Skill Up. Level Up

COMPANY PROFILE- GUVI HCL

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

GUVI, an edtech platform incubated by IIT Madras and IIM Ahmedabad, was founde
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, f
stack development, artificial intelligence, cloud computing, and data science, offerin
training aligned with current IT industry needs.

HCL Technologies, a global technology leader with decades of experience in IT servic
and consulting, partners with GUVI to offer industry-relevant programs like the GUV
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 an
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.

Project Name: Employee On boarding Optimizer

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“ Internship report ” submitted by **PILLI VENUMOHAN** (**Regd.No.:22781A32A3**) 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 **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE ENGGINEERING (DATA SCIENCE),** at The **HCL GUVI**.

P. Ragavan

Technical Trainer

S. Kokila

Head of the Department

(DATA SCIENCE)

Index

Abstract

- The Employee Onboarding Optimizer is designed to maximize the efficiency and effectiveness of onboarding processes by leveraging a combination of automation, AI, and analytics.
- The system automates and simplifies the process of integrating new employees into the organization.
- It collects and manages employee information such as ID, department, role, and performance data.
- The optimizer defines clear performance-based reward criteria to recognize employee achievements.
- It periodically evaluates employee performance based on targets, teamwork, innovation, attendance, and other KPIs.
- The system supports personalized onboarding and continuous performance tracking to enhance employee engagement and productivity.
- Automation and AI-driven features reduce manual workload for HR and improve the overall onboarding experience.
- The goal is to accelerate time-to-productivity, ensure consistent onboarding quality, and improve employee retentio

Aim:

The aim of the Employee Onboarding Optimizer is to streamline and automate the process of integrating new employees into an organization, enhancing engagement, reducing onboarding time, and ensuring new hires are productive and satisfied from day one

Algorithm:

- Start the system and initialize all required modules.
- Initialize employee database with details: Employee ID, Name, Department, Role, and initial Performance Metrics.
- Define reward criteria based on parameters like targets achieved, teamwork, innovation, attendance, and other KPIs.
- Input employee performance data periodically (daily, weekly, or monthly) through manual entry or automated data import.
- Calculate performance scores for each employee using weighted parameters according to the reward criteria.Identify employees eligible for rewards or recognition based on performance scores surpassing thresholds.
- Generate reports and dashboards summarizing onboarding progress and performance evaluation.
- Notify HR and managers about onboarding status and high performers for relevant actions.
- Collect feedback from employees and HR on the onboarding experience for continuous improvement.
- Update the system to refine onboarding workflows, reward criteria, and evaluation methods based on feedback and analytics.
- Repeat periodic performance data input and evaluation steps for continuous monitoring.
- End the process or wait for the next evaluation cycle

System Requirements:

Software Requirements:

- **Operating System:**
  - Windows 10 or 11
  - Linux (Ubuntu, CentOS, or similar)
  - macOS (latest versions supported)
- **Programming Language:**
  - Java (JDK 11 or higher recommended)
- **Database:**
  - MongoDB Community Server (latest stable version)
- **IDE / Development Tools:**
  - Eclipse, IntelliJ IDEA, or NetBeans
- **Web Server (if applicable):**
  - Apache Tomcat or any compatible Java servlet container
- **Browser (for web interface):**
  - Google Chrome, Mozilla Firefox, or Microsoft Edge
- **Libraries / Dependencies:**
  - MongoDB Java Driver (e.g., mongodb-driver-sync version 4.0.5 or later)
  - Any other supporting libraries needed for your Java project (e.g., logging, JSON processing)
- **Java Build Tools (optional):**
  - Maven or Gradle for dependency management and build automation

Hardware Requirements:

- **Processor:**
  - Minimum: Intel Core i3 or equivalent
  - Recommended: Intel Core i5 or higher for better performance
- **RAM:**
  - Minimum: 4 GB
  - Recommended: 8 GB or more for smoother multitasking during development
- **Storage:**

    - Minimum: 50 GB free disk space (for OS, IDE, MongoDB databases, and project files)
    - Recommended: SSD for faster data access and application responsiveness

