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 ed-tech 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 26 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, partnered 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. (Batch: 2022-2026)

Sri Venkateswara College of Engineering and Technology (Autonomous), Chittoor

Approved by AICTE, Permanently affiliated to JNTU, ANANTHAPURAMU
RVS NAGAR CHITTOOR, ANDHRA PRADESH
(An Autonomous Institution)
CHITTOOR

PROJECT REPORT

A report submitted in partial fulfilment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

DEPARTMENT OF

“COMPUTER SCIENCE AND ENGINEERING”

BY

H N BHARATH KUMAR

REGD_NO: 22781A0545

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, Permanently affiliated to JNTU, ANANTHAPURAMU

RVS NAGAR CHITTOOR, ANDHRA PRADESH

(An Autonomous Institution)

CHITTOOR

CERTIFICATION

This is to certify that, this is a Bonafide record of the project work on "**JAVA DEVELOPMENT**" submitted by **H N BHARATH KUMAR (Regd. No.: 22781A0545)** is work done by him and submitted during 2025 – 2026 Academic year, in partial fulfilment of the requirements for the award of the degree of **B. TECH** of **COMPUTER SCIENCE AND ENGINEERING,** at **GUVI HCL.**

GUVI HCL Technical Trainer

P. Ragavan

Head of the Department (HOD) of CSE

Dr. P. Jyotheeswari

GUVI | HCL

Skill Up. Level Up

Abstract

Development of a Employee Retention predictor with MongoDB Integration

Employee retention is one of the most critical challenges faced by organizations today. High employee turnover leads to increased recruitment costs and loss of productivity. The *Employee Retention Predictor* system is designed to help organizations analyze employee data and predict the likelihood of employees staying or leaving the company.

This project is developed in **Java** and uses **MongoDB** as the database for storing employee details such as age, years of experience, satisfaction level, and performance rating. The system performs basic **CRUD (Create, Read, Update, Delete)** operations on employee data through a simple command-line interface. A heuristic-based **retention prediction algorithm** is implemented to estimate each employee's retention probability using satisfaction, performance, and tenure factors.

MongoDB is connected directly using the official **MongoDB Java Driver**, allowing for flexible and scalable data handling without requiring Maven or additional frameworks. The project demonstrates practical integration between Java applications and NoSQL databases, showing how predictive analytics can be applied to human resource management systems.

Overall, the Employee Retention Predictor provides a foundation for building more advanced HR analytics tools that can support data-driven decision-making in employee management.

TABLE OF CONTENTS

S.NO	CONTENT	PAGE NO
1	AIM	6
2	ALGORITHM	7 - 9
3	System Requirements (Software & Hardware Requirements)	10 - 14
4	Project Code	15 - 19
5	Output Screenshots/COPY PASTE OUTPUT	20 - 23
6	Conclusion	24

AIM:

The main aim of the project **“Employee Retention Predictor using Java and MongoDB”** is to design and develop a system that can **store, manage, and analyze employee data** to **predict the likelihood of employees staying or leaving an organization**.

Key Focus Areas:

- ✅ **Database Integration** - Java + MongoDB connectivity
- ✅ **CRUD Operations** - Complete data management lifecycle
- ✅ **User Experience** - Intuitive interface and navigation
- ✅ **Data Integrity** - Error handling and validation
- ✅ **Educational Value** - Demonstrates software development principles

ALGORITHM FOR EMPLOYEE RETENTION PREDICTOR

1. Main Program Flow

```
1. START
2. Initialize employeeretentionpredictor object.
3. Display "MongoDB connection successful" message.
4. SET running = true.
5. WHILE (running == true) DO
   6.     CALL displayMenu().
   7.     READ user's choice.
   8.     SWITCH (choice) DO
   9.        CASE "1": CALL createEmployee().
   10.       CASE "2": CALL readAllEmployees().
   11.       CASE "3": CALL readEmployeeById().
   12.       CASE "4": CALL updateEmployee().
   13.       CASE "5": CALL deleteEmployee().
   14.       CASE "6": CALL predictRetention().
   15.       CASE "7": SET running = false.
   16.       DEFAULT: DISPLAY "Invalid option! Try again."
   17.    END SWITCH
6. END WHILE
7. CALL close() method.
8. STOP
```

