Diversity and Inclusion audit tool

1. Company Profile – GUVI HCL

GUVI HCL is a collaborative initiative between GUVI (Grab Ur Vernacular Imprint) and HCL Technologies, designed to bridge the gap between academic learning and industry skills through hands-on technical training and real-world exposure.

GUVI, an edtech platform incubated by IIT Madras and IIM Ahmedabad, was founded in 2014 with the mission of making technology education accessible to everyone in vernacular languages such as Tamil, Telugu, Hindi, and Kannada. Headquartered in Chennai, India, GUVI has empowered over 10 lakh learners through online courses, coding bootcamps, and career programs. The platform specializes in programming, full-stack development, artificial intelligence, cloud computing, and data science, offering training aligned with current IT industry needs.

HCL Technologies, a global technology leader with decades of experience in IT services and consulting, partners with GUVI to offer industry-relevant programs like the GUVI–HCL Tech Career Program and HCL Career Launchpad. Through this collaboration, students gain exposure to enterprise-level technologies, mentorship from HCL professionals, and opportunities to work on real-time industrial projects.

The GUVI–HCL partnership focuses on transforming aspiring students into skilled and job-ready IT professionals by integrating theoretical learning with practical implementation. Together, they aim to create a new generation of tech talent that is proficient, confident, and ready to contribute to India’s fast-growing digital and innovation ecosystem.

SRI VENKATESWARA COLLEGEOFENGINEERINGANDTECHNOLOGY

(AUTONOMOUS)

R.V.S.Nagar, Chittoor–517127.(A.P)

(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapur)

(Accredited by NBA, New Delhi C NAAC, Bangalore)

(An ISO 9001:2000 Certified Institution)

2025-2026

This is to certify that the project report submitted by Balu Mohan sai, Regd. No.: 22781A0505, is the original work carried out by him during the Academic Year 2025–2026 as part of the Java Training Program conducted by HCL GUVI.

P.Ragavan

Technical Trainer

P.Jyotheeswari

Head of the department

(COMPUTER SCIENCE & ENGINEERING)

Abstract

The Diversity and Inclusion Audit Tool is a Java-based application designed to help organizations monitor, evaluate, and enhance workplace diversity and inclusivity. The system integrates both in-memory data management using a Doubly Linked List (DLL) and persistent storage through MongoDB, ensuring e icient handling and long-term storage of employee and audit records.

The tool allows users to add, update, search, and delete employee profiles, as well as record and review audit entries that assess inclusion metrics such as gender representation, disability status, ethnicity distribution, and inclusion scores. It automatically computes key diversity and inclusion (D&I) metrics, providing valuable insights into the organization's demographic balance and inclusivity progress.

By combining structured data management, database integration, and analytical reporting, this project demonstrates how software systems can support equitable and inclusive organizational practices through data-driven auditing and decision-making.

Index

S.NO	Content	Page No
1	Company profile	1
2	certificate	2
3	Abstract	3
4	Index	4
5	Aim	5
6	Algorithm	5-9
7	System requirements	10
8	Program code	11-27
9	Output screenshot/output	28
10	conclusion	29

Aim:

The aim of the Diversity and Inclusion Audit Tool is to design and implement a Java-based system that e iciently manages and audits employee diversity data to promote inclusivity within an organization. The tool allows storing, updating, and analyzing employee demographic and inclusion information, while maintaining records in both in-memory data structures (Doubly Linked List) and a persistent MongoDB database.

It further enables generating audit records, computing diversity metrics such as gender distribution, disability representation, ethnicity breakdown, and average inclusion scores — helping organizations monitor and improve their diversity and inclusion practices.

Algorithm:

1.Start

1. Initialize the program.
2. Connect to MongoDB using connection URI (mongodb://localhost:27017).
3. Select database: diversity_audit_db.
4. Create or access two collections:
    - employees → to store employee data.
    - audits → to store audit records.

---

2. Define Data Structures

1. EmployeeNode (for Doubly Linked List)
    - Fields:
      id, name, age, gender, ethnicity, hasDisability, prev, next.
2. EmployeeDLL

- Fields:
  head, tail, empCollection, auditCollection.