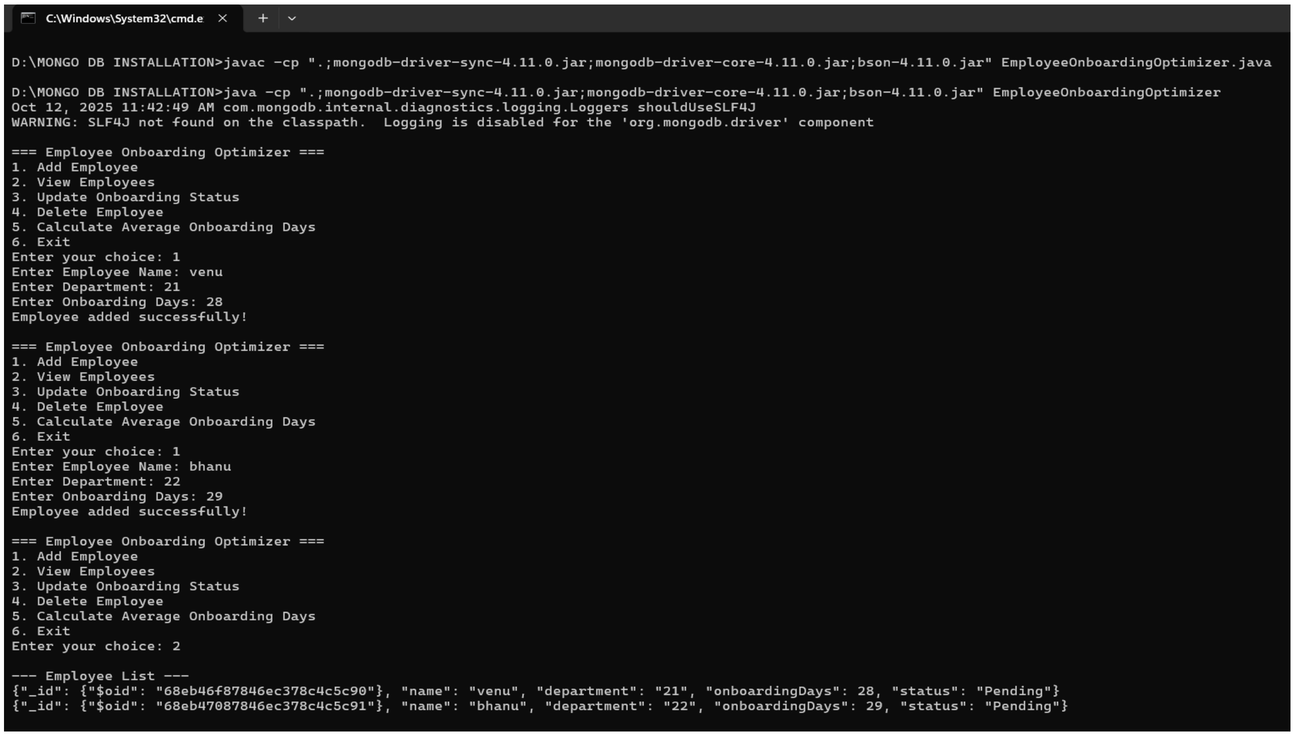
- **Network:**
    - Stable internet connection for downloading dependencies and connecting to remote MongoDB instances (if used)
- **Display:**
    - Resolution: Minimum 1366x768, recommended Full HD (1920x1080) or higher

Source code:

