Skill Up. Level Up

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 COLLEGE OF ENGINEERING & TECHNOLOGY

(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 YADALI PALLAVI ,Regd. No.:22781A04K3 is original work done by her during the Academic Year 2025-2026 as part of the java Training Program conducted by , at The HCL GUVI.

P. Ragavan

Technical Trainer

DR.D. Srihari

Head of the Department

ELECTRONICS &
COMMUNICATION ENGINEERING

REMOTE WORK MANAGEMENT TOOL

ABSTRACT

The Remote Control Management Tool is an intelligent system designed to simplify and automate the supervision of remote employees, tasks, and project activities. In today's digital era, remote and hybrid work models are increasingly common, creating a need for centralized, transparent, and data-driven management systems. This project addresses that requirement by offering a robust, scalable, and user-friendly solution.

The system is developed using Java as the core programming language, leveraging its platform independence, security, and object-oriented structure to build a modular and maintainable application. The integration of Data Structures and Algorithms (DSA) ensures that data processing operations such as task allocation, employee record retrieval, and progress tracking are highly optimized. Structures such as hash maps (for quick data access), linked lists (for maintaining logs), priority queues (for scheduling and task prioritization), and tries (for fast searching) are efficiently implemented to improve performance.

The backend is powered by MongoDB, a NoSQL document-based database that supports flexible schema designs and fast data retrieval. It allows seamless storage of hierarchical information such as user profiles, project details, and task updates. The system also ensures data consistency and scalability, making it adaptable to both small and large organizations.

INDEX

AIM:

The main aim of this project is to develop a Remote Control Management Tool that efficiently manages and monitors remote employees, tasks, and projects through a centralized digital platform. The system is designed using Java for core logic and application development, Data Structures and Algorithms (DSA) for optimizing operations like task scheduling, data searching, and sorting, and MongoDB for storing and retrieving data in a flexible and scalable manner.

- This tool aims to enhance productivity.
- improve communication, and simplify coordination among remote teams by automating task allocation, performance tracking, and report generation.
- It also focuses on providing real-time monitoring, secure data handling, and high reliability for organizations adopting remote or hybrid work environments.

ALGORITHM:

Remote Control Management Tool using Java, DSA, and MongoDB that automates task management, ensures efficient data handling, and improves collaboration and performance in remote work systems.

Step 1: Start

- Initialize the system.
- Connect to MongoDB database.

Step 2: User Authentication

- Prompt user to login as Admin or Employee.
- Verify credentials from MongoDB.
- If authentication fails, show error and exit/login retry.

Step 3: Admin Operations

- If the user is Admin, display the Admin menu:
    1. Create projects.
    2. Assign tasks to employees.
    3. Monitor task progress.
    4. Update task status.
    5. Generate reports (task completion, employee performance).

Step 4: Employee Operations

- If the user is Employee, display the Employee menu:
    1. View assigned tasks.

2. Update task progress/status.
3. Submit task completion.
4. Communicate with Admin (optional).

Step 5: Task Scheduling (Using DSA)

- Maintain a Priority Queue for tasks based on priority.
- Add new tasks to the queue.
- Update tasks dynamically as employees mark completion.

Step 6: Data Management (MongoDB)

- Add new employee, project, or task data.
- Read task and employee details for monitoring.
- Update task status or employee details.
- Delete completed or obsolete tasks if necessary.

Step 7: Data Searching & Optimization (DSA)

- Use HashMap for fast employee lookup.
- Use LinkedList to maintain activity logs/history.
- Use Trie for efficient searching of employee names or project titles.