2. Algorithm for Create Operation
Procedure: createEmployee()

```
1. DISPLAY "Enter Employee Details".
2. READ name, age, experience, satisfactionLevel, performanceScore.
3. CREATE a new document emp = { name, age, experience, satisfactionLevel, performanceScore }.
4. INSERT emp into MongoDB collection using insertOne().
5. DISPLAY "Employee record added successfully."
6. RETURN to main menu.
```

3. Algorithm for Read All Employees
Procedure: readAllEmployees()

```
1. FETCH all employee records from MongoDB collection using find().
2. IF no records found
       DISPLAY "No employee data available."
       RETURN.
```

```
3. FOR EACH document in fetched results
       DISPLAY employee ID, name, and key details.
4. END FOR
5. RETURN to main menu.
```

4. Algorithm for Read by ID

Procedure: readEmployeeById()

```
1. DISPLAY "Enter Employee ID to view details".
2. READ empId.
3. CONVERT empId to MongoDB ObjectId.
4. SEARCH MongoDB using find(Filters.eq("_id", empId)).
5. IF employee found
       DISPLAY all employee details.
6. ELSE
       DISPLAY "Employee not found."
7. RETURN to main menu.
```

5. Algorithm for Update Operation

Procedure: updateEmployee()

```
1. DISPLAY "Enter Employee ID to update".
2. READ empId.
3. FETCH employee record by ID.
4. IF record not found
       DISPLAY "Employee not found."
       RETURN.
5. DISPLAY current values and ASK user for new values.
6. CREATE update document updateData = { updatedFields }.
7. EXECUTE updateOne(Filters.eq("_id", empId), new Document("$set", updateData)).
8. DISPLAY "Employee details updated successfully."
9. RETURN to main menu.
```

6. Algorithm for Delete Operation

Procedure: deleteEmployee()

```
1. DISPLAY "Enter Employee ID to delete".
2. READ empId.
3. EXECUTE deleteOne(Filters.eq("_id", empId)).
4. IF deletedCount > 0
       DISPLAY "Employee deleted successfully."
5. ELSE
       DISPLAY "Employee not found."
6. RETURN to main menu.
```

7. Algorithm for Retention Prediction

Procedure: predictRetention()

```
1. DISPLAY "Enter Employee ID for retention prediction."
2. READ empId.
3. FETCH employee record by ID.
4. IF record not found
       DISPLAY "Employee not found."
       RETURN.
5. EXTRACT:
       - satisfactionLevel
       - performanceScore
       - experience
6. COMPUTE Retention Score using heuristic formula:
       $retentionScore = (0.5 \times satisfactionLevel) + (0.3 \times performanceScore) + (0.2 \times experience)$
7. IF retentionScore >= 7.0
       SET status = "High Retention Probability".
8. ELSE IF retentionScore >= 4.0
       SET status = "Moderate Retention Probability".
9. ELSE
       SET status = "Low Retention Probability".
10. DISPLAY employee name, retention score, and status.
11. RETURN to main menu.
```

8. Algorithm for Closing Connection

Procedure: close()

```
1. CALL mongoClient.close().
2. DISPLAY "MongoDB connection closed."
3. EXIT program.
```

CRUD Operations Flow:

CREATE: Input Validation → Duplicate Check → Insert → Success Message

READ: Query → Check Results → Display → Return

UPDATE: Find → Display Current → Input Changes → Update → Confirm

DELETE: Find → Confirm → Delete → Success Message

SYSTEM REQUIREMENTS

Knowledge Base System with Java and MongoDB

SOFTWARE REQUIREMENTS

Operating System

- **Minimum:** Windows 10, macOS 10.14+, or Linux Ubuntu 18.04+
- **Recommended:** Windows 11, macOS 12+, or Linux Ubuntu 20.04+

Development Environment

- **Java Development Kit (JDK)**
  - **Version:** JDK 8 or higher
  - **Minimum:** JDK 8u191
  - **Recommended:** JDK 11 or JDK 17 LTS
  - **Vendor:** Oracle JDK or OpenJDK
- **Integrated Development Environment (IDE)**
  - Visual Studio Code (with Java Extension Pack)

Database System

- **MongoDB Community Server**
  - **Version:** 4.4 or higher
  - **Recommended:** MongoDB 5.0+ or 6.0+
  - **GUI Tool:** MongoDB Compass (optional but recommended)

Java Dependencies

- **MongoDB Java Driver**
  - **Version:** 4.0+ (compatible with MongoDB server)
  - **Specific:** mongodb-driver-sync-4.10.2.jar
  - **Additional JARs:**
    - mongodb-driver-core-4.10.2.jar
    - bson-4.10.2.jar