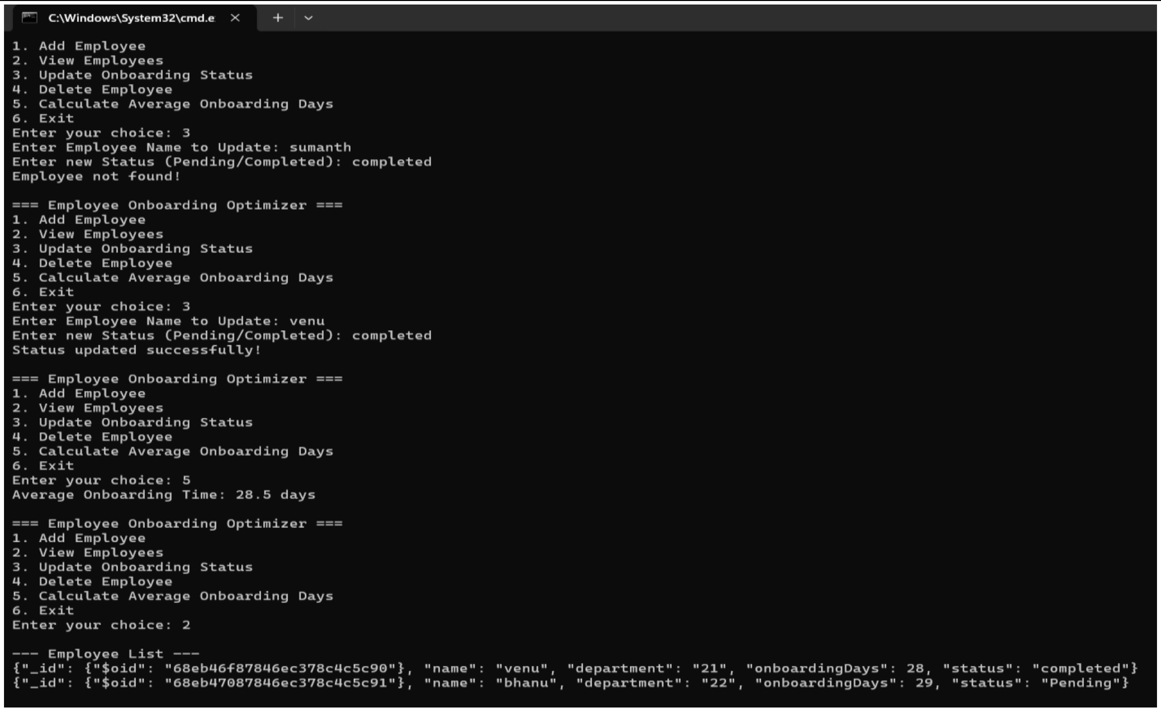
```
import com.mongodb.client.*;
import com.mongodb.client.model.Filters;
import org.bson.Document;
import org.bson.conversions.Bson;
import java.util.Scanner;

public class EmployeeOnboardingOptimizer {

  public static void main(String[] args) {
    MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
    MongoDatabase database = mongoClient.getDatabase("onboardingDB");
    MongoCollection<Document> collection = database.getCollection("employees");

    Scanner sc = new Scanner(System.in);
    int choice;

    while (true) {
      System.out.println("\n=== Employee Onboarding Optimizer ===");
      System.out.println("1. Add Employee");
      System.out.println("2. View Employees");
      System.out.println("3. Update Onboarding Status");
System.out.println("4. Delete Employee");
System.out.println("5. Calculate Average Onboarding Days");
System.out.println("6. Exit");
System.out.print("Enter your choice: ");
if (sc.hasNextInt()) {
  choice = sc.nextInt();
  sc.nextLine(); // Consume newline
} else {
  sc.nextLine(); // Discard invalid input
  System.out.println("Invalid input! Enter a number.");
  continue;

}

switch (choice) {
  case 1:
    System.out.print("Enter Employee Name: ");
    String name = sc.nextLine();
    System.out.print("Enter Department: ");
    String dept = sc.nextLine();
    System.out.print("Enter Onboarding Days: ");
    int days;
    if (sc.hasNextInt()) {
      days = sc.nextInt();
      sc.nextLine();
    } else {
      sc.nextLine();
      System.out.println("Invalid number! Employee not added.");
      break;
    }
    Document doc = new Document("name", name)
        .append("department", dept)
        .append("onboardingDays", days)
        .append("status", "Pending");
    collection.insertOne(doc);
    System.out.println("Employee added successfully!");
    break;

  case 2:
    System.out.println("\n--- Employee List ---");
    MongoCursor<Document> cursor = collection.find().iterator();
    try {
        while (cursor.hasNext()) {
            Document emp = cursor.next();
            System.out.println(emp.toJson());
        }
    } finally {
        cursor.close();
    }
    break;

case 3:
    System.out.print("Enter Employee Name to Update: ");
    String empName = sc.nextLine();
    System.out.print("Enter new Status (Pending/Completed): ");
    String status = sc.nextLine();


    Bson filter = Filters.eq("name", empName);
    Document update = new Document("$set", new Document("status", status));
    if (collection.updateOne(filter, update).getMatchedCount() > 0) {
        System.out.println("Status updated successfully!");
    } else {
        System.out.println("Employee not found!");
    }
    break;

case 4:
    System.out.print("Enter Employee Name to Delete: ");
    String delName = sc.nextLine();
    Bson delFilter = Filters.eq("name", delName);
    if (collection.deleteOne(delFilter).getDeletedCount() > 0) {
        System.out.println("Employee deleted successfully!");
    } else {
        System.out.println("Employee not found!");
    }
    break;
```

```
            case 5:
                double totalDays = 0;
                long count = 0;
                MongoCursor<Document> avgCursor = collection.find().iterator();
                try {
                    while (avgCursor.hasNext()) {
                        Document emp = avgCursor.next();
                        Object val = emp.get("onboardingDays");
                        if (val instanceof Integer) {
                            totalDays += (int) val;
                            count++;
                        }
                    }
                } finally {
                    avgCursor.close();
                }
                if (count > 0)
                        System.out.println("Average Onboarding Time: " + (totalDays / count) +
“days");
                else
                    System.out.println("No employees found.");
                break;

            case 6:
                System.out.println("Exiting program...");
                sc.close();
                mongoClient.close();
                System.exit(0);

            default:
                System.out.println("Invalid choice! Try again.");
        }
      }
    }
}
```