---

3. Load Existing Employees

1. Retrieve all documents from employees collection.
2. For each document:
   - Extract details (ID, name, age, gender, ethnicity, disability).
   - Create an EmployeeNode.
   - Insert the node at the end of the doubly linked list.

---

4. Display Main Menu

Repeatedly display menu options until the user selects “Exit”:

1. Add Employee
2. Display Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Add Audit Record
7. Display Audits for Employee
8. Compute D&I Metrics
9. Exit

---

5. Menu Option Functionalities

(1) Add Employee

1. Input employee details: ID, name, age, gender, ethnicity, disability status.
2. Check if ID already exists in the DLL.
    - If exists → print "Employee already exists".
    - Else:
        - Create a new node.
        - Append to the end of DLL.
        - Insert into employees MongoDB collection.

(2) Display Employees

1. Traverse DLL from head to tail.
2. Print each employee's details in table format.

(3) Search Employee

1. Input employee ID.
2. Traverse DLL until match is found.
    - If found → display employee info.
    - Else → show "Employee not found".

(4) Update Employee

1. Input employee ID.
2. If found:
    - Prompt for new (or same) values for each field.
    - Update node in DLL.
    - Update corresponding record in MongoDB.
3. If not found → show "Employee not found".

(5) Delete Employee

1. Input employee ID.
2. If found:

- Remove node from DLL.
- Delete employee document from MongoDB.
- Delete all related audit records from the audits collection.

3. Else → print “Employee not found”.

(6) Add Audit Record

1. Input employee ID, auditor name, date, inclusion score, and notes.
2. Validate that employee exists in DLL.
3. If found:
    - Create new audit document.
    - Include employee details (gender, ethnicity, disability) for easy reporting.
    - Insert record into audits collection.
4. Else → print “Employee not found”.

(7) Display Audits for Employee

1. Input employee ID.
2. Retrieve all audit documents with matching ID from MongoDB.
3. Display in chronological order by date.
4. If none exist → print “No audit records found”.

(8) Compute D&I Metrics

Compute and display key diversity statistics:

1. Employee Demographics (from DLL):
    - Total employees. o Count and % by gender (male, female, non-binary, other).
    - Count and % of employees with disabilities.
2. Inclusion Scores (from MongoDB):

- Fetch all inclusionScore fields.
- Compute average inclusion score and number of records.

3. Ethnicity Breakdown (from DLL): o Count frequency of each ethnicity.
   o Display each with % of total.

(9) Exit

 Close MongoDB connection and terminate the program.

---

6. Helper Functions

1. readIntSafe() → Safely read integer input from user.
2. readIntSafe(min, max) → Read integer within a specific range.
3. readYesNo() → Read and validate yes/no responses.
4. safeParseInt() → Safely convert MongoDB field to integer.

---

7. End Program

1. Display “Exiting…” message.
2. Close Scanner and MongoDB client.
3. Stop execution.

System Requirements:

The Diversity and inclusion audit tool is a Java-based application integrated with a MongoDB database. It requires both hardware and software components to function e iciently. The following are the minimum and recommended system requirements for developing and executing the project.

> Hardware Requirements:

Component	Minimum Requirement	Recommended Requirement
Processor	Intel Core i3 or equivalent	Intel Core i5 / i7 or higher
RAM	4 GB	8 GB or more
Hard Disk	500 MB free space	1 GB free space
Monitor	1024 × 768 resolution	1920 × 1080 resolution
Keyboard & Mouse	Standard input devices	Standard input devices

>Software Requirements:

Component	Specification
Operating System	Windows 10 / 11, Linux, or macOS
Programming Language	Java (JDK 8 or higher)
Database	MongoDB (version 4.0 or higher)
IDE / Editor	IntelliJ IDEA, Eclipse, or NetBeans
Build Tool (optional)	Apache Maven or Gradle
Driver Library	MongoDB Java Driver (mongodb-driver-sync)
Command Line Tool	MongoDB Compass / Command Prompt for testing
Additional Libraries	org.bson and com.mongodb.client packages