Development Tools

- **Build Tool:** None (manual compilation) or Maven 3.6+
- **Version Control:** Git 2.30+
- **Terminal/Command Line:**
  - Windows: Command Prompt or PowerShell
  - macOS: Terminal or iTerm2
  - Linux: Bash terminal

HARDWARE REQUIREMENTS

Minimum Hardware Configuration

- **Processor:** Intel Core i3 or AMD Ryzen 3 equivalent
- **RAM:** 4 GB DDR3
- **Storage:** 2 GB free space
- **Network:** Basic internet connection for MongoDB setup

Recommended Hardware Configuration

- **Processor:** Intel Core i5 or AMD Ryzen 5 equivalent (2.0 GHz+)
- **RAM:** 8 GB DDR4
- **Storage:** 5 GB free SSD space
- **Network:** Stable internet connection

Optimal Development Configuration

- **Processor:** Intel Core i7 or AMD Ryzen 7 equivalent (3.0 GHz+)
- **RAM:** 16 GB DDR4
- **Storage:** 10 GB free SSD space
- **Network:** High-speed internet for quick downloads

SOFTWARE INSTALLATION REQUIREMENTS

Java Development Kit Setup

```
bash
# Verify Installation
java -version
javac -version

# Expected Output:
# java version "11.0.15" 2022-04-19 LTS
# Java(TM) SE Runtime Environment 18.9 (build 11.0.15+8-LTS-149)
# Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.15+8-LTS-149, mixed mode)
```

MongoDB Installation

```
bash
# macOS (using Homebrew)
brew tap mongodb/brew
brew install mongodb-community

# Windows
# Download from https://www.mongodb.com/try/download/community

# Linux Ubuntu
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
echo "deb [ arch=amd64, arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
sudo apt-get update
sudo apt-get install -y mongodb-org
```

VS Code Extensions

- **Extension Pack for Java** (by Microsoft)
- **MongoDB for VS Code** (optional)
- **Project Manager for Java** (optional)

SYSTEM CONFIGURATION

Environment Variables

- **JAVA_HOME:** Set to JDK installation path
- **PATH:** Include JDK bin directory
- **MongoDB:** Data and log directory paths

Network Configuration

- **MongoDB Port:** 27017 (default)
- **Localhost Access:** Enabled
- **Firewall:** Allow MongoDB connections if firewall is active

Database Configuration

- **Database Name:** KnowledgeBaseDB
- **Collection Name:** articles
- **Storage Engine:** WiredTiger (MongoDB default)

DEPENDENCY MANAGEMENT

Manual JAR Management

Project Structure:

```
KnowledgeBaseSystem/
├── lib/
│   ├── mongodb-driver-sync-4.10.2.jar
│   ├── mongodb-driver-core-4.10.2.jar
│   └── bson-4.10.2.jar
├── src/
│   └── KnowledgeBaseApp.java
└── README.md
```

Compilation Commands

```
bash
# Compile
javac -cp "lib/*" src/KnowledgeBaseApp.java -d .

# Run
java -cp ".:lib/*" KnowledgeBaseApp
```

COMPATIBILITY MATRIX

Component	Minimum Version	Recommended Version	Tested Version
Java JDK	8	11/17 LTS	11.0.15
MongoDB	4.4	6.0	6.0.4
MongoDB Driver	4.0	4.10	4.10.2
OS - Windows	10	11	11 (22H2)
OS - macOS	10.14	12.0	13.0
OS - Linux	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04

PERFORMANCE REQUIREMENTS

Application Performance

- **Startup Time:** < 5 seconds
- **Database Connection:** < 2 seconds
- **CRUD Operations:** < 1 second each
- **Search Operations:** < 3 seconds for large datasets

Scalability

- **Maximum Articles:** 10,000+ articles
- **Concurrent Users:** 1 (console application)
- **Data Storage:** Efficient document storage in MongoDB

Reliability

- **Uptime:** Application should run without crashes
- **Data Integrity:** No data loss during operations
- **Error Recovery:** Graceful handling of connection issues

SECURITY REQUIREMENTS

Basic Security

- **Local Database:** No authentication required for local development
- **Input Validation:** Basic validation for user inputs
- **Error Messages:** Non-revealing error messages