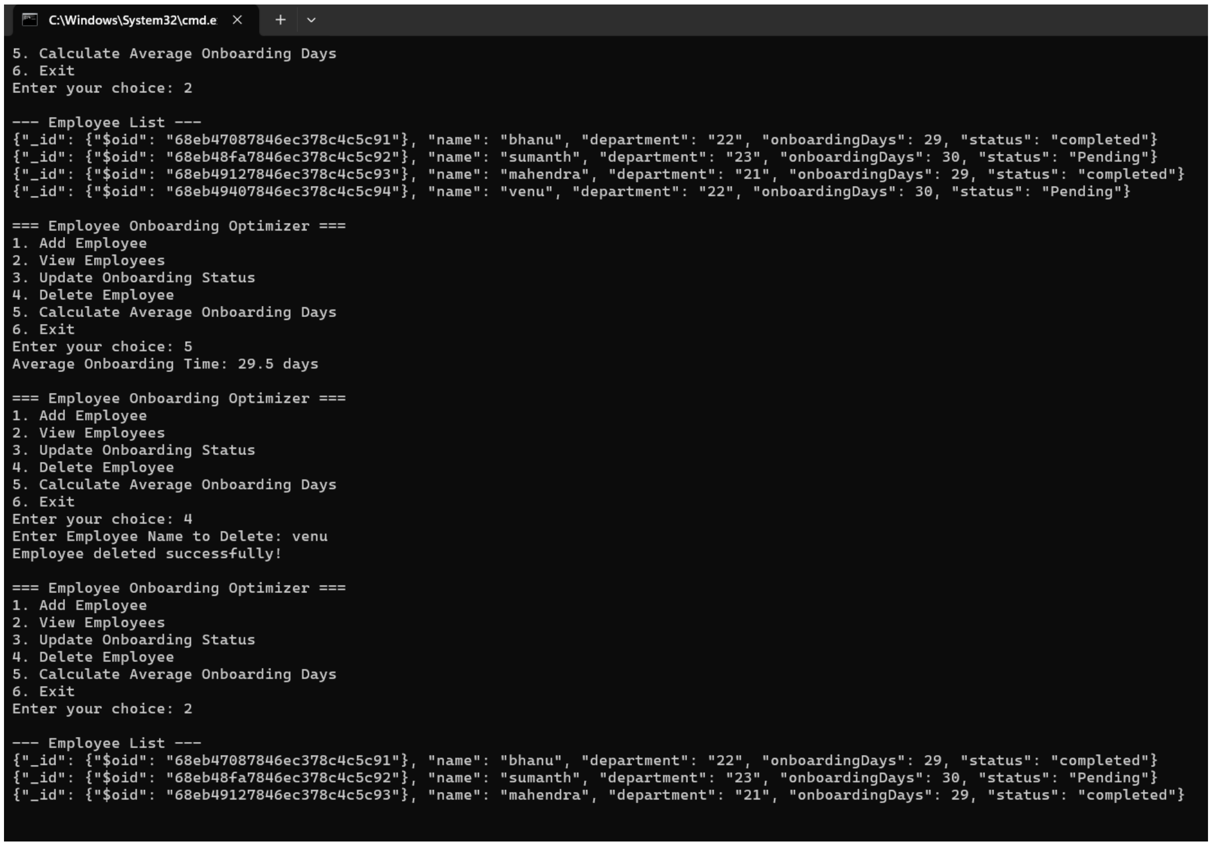
Output :