Source Code:

```
import com.mongodb.client.*; import
com.mongodb.client.model.Sorts;
import org.bson.Document;

import java.time.LocalDate; import
java.time.format.DateTimeFormatter;
import java.util.Scanner;

/**
* Diversity & Inclusion Audit Tool
* - Stores Employee records in a doubly linked list (in-memory) and persists to
  MongoDB.
* - Stores Audit records in a separate MongoDB collection (no in-memory DLL for
  audits to keep things simple).
* - Basic metrics: gender distribution, disability count, average inclusion score, ethnicity
  breakdown.
 *
* Replace MongoDB connection string if necessary.
 */
public class DiversityInclusionAudit {
```

```
   // Node for Employee DLL
static class EmployeeNode {
        String id;
        String name;
int age;
        String gender;      // e.g.,
Male/Female/Non-binary         String ethnicity;
// free text         boolean hasDisability;
        EmployeeNode prev, next;


        EmployeeNode(String id, String name, int age, String gender, String ethnicity,
boolean hasDisability) {
            this.id = id;
            this.name = name;
this.age = age;             this.gender =
gender;             this.ethnicity =
ethnicity;             this.hasDisability
= hasDisability;
        }
    }


    // DLL managing employees with
MongoDB     static class EmployeeDLL {
```

```
    EmployeeNode head, tail;
    MongoCollection<Document> empCollection;
    MongoCollection<Document> auditCollection;
    DateTimeFormatter dateFmt = DateTimeFormatter.ISO_DATE;

    EmployeeDLL(MongoCollection<Document> empCollection,
MongoCollection<Document> auditCollection) {        this.empCollection =
empCollection;        this.auditCollection = auditCollection;
    }

    // Load employees from MongoDB into
DLL      void loadFromDatabase() {
        FindIterable<Document> docs =
empCollection.find().sort(Sorts.ascending("id"));        for (Document doc :
docs) {
            String id = doc.getString("id");
            String name =
doc.getString("name");            int age =
safeParseInt(doc.get("age"));
            String gender = doc.getString("gender");
String ethnicity = doc.getString("ethnicity");            boolean
hasDisability = doc.getBoolean("hasDisability", false);
```

```
            EmployeeNode newNode = new EmployeeNode(id, name, age,
gender, ethnicity, hasDisability);                if (head == null) head = tail =
newNode;                else {
                tail.next =
newNode;
newNode.prev = tail;
tail = newNode;
            }
        }
    }

    private int safeParseInt(Object
obj) {            if (obj == null) return
0;
        if (obj instanceof Integer) return (Integer) obj;            try {
return Integer.parseInt(obj.toString()); } catch (Exception e) { return 0;
}
    }

    // Add employee (DLL + Mongo)        void addEmployee(String id, String
name, int age, String gender, String ethnicity, boolean hasDisability) {            if
(searchEmployee(id) != null) {
```

```
            System.out.println("Employee with this ID already
exists.");                return;
        }
        EmployeeNode newNode = new EmployeeNode(id, name, age,
gender, ethnicity, hasDisability);            if (head == null) head = tail =
newNode;            else {
            tail.next =
newNode;
newNode.prev = tail;
tail = newNode;
        }

        Document doc = new Document("id", id)
                .append("name", name)
                .append("age", age)
                .append("gender", gender)
                .append("ethnicity", ethnicity)
                .append("hasDisability", hasDisability);
empCollection.insertOne(doc);
        System.out.println("☑ Employee added.");
    }
```

```
    // Display all employees
from DLL     void
displayEmployees() {         if
(head == null) {
            System.out.println("No employees found.");
            return;
        }
        System.out.printf("%-10s %-20s %-5s %-12s %-15s %-10s%n", "ID",
"Name", "Age", "Gender", "Ethnicity", "Disability");         EmployeeNode
cur = head;         while (cur != null) {
            System.out.printf("%-10s %-20s %-5d %-12s %-15s %-10s%n",
cur.id, cur.name, cur.age, cur.gender, cur.ethnicity, cur.hasDisability ? "Yes" : "No");
cur = cur.next;
        }
    }

    // Search by ID
    EmployeeNode
searchEmployee(String id) {
EmployeeNode cur = head;         while
(cur != null) {             if
(cur.id.equals(id)) return cur;             cur
= cur.next;
```

```
        }
        return null;
    }

    // Update employee     void updateEmployee(String id, String newName,