Network Security

- **Localhost Only:** MongoDB runs on localhost by default
- **No External Access:** Database not exposed to network

TESTING REQUIREMENTS

Testing Environment

- **Unit Testing:** JUnit 5 (optional)
- **Integration Testing:** Manual testing with MongoDB
- **Browser:** Not applicable (console application)

Quality Assurance

- **Code Compilation:** Zero warnings
- **Functionality:** All CRUD operations working
- **User Experience:** Clear menu and prompts

MAINTENANCE REQUIREMENTS

Updates

- **Java Updates:** Quarterly security updates
- **MongoDB Updates:** As per MongoDB release cycle
- **Driver Updates:** When updating MongoDB version

Monitoring

- **Disk Space:** Monitor MongoDB data directory
- **Logs:** Check application and MongoDB logs
- **Performance:** Monitor response times

PROJECT CODE:

GITHUB LINK: https://github.com/hnbharathkumaryadav/HCL-PROJECT

```
import com.mongodb.client.*;
import com.mongodb.client.model.Filters;
import org.bson.Document;
import org.bson.types.ObjectId;

import java.util.Scanner;

public class employeeretentionpredictor {
    private static final String CONNECTION_STRING = "mongodb://localhost:27017";
    private static final String DB_NAME = "employeeDB";
    private static final String COLL_NAME = "employees";

    private final MongoClient mongoClient;
    private final MongoDatabase database;
    private final MongoCollection<Document> collection;

    public employeeretentionpredictor() {
        mongoClient = MongoClients.create(CONNECTION_STRING);
        database = mongoClient.getDatabase(DB_NAME);
        collection = database.getCollection(COLL_NAME);
    }



    public String createEmployee(String name, int age, int yearsAtCompany, double satisfaction, int
performanceRating) {
        Document doc = new Document("name", name)
                .append("age", age)
                .append("yearsAtCompany", yearsAtCompany)
                .append("satisfaction", satisfaction)
```