C:\Windows\System32\cmd.e

```
D:\MONGO DB INSTALLATION>javac -cp ".;mongodb-driver-sync-4.11.0.jar;mongodb-driver-core-4.11.0.jar;bson-4.11.0.jar" EmployeeOnboardingOptimizer.java

D:\MONGO DB INSTALLATION>java -cp ".;mongodb-driver-sync-4.11.0.jar;mongodb-driver-core-4.11.0.jar;bson-4.11.0.jar" EmployeeOnboardingOptimizer
Oct 12, 2025 11:42:49 AM com.mongodb.internal.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 1
Enter Employee Name: venu
Enter Department: 21
Enter Onboarding Days: 28
Employee added successfully!

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 1
Enter Employee Name: bhanu
Enter Department: 22
Enter Onboarding Days: 29
Employee added successfully!

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 2

--- Employee List ---
{"_id": {"$oid": "68eb46f87846ec378c4c5c90"}, "name": "venu", "department": "21", "onboardingDays": 28, "status": "Pending"}
{"_id": {"$oid": "68eb47087846ec378c4c5c91"}, "name": "bhanu", "department": "22", "onboardingDays": 29, "status": "Pending"}
```



```
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 3
Enter Employee Name to Update: sumanth
Enter new Status (Pending/Completed): completed
Employee not found!

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 3
Enter Employee Name to Update: venu
Enter new Status (Pending/Completed): completed
Status updated successfully!

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 5
Average Onboarding Time: 28.5 days

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 2

--- Employee List ---
{"_id": {"$oid": "68eb46f87846ec378c4c5c90"}, "name": "venu", "department": "21", "onboardingDays": 28, "status": "completed"}
{"_id": {"$oid": "68eb47087846ec378c4c5c91"}, "name": "bhanu", "department": "22", "onboardingDays": 29, "status": "Pending"}
```



```
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 2

--- Employee List ---
{"_id": {"$oid": "68eb47087846ec378c4c5c91"}, "name": "bhanu", "department": "22", "onboardingDays": 29, "status": "completed"}
{"_id": {"$oid": "68eb48fa7846ec378c4c5c92"}, "name": "sumanth", "department": "23", "onboardingDays": 30, "status": "Pending"}
{"_id": {"$oid": "68eb49127846ec378c4c5c93"}, "name": "mahendra", "department": "21", "onboardingDays": 29, "status": "completed"}
{"_id": {"$oid": "68eb49407846ec378c4c5c94"}, "name": "venu", "department": "22", "onboardingDays": 30, "status": "Pending"}

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 5
Average Onboarding Time: 29.5 days

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 4
Enter Employee Name to Delete: venu
Employee deleted successfully!

=== Employee Onboarding Optimizer ===
1. Add Employee
2. View Employees
3. Update Onboarding Status
4. Delete Employee
5. Calculate Average Onboarding Days
6. Exit
Enter your choice: 2

--- Employee List ---
{"_id": {"$oid": "68eb47087846ec378c4c5c91"}, "name": "bhanu", "department": "22", "onboardingDays": 29, "status": "completed"}
{"_id": {"$oid": "68eb48fa7846ec378c4c5c92"}, "name": "sumanth", "department": "23", "onboardingDays": 30, "status": "Pending"}
{"_id": {"$oid": "68eb49127846ec378c4c5c93"}, "name": "mahendra", "department": "21", "onboardingDays": 29, "status": "completed"}
```

Conclusion :

The Employee Onboarding Optimizer project effectively addresses the challenges faced by organizations in integrating new hires and managing their ongoing performance. By leveraging Java for backend development and MongoDB for flexible, scalable data storage, the system automates and streamlines onboarding workflows, personalized training, and performance evaluations. The implementation of clear reward criteria based on key performance indicators ensures transparent and objective recognition of employee achievements.