int newAge, String newGender, String newEthnicity, boolean newDisability) {
        EmployeeNode node =
searchEmployee(id);         if (node == null)
{             System.out.println("Employee not
found.");             return;
        }
        node.name = newName;
node.age = newAge;
node.gender = newGender;
node.ethnicity = newEthnicity;
node.hasDisability = newDisability;

        empCollection.updateOne(new Document("id", id),
new Document("$set", new Document("name", newName)
                    .append("age", newAge)
                    .append("gender", newGender)
                    .append("ethnicity", newEthnicity)
                    .append("hasDisability", newDisability)));
```

```
            System.out.println("☑ Employee updated.");
        }

        // Delete employee & related audit
records         void deleteEmployee(String id) {
            EmployeeNode node = searchEmployee(id);
            if (node == null) {
                System.out.println("Employee not
found.");                 return;
            }
            if (node.prev != null) node.prev.next = node.next;
            if (node.next != null) node.next.prev =
node.prev;             if (node == head) head =
node.next;             if (node == tail) tail =
node.prev;

            empCollection.deleteOne(new Document("id", id));
auditCollection.deleteMany(new Document("employeeId", id));
            System.out.println("☑ Employee and their audit records deleted.");
        }

        // Add an audit record for an employee
```

```
    void addAuditRecord(String employeeId, String auditor, LocalDate date, int
inclusionScore, String notes) {
        EmployeeNode node =
searchEmployee(employeeId);          if (node == null)
{
            System.out.println("Employee not found. Cannot add
audit.");              return;
        }
        Document auditDoc = new Document("employeeId", employeeId)
                .append("auditor", auditor)
                .append("date", dateFmt.format(date))
                .append("inclusionScore", inclusionScore) // 0-100
                .append("notes", notes)
                .append("employeeGender", node.gender)
                .append("employeeEthnicity", node.ethnicity)
                .append("employeeHasDisability", node.hasDisability);
auditCollection.insertOne(auditDoc);          System.out.println("☑ Audit record
saved.");
    }

    // Display audits for a given employee
    void displayAuditsForEmployee(String employeeId) {
```

```
        FindIterable<Document> docs = auditCollection.find(new
Document("employeeId", employeeId)).sort(Sorts.ascending("date"));
boolean any = false;            for (Document d : docs) {               any = true;
            System.out.println("----");
            System.out.println("Date: " + d.getString("date"));
            System.out.println("Auditor: " + d.getString("auditor"));
            System.out.println("Inclusion Score: " + d.getInteger("inclusionScore", 0));
System.out.println("Notes: " + d.getString("notes"));
        }
        if (!any) System.out.println("No audit records for employee " + employeeId);
    }

    // Compute basic D&I metrics from employees collection and
audits        void computeMetrics() {
        // Employees stats (iterate
DLL)            int total = 0;
        int male = 0, female = 0, nonBinary = 0,
otherGender = 0;            int disabilityCount = 0;

        EmployeeNode cur =
head;           while (cur !=
null) {               total++;
```

```
            String g = cur.gender == null ? "" :
cur.gender.toLowerCase();               if (g.contains("male"))
male++;             else if (g.contains("female")) female++;
else if (g.contains("non")) nonBinary++;            else
otherGender++;
            if (cur.hasDisability)
disabilityCount++;              cur = cur.next;
        }


        System.out.println("=== Employee Diversity
Snapshot ===");          System.out.println("Total
employees: " + total);          if (total > 0) {
            System.out.printf("Male: %d (%.1f%%)%n", male, percent(male, total));
            System.out.printf("Female: %d (%.1f%%)%n", female, percent(female,
total));
            System.out.printf("Non-binary: %d (%.1f%%)%n", nonBinary,
percent(nonBinary, total));
            System.out.printf("Other/Unspecified: %d (%.1f%%)%n", otherGender,
percent(otherGender, total));
            System.out.printf("Employees with disability: %d (%.1f%%)%n",
disabilityCount, percent(disabilityCount, total));
        }
```

```
        // Inclusion score metrics from
audits          double sumScore = 0;
int countScore = 0;