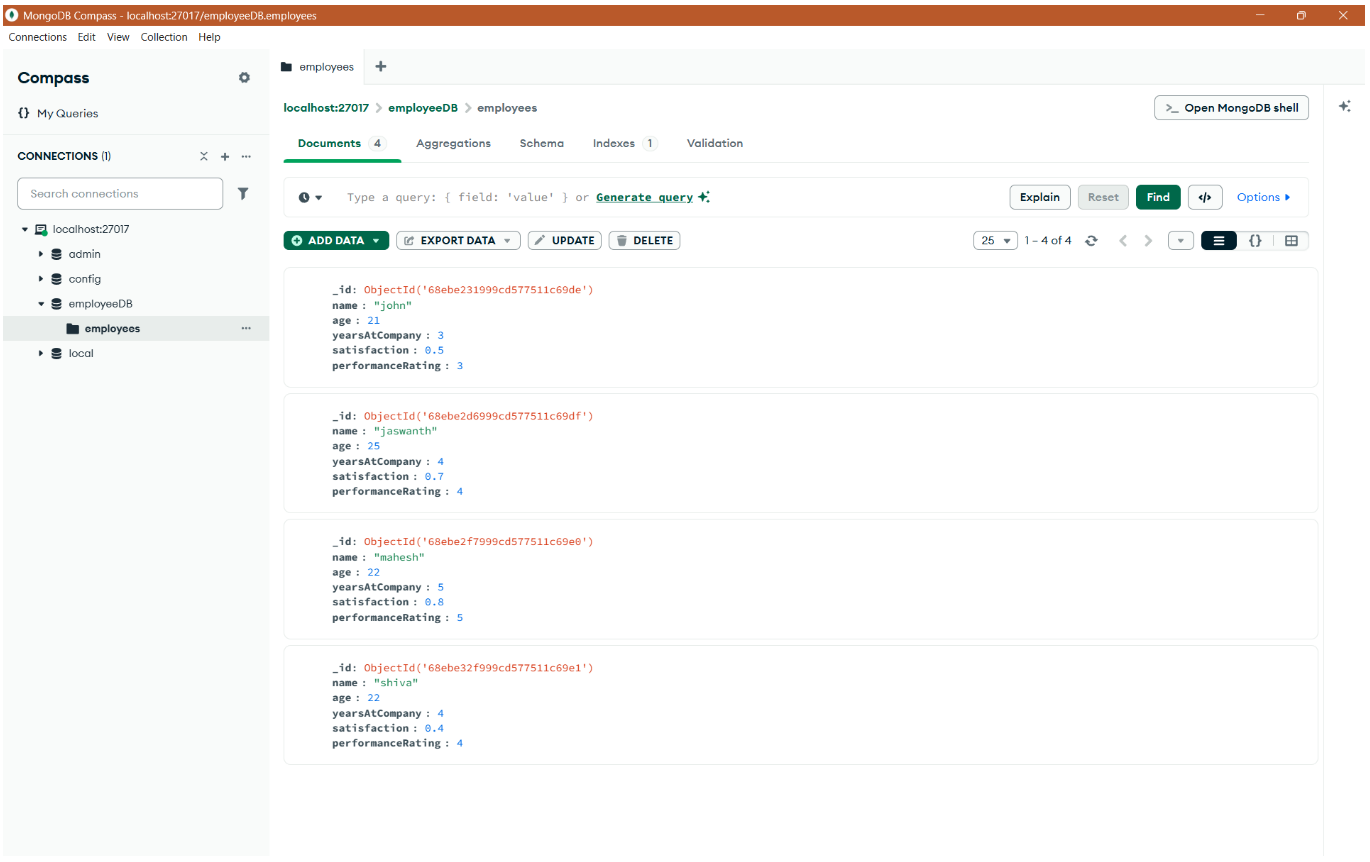
```
            .append("performanceRating", performanceRating);
        collection.insertOne(doc);
        ObjectId id = doc.getObjectId("_id");
        return id.toHexString();
    }

    public Document readEmployee(String idHex) {
        ObjectId id = new ObjectId(idHex);
        Document doc = collection.find(Filters.eq("_id", id)).first();
        return doc;
    }



    public boolean updateEmployee(String idHex, String name, Integer age, Integer yearsAtCompany,
Double satisfaction, Integer performanceRating) {
        ObjectId id = new ObjectId(idHex);
        Document setDoc = new Document();
        if (name != null) setDoc.append("name", name);
        if (age != null) setDoc.append("age", age);
        if (yearsAtCompany != null) setDoc.append("yearsAtCompany", yearsAtCompany);
        if (satisfaction != null) setDoc.append("satisfaction", satisfaction);
        if (performanceRating != null) setDoc.append("performanceRating", performanceRating);

        if (setDoc.isEmpty()) return false;
        Document update = new Document("$set", setDoc);
        return collection.updateOne(Filters.eq("_id", id), update).getModifiedCount() > 0;
    }



    public boolean deleteEmployee(String idHex) {
        ObjectId id = new ObjectId(idHex);
        return collection.deleteOne(Filters.eq("_id", id)).getDeletedCount() > 0;
    }
```

```
public void listEmployees() {
    System.out.println("=== Employees in DB ===");
    for (Document doc : collection.find()) {
        System.out.println(doc.toJson());
    }
}


public double predictRetentionScore(Document emp) {
    double satisfaction = emp.getDouble("satisfaction");
    int years = emp.getInteger("yearsAtCompany");
    int perf = emp.getInteger("performanceRating");

    double score = 0.5;
    score += satisfaction * 0.3;
    score += ((perf - 1) / 4.0) * 0.15;
    score += Math.min(years, 10) / 10.0 * 0.05;


    if (score < 0) score = 0;
    if (score > 1) score = 1;
    return score;
}

public String predictRetentionLabel(Document emp) {
    double score = predictRetentionScore(emp);
    if (score >= 0.7) return "Likely to stay (" + String.format("%.2f", score) + ")";
    if (score >= 0.45) return "Neutral / monitor (" + String.format("%.2f", score) + ")";
    return "At risk of leaving (" + String.format("%.2f", score) + ")";
}

public void close() {
    mongoClient.close();
}
```

```
    public static void main(String[] args) {
        employeeretentionpredictor app = new employeeretentionpredictor();
        Scanner sc = new Scanner(System.in);
        System.out.println("Employee Retention App (MongoDB). Connected to " +
CONNECTION_STRING + " DB: " + DB_NAME);

        while (true) {
            System.out.println("\nChoose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit");
            String cmd = sc.nextLine().trim();
            try {
                if (cmd.equals("1")) {
                    System.out.print("Name: "); String name = sc.nextLine();
