employee leaves the organization or if duplicate records exist, authorized users can delete records to maintain data integrity.
6. **Security and Access Control:**
   * The application incorporates security measures to protect sensitive employee information. Access control mechanisms are implemented to ensure that only authorized personnel can perform certain actions, such as adding, updating, or deleting records.
7. **Search and Filtering:**
   * The application provides search and filtering functionalities, enabling users to quickly locate specific employee records based on various criteria. This feature is particularly useful in large organizations with a significant number of employees.
8. **Audit Trail:**
   * The application may include an audit trail feature that logs all changes made to employee records. This ensures transparency and accountability by tracking who made specific modifications and when.
9. **Data Export:**
   * The ability to export employee data in various formats (e.g., CSV, Excel) facilitates reporting and integration with other systems.

---------------------------------------------------------------------END--------------------------------------------------