FindIterable<Document> audits =
auditCollection.find();          for
(Document d : audits) {
            Object o =
d.get("inclusionScore");              if
(o != null) {                try {
                sumScore += Double.parseDouble(o.toString());
countScore++;
            } catch (Exception ignored) {}
        }
    }
    System.out.println("\n=== Inclusion Score Metrics (from audit
records) ===");          if (countScore == 0) {
        System.out.println("No audit scores available.");
    } else {
        double avg = sumScore / countScore;
        System.out.printf("Average Inclusion Score: %.2f (based on %d records)%n",
avg, countScore);
    }
```

```
        // Ethnicity breakdown (from employees)
        System.out.println("\n=== Ethnicity Breakdown ===");
        // simple counting by scanning employees and collecting
frequencies          java.util.Map<String, Integer> ethCount = new
java.util.HashMap<>();          cur = head;          while (cur !=
null) {
            String e = (cur.ethnicity == null || cur.ethnicity.isBlank()) ? "Unspecified" :
cur.ethnicity.trim();
            ethCount.put(e, ethCount.getOrDefault(e, 0)
+ 1);              cur = cur.next;
        }
        if (ethCount.isEmpty()) {
            System.out.println("No ethnicity data.");
        } else {
            for (var entry : ethCount.entrySet()) {
                System.out.printf("%s: %d (%.1f%%)%n", entry.getKey(),
entry.getValue(), percent(entry.getValue(), total));
            }
        }
    }

    private double percent(int part, int
total) {          if (total == 0) return 0.0;
return (100.0 * part) / total;
```

```
        }
    }

    // Console menu     public static
void main(String[] args) {
        // MongoDB setup - change URI if needed
        MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
        MongoDatabase database = mongoClient.getDatabase("diversity_audit_db");
        MongoCollection<Document> empCollection =
database.getCollection("employees");
        MongoCollection<Document> auditCollection = database.getCollection("audits");
EmployeeDLL employees = new EmployeeDLL(empCollection, auditCollection);
employees.loadFromDatabase(); // populate DLL

        Scanner sc = new
Scanner(System.in);         int choice = -1;
        DateTimeFormatter dateFmt = DateTimeFormatter.ISO_DATE;

        menuLoop:
        while (true) {
            System.out.println("\n--- Diversity & Inclusion Audit Tool ---");
            System.out.println("1. Add Employee");
            System.out.println("2. Display Employees");
            System.out.println("3. Search Employee");
```

```
            System.out.println("4. Update Employee");
            System.out.println("5. Delete Employee");
            System.out.println("6. Add Audit Record");
            System.out.println("7. Display Audits for Employee");
            System.out.println("8. Compute D&I Metrics");
            System.out.println("9. Exit");
            System.out.print("Enter choice: ");

            if (!sc.hasNextInt()) {
               System.out.print("Invalid input. Enter a
number: ");              sc.next();              continue;
            }
            choice = sc.nextInt();
sc.nextLine();

            switch (choice) {
case 1 -> {
                  System.out.print("Enter ID: ");
                  String id = sc.nextLine().trim();
                  System.out.print("Enter Name: ");
                  String name =
sc.nextLine().trim();
```

```
System.out.print("Enter Age: ");
int age = readIntSafe(sc);
                System.out.print("Enter Gender (Male/Female/Non-binary/Other): ");
                String gender = sc.nextLine().trim();
                System.out.print("Enter Ethnicity: ");
                String ethnicity = sc.nextLine().trim();
System.out.print("Has disability? (yes/no): ");                    boolean
hasDisability = readYesNo(sc);                    employees.addEmployee(id,
name, age, gender, ethnicity, hasDisability);
            }
            case 2 ->
employees.displayEmployees();                case 3
-> {
                System.out.print("Enter ID to search: ");
                String id = sc.nextLine().trim();
                EmployeeNode n =
employees.searchEmployee(id);                    if (n != null)
{
                    System.out.printf("Found - ID: %s, Name: %s, Age: %d, Gender: %s,
Ethnicity: %s,
Disability: %s%n",
                            n.id, n.name, n.age, n.gender, n.ethnicity, n.hasDisability ?
"Yes" : "No");                    } else System.out.println("Employee not found.");
```