                    System.out.print("Age: "); int age = Integer.parseInt(sc.nextLine());
                    System.out.print("Years at company: "); int years = Integer.parseInt(sc.nextLine());
                    System.out.print("Satisfaction (0.0 - 1.0): "); double sat =
Double.parseDouble(sc.nextLine());
                    System.out.print("Performance rating (1-5): "); int perf = Integer.parseInt(sc.nextLine());
                    String id = app.createEmployee(name, age, years, sat, perf);
                    System.out.println("Inserted with id: " + id);
                } else if (cmd.equals("2")) {
                    System.out.print("Employee id: "); String id = sc.nextLine();
                    Document d = app.readEmployee(id);
                    System.out.println(d == null ? "Not found" : d.toJson());
                } else if (cmd.equals("3")) {
                    System.out.print("Employee id: "); String id = sc.nextLine();
                    System.out.print("Name (or leave blank): "); String name = sc.nextLine(); if
(name.isEmpty()) name = null;
                    System.out.print("Age (or leave blank): "); String ageS = sc.nextLine(); Integer age =
ageS.isEmpty()?null:Integer.parseInt(ageS);
                    System.out.print("Years at company (or leave blank): "); String yrsS = sc.nextLine(); Integer
yrs = yrsS.isEmpty()?null:Integer.parseInt(yrsS);
                    System.out.print("Satisfaction (0-1) (or leave blank): "); String satS = sc.nextLine(); Double
sat = satS.isEmpty()?null:Double.parseDouble(satS);
```

```
                    System.out.print("Performance rating (1-5) (or leave blank): "); String pS = sc.nextLine();
Integer p = pS.isEmpty()?null:Integer.parseInt(pS);
                    boolean ok = app.updateEmployee(id, name, age, yrs, sat, p);
                    System.out.println(ok ? "Updated" : "No changes / not found");
                } else if (cmd.equals("4")) {
                    System.out.print("Employee id: "); String id = sc.nextLine();
                    boolean ok = app.deleteEmployee(id);
                    System.out.println(ok ? "Deleted" : "Not found");
                } else if (cmd.equals("5")) {
                    app.listEmployees();
                } else if (cmd.equals("6")) {
                    System.out.print("Employee id: "); String id = sc.nextLine();
                    Document d = app.readEmployee(id);
                    if (d == null) System.out.println("Not found");
                    else System.out.println(app.predictRetentionLabel(d));
                } else if (cmd.equals("7")) {
                    break;
                } else {
                    System.out.println("Unknown option");
                }
            } catch (Exception ex) {
                System.out.println("Error: " + ex.getMessage());
                ex.printStackTrace(System.out);
            }
        }
 app.close();
        System.out.println("Bye.");
    }
}
```