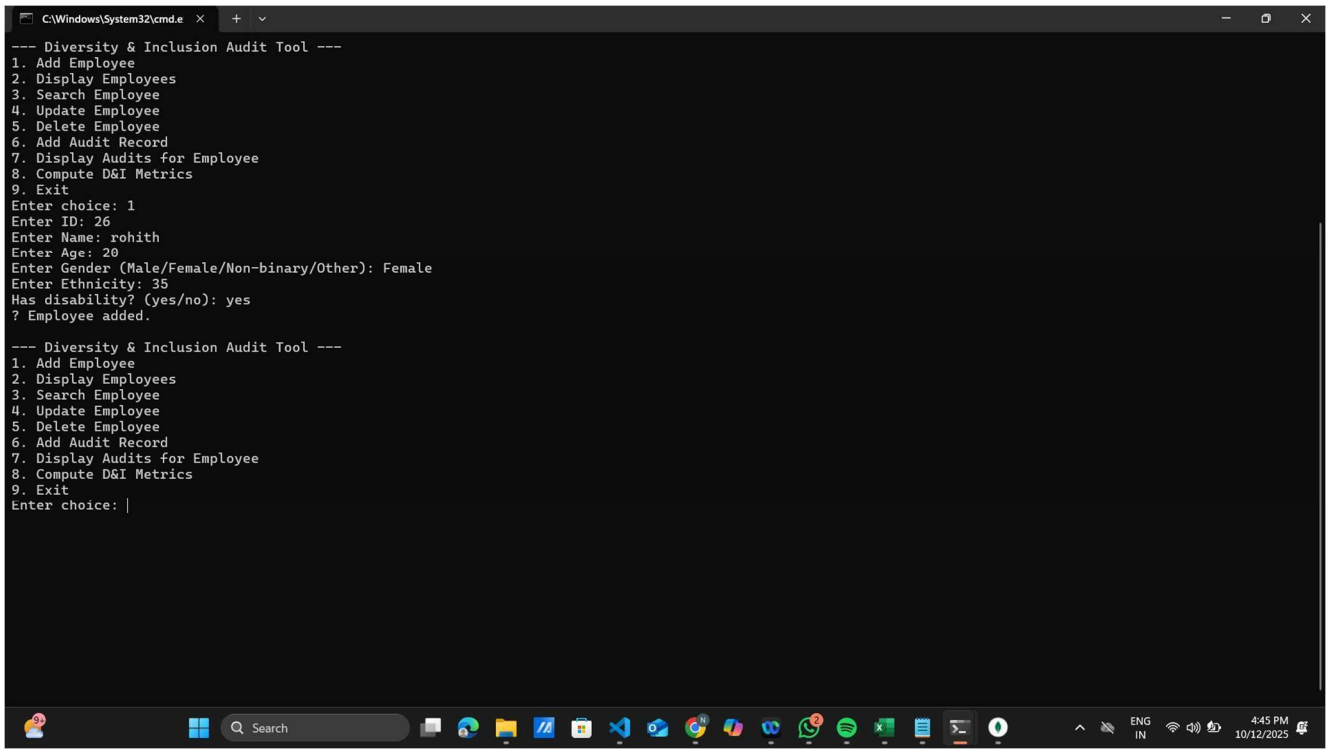
```
            }
            case 4 -> {
                System.out.print("Enter ID to update: ");
                String id = sc.nextLine().trim();
                EmployeeNode n = employees.searchEmployee(id);
                if (n == null) { System.out.println("Employee not found."); break; }
                System.out.print("Enter new name [" + n.name + "]: ");
                String name = sc.nextLine().trim(); if (name.isEmpty()) name = n.name;
                System.out.print("Enter new age [" + n.age + "]: ");
                String ageStr = sc.nextLine().trim(); int age = ageStr.isEmpty() ? n.age :
Integer.parseInt(ageStr);
                System.out.print("Enter new gender [" + n.gender + "]: ");
                String gender = sc.nextLine().trim(); if (gender.isEmpty()) gender =
n.gender;
                System.out.print("Enter new ethnicity [" + n.ethnicity + "]: ");
                String ethnicity = sc.nextLine().trim(); if (ethnicity.isEmpty()) ethnicity =
n.ethnicity;
                System.out.print("Has disability? (yes/no) [" + (n.hasDisability ? "yes" :
"no") + "]: ");
                String dis = sc.nextLine().trim(); boolean hasDisability = dis.isEmpty() ?
n.hasDisability : dis.equalsIgnoreCase("yes");
employees.updateEmployee(id, name, age, gender, ethnicity, hasDisability);
            }
            case 5 -> {
                System.out.print("Enter ID to delete: ");
```

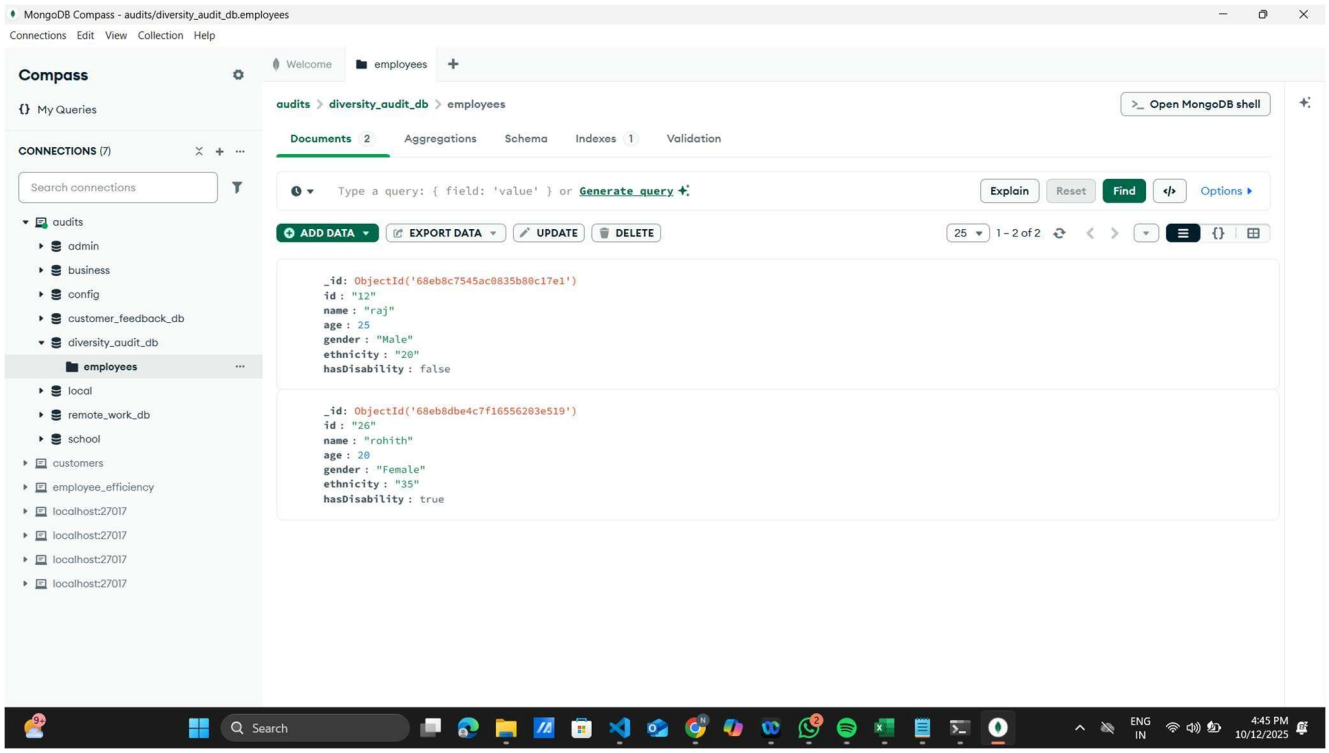
```
                String id = sc.nextLine().trim();
                employees.deleteEmployee(id);
            }
            case 6 -> {
                System.out.print("Enter Employee ID for audit: ");
                String id = sc.nextLine().trim();
                System.out.print("Enter Auditor name: ");
                String auditor = sc.nextLine().trim();
                System.out.print("Enter date (YYYY-MM-DD) or leave blank for today: ");
                String dd = sc.nextLine().trim();
                LocalDate date = dd.isEmpty() ? LocalDate.now() :
LocalDate.parse(dd, dateFmt);                System.out.print("Enter inclusion score
(0-100): ");                int score = readIntSafe(sc, 0, 100);
System.out.print("Enter notes: ");                String notes = sc.nextLine().trim();
employees.addAuditRecord(id, auditor, date, score, notes);
            }
            case 7 -> {
                System.out.print("Enter Employee ID to view
audits: ");                String id = sc.nextLine().trim();
employees.displayAuditsForEmployee(id);
            }
```

```
            case 8 ->
employees.computeMetrics();             case 9 -
> {

System.out.println("Exiting...");
break menuLoop;
            }
            default -> System.out.println("Invalid choice.");
        }
    }

    // Close resources
mongoClient.close();
sc.close();
  }

  // Helpers          private static int
readIntSafe(Scanner sc) {          while
(!sc.hasNextInt()) {
        System.out.print("Enter a valid integer: ");
sc.next();
    }
  int val = sc.nextInt();      sc.nextLine();      return val;
```

```
    }

    private static int readIntSafe(Scanner sc, int min, int
max)
{
        int val = readIntSafe(sc);         while (val < min ||
val > max) {
            System.out.printf("Enter a value between %d and %d: ", min,
max);             val = readIntSafe(sc);
        }
        return val;
    }

    private static boolean readYesNo(Scanner sc) {         String resp
= sc.nextLine().trim();         while (!(resp.equalsIgnoreCase("yes") ||
resp.equalsIgnoreCase("no") || resp.equalsIgnoreCase("y") ||
resp.equalsIgnoreCase("n"))) {             System.out.print("Please
answer yes or no: ");             resp = sc.nextLine().trim();
        }
        return resp.equalsIgnoreCase("yes") || resp.equalsIgnoreCase("y");
    }
}
```