<u>OUTPUT:</u>



```
_id: ObjectId('68ebe231999cd577511c69de')
name : "john"
age : 21
yearsAtCompany : 3
satisfaction : 0.5
performanceRating : 3

_id: ObjectId('68ebe2d6999cd577511c69df')
name : "jaswanth"
age : 25
yearsAtCompany : 4
satisfaction : 0.7
performanceRating : 4

_id: ObjectId('68ebe2f7999cd577511c69e0')
name : "mahesh"
age : 22
yearsAtCompany : 5
satisfaction : 0.8
performanceRating : 5

_id: ObjectId('68ebe32f999cd577511c69e1')
name : "shiva"
age : 22
yearsAtCompany : 4
satisfaction : 0.4
performanceRating : 4
```

```
PS B:\employeeretentionpredictor> javac -cp "libs\mongodb-driver-sync-4.10.2.jar;libs\mongodb-driver-core-4.10.2.jar;libs\bson-4.10.2.jar;." employeeretentionpredictor.java

PS B:\employeeretentionpredictor> java -cp "libs\mongodb-driver-sync-4.10.2.jar;libs\mongodb-driver-core-4.10.2.jar;libs\bson-4.10.2.jar;." employeeretentionpredictor

Oct 12, 2025 11:23:46 PM com.mongodb.internal.diagnostics.logging.Loggers shouldUseSLF4J

WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component

=== Employee Retention Predictor ===

Connected to MongoDB at mongodb://localhost:27017, Database: employeeDB


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

1

Name: john

Age: 21
```

```
Years at company: 4

Satisfaction (0.0 - 1.0): 0.6

Performance rating (1-5): 4

Inserted with id: 68ebeb40e67f28333a1a047f


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

1

Name: jaswanth

Age: 22

Years at company: 5

Satisfaction (0.0 - 1.0): 0.8

Performance rating (1-5): 5

Inserted with id: 68ebeb64e67f28333a1a0480


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

1

Name: mahesh

Age: 22

Years at company: 3

Satisfaction (0.0 - 1.0): 0.6

Performance rating (1-5): 4

Inserted with id: 68ebeb7be67f28333a1a0481


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

1

Name: shiva

Age: 25

Years at company: 5

Satisfaction (0.0 - 1.0): 0.7

Performance rating (1-5): 1
```

```
Inserted with id: 68ebeb94e67f28333a1a0482


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

5

=== Employees in DB ===

{"_id": {"$oid": "68ebeb40e67f28333a1a047f"}, "name": "john", "age": 21, "yearsAtCompany": 4,
"satisfaction": 0.6, "performanceRating": 4}

{"_id": {"$oid": "68ebeb64e67f28333a1a0480"}, "name": "jaswanth", "age": 22, "yearsAtCompany": 5,
"satisfaction": 0.8, "performanceRating": 5}

{"_id": {"$oid": "68ebeb7be67f28333a1a0481"}, "name": "mahesh", "age": 22, "yearsAtCompany": 3,
"satisfaction": 0.6, "performanceRating": 4}

{"_id": {"$oid": "68ebeb94e67f28333a1a0482"}, "name": "shiva", "age": 25, "yearsAtCompany": 5,
"satisfaction": 0.7, "performanceRating": 1}


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

6

Employee id: 68ebeb94e67f28333a1a0482

Likely to stay (0.74)


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

2

Employee id: 68ebeb64e67f28333a1a0480

{"_id": {"$oid": "68ebeb64e67f28333a1a0480"}, "name": "jaswanth", "age": 22, "yearsAtCompany": 5,
"satisfaction": 0.8, "performanceRating": 5}


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

3

Employee id: 68ebeb94e67f28333a1a0482

Name (or leave blank):

Age (or leave blank):

Years at company (or leave blank): 1

Satisfaction (0-1) (or leave blank): 0.1
```

```
Performance rating (1-5) (or leave blank): 1

Updated


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

2

Employee id: 68ebeb94e67f28333a1a0482

{"_id": {"$oid": "68ebeb94e67f28333a1a0482"}, "name": "shiva", "age": 25, "yearsAtCompany": 1,
"satisfaction": 0.1, "performanceRating": 1}


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

5

=== Employees in DB ===

{"_id": {"$oid": "68ebeb40e67f28333a1a047f"}, "name": "john", "age": 21, "yearsAtCompany": 4,
"satisfaction": 0.6, "performanceRating": 4}

{"_id": {"$oid": "68ebeb64e67f28333a1a0480"}, "name": "jaswanth", "age": 22, "yearsAtCompany": 5,
"satisfaction": 0.8, "performanceRating": 5}

{"_id": {"$oid": "68ebeb7be67f28333a1a0481"}, "name": "mahesh", "age": 22, "yearsAtCompany": 3,
"satisfaction": 0.6, "performanceRating": 4}

{"_id": {"$oid": "68ebeb94e67f28333a1a0482"}, "name": "shiva", "age": 25, "yearsAtCompany": 1,
"satisfaction": 0.1, "performanceRating": 1}


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

6

Employee id: 68ebeb94e67f28333a1a0482

Neutral / monitor (0.54)


Choose: 1=create 2=read 3=update 4=delete 5=list 6=predict 7=exit

7
Bye.
```

CONCLUSION:

The **Employee Retention Predictor** project has been successfully completed, delivering a fully functional console-based application for managing informational articles. The system effectively demonstrates **Java and MongoDB integration** with complete **CRUD operations** (Create, Read, Update, Delete).

Key Achievements:

- ✅ **Successful Implementation** of all core features
- ✅ **Seamless MongoDB Integration** for data storage
- ✅ **User-Friendly Console Interface** with clear navigation
- ✅ **Robust Error Handling** and input validation
- ✅ **Efficient Search Functionality** across article content

Project Value:

This project serves as **"Employee Retention Predictor using Java and MongoDB"** can be described in terms of its **practical usefulness, learning outcomes, and business impact** and helps organizations identify employees at risk of leaving, enabling proactive retention strategies. It demonstrates practical use of Java with MongoDB for CRUD operations and data management. Additionally, it provides a foundation for data-driven decision-making and predictive analytics in HR.