Output :


C:\Windows\System32\cmd.e
--- Diversity & Inclusion Audit Tool ---
1. Add Employee
2. Display Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Add Audit Record
7. Display Audits for Employee
8. Compute D&I Metrics
9. Exit
Enter choice: 1
Enter ID: 26
Enter Name: rohith
Enter Age: 20
Enter Gender (Male/Female/Non-binary/Other): Female
Enter Ethnicity: 35
Has disability? (yes/no): yes
? Employee added.
--- Diversity & Inclusion Audit Tool ---
1. Add Employee
2. Display Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Add Audit Record
7. Display Audits for Employee
8. Compute D&I Metrics
9. Exit
Enter choice:
Search
ENG
IN
4:45 PM
10/12/2025


MongoDB Compass - audits/diversity_audit_db.employees
Connections Edit View Collection Help
Compass
My Queries
CONNECTIONS (7)
Search connections
audits
admin
business
config
customer_feedback_db
diversity_audit_db
employees
local
remote_work_db
school
customers
employee_efficiency
localhost:27017
localhost:27017
localhost:27017
localhost:27017
Welcome
employees
audits > diversity_audit_db > employees
Open MongoDB shell
Documents 2
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 2 of 2
_id: ObjectId('68eb8c7545ac0835b80c17e1')
id : "12"
name : "raj"
age : 25
gender : "Male"
ethnicity : "20"
hasDisability : false
_id: ObjectId('68eb8dbe4c7f16556203e519')
id : "26"
name : "rohith"
age : 20
gender : "Female"
ethnicity : "35"
hasDisability : true
Search
ENG
IN
4:45 PM
10/12/2025

Conclusion:

The Diversity and Inclusion Audit Tool successfully demonstrates how technology can be leveraged to promote equity and inclusivity within an organization. By combining data structures such as a Doubly Linked List for in-memory management and MongoDB for persistent storage, the system ensures e icient handling and retrieval of employee and audit data.

The tool not only simplifies employee record management but also provides meaningful insights through the computation of diversity and inclusion metrics, such as gender balance, disability representation, ethnicity breakdown, and average inclusion scores. These analytics enable organizations to make informed decisions and identify areas for improvement in their inclusion policies.

Overall, this project highlights the importance of integrating data-driven auditing into workplace practices, paving the way for a more inclusive and transparent work environment. Future enhancements could include a graphical user interface (GUI), automated report generation, and real-time dashboards for deeper analytical insights.