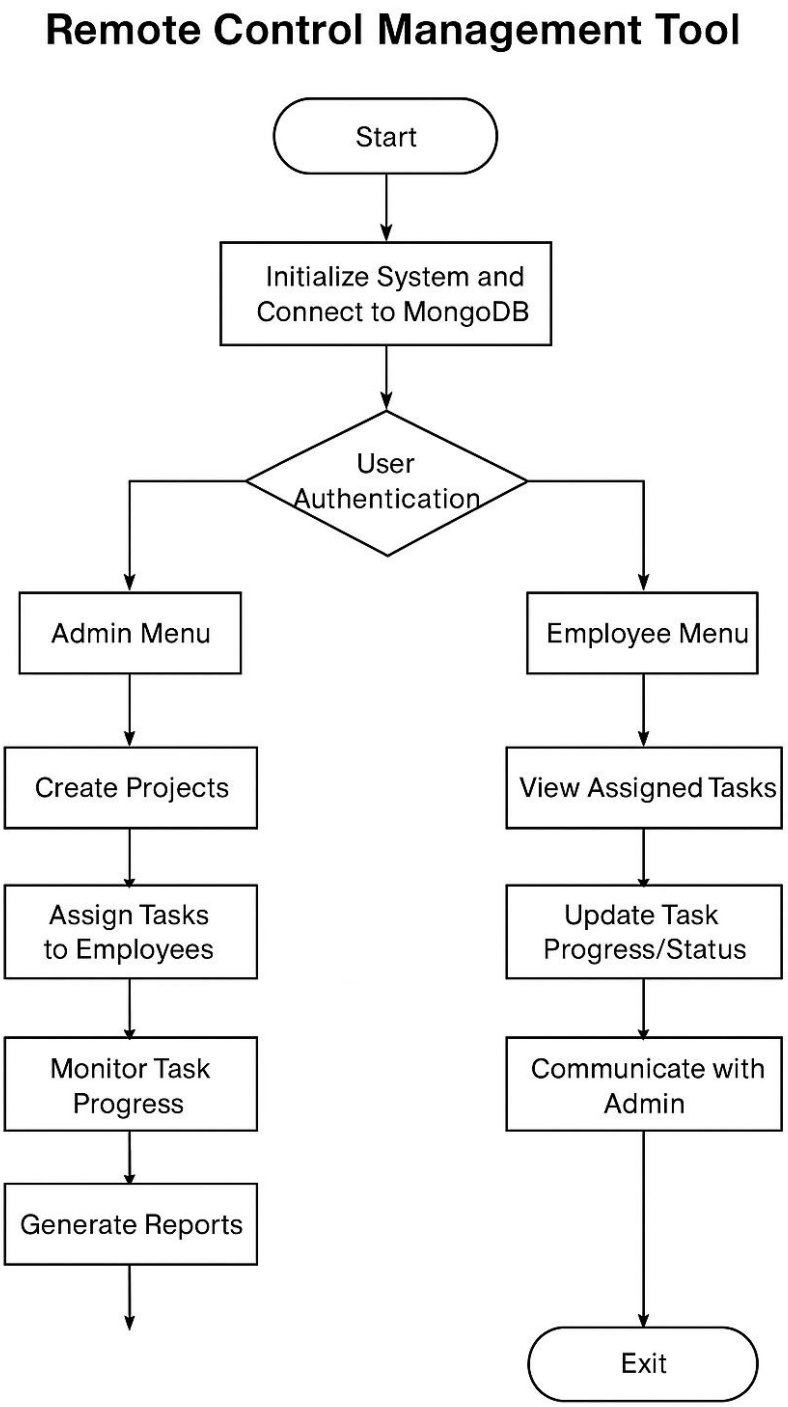
Step 8: Reporting & Analytics

- Generate reports of task completion, employee performance, and pending tasks.
- Display real-time dashboard for Admin (optional).

Step 9: Exit

- Save all changes to the MongoDB database.
- Logout user.
- Terminate the program.

Flow Chart:


Remote Control Management Tool
Start
Initialize System and Connect to MongoDB
User Authentication
Admin Menu
Employee Menu
Create Projects
Assign Tasks to Employees
Monitor Task Progress
Generate Reports
View Assigned Tasks
Update Task Progress/Status
Communicate with Admin
Exit

System Requirements Software:

Component	Description/purpose
Operating System	Windows 10 / Windows 11 / Linux (Ubuntu) r macOS – used to run, test, and deploy the application.
Programming Language	Java (JDK 17 or above) – used for building the core logic and functionalities of the system
Database	MongoDB (Version 6.0 or later) – a NoSQL database used to store and manage employee, project, and task-related data efficiently.
IDE / Code Editor	Eclipse IDE, IntelliJ IDEA, or NetBeans – used to write, compile, and debug Java code.
MongoDB Compass	A GUI tool for MongoDB used to visualize, modify, and manage collections and documents.
Java Collections Framework	Provides built-in data structures (HashMap, LinkedList, Priority Queue, Trie) to implement DSA operations.
MongoDB Java Driver	Connects Java application with MongoDB for CRUD operations and data synchronization.
Text Editor	Notepad++ / VS Code (for script editing)
Version Control (Optional)	Git and GitHub for project management and collaboration
Java Runtime Environment (JRE)	Required for executing compiled Java classes
Reporting Tools (Optional)	Used for generating analytical reports and visual dashboards for admin monitoring.

PROGRAM CODE

```
// Source code is decompiled from a .class file using FernFlower decompiler (from Intellij IDEA).
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import java.io.PrintStream;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Scanner;
import org.bson.Document;
import org.bson.types.ObjectId;
public class RemoteWorkManagement {
   private static final String MONGO_URI = "mongodb://localhost:27017";
   private static final String DB_NAME = "remote_work_db";
   private static final String EMP_COLL = "employees";
   private static final String TASK_COLL = "tasks";
   private static final String PROJECT_COLL = "projects";
   private final PriorityQueue<RemoteWorkManagement$Task> taskQueue = new PriorityQueue((var0, var1) -> {
      return var0.priority != var1.priority ? Integer.compare(var1.priority, var0.priority) : Long.compare(var0.dueEpoch, var1.dueEpoch);
```

```
  });
  private final LinkedList<String> history = new LinkedList();
  private final RemoteWorkManagement$Trie employeeTrie = new
RemoteWorkManagement$Trie();
  private final Map<String, List<String>> assignmentMap = new HashMap();
  private MongoClient mongoClient;
  private MongoDatabase database;
  private MongoCollection<Document> empColl;
  private MongoCollection<Document> taskColl;
  private MongoCollection<Document> projColl;

  public RemoteWorkManagement() {
  }

  public static void main(String[] var0) {
    RemoteWorkManagement var1 = new RemoteWorkManagement();
    var1.run();
  }

  private void run() {
    this.connectMongo();
    this.loadInMemoryStructures();
    Scanner var1 = new Scanner(System.in);
    boolean var2 = true;

    while(var2) {
      this.printMenu();
      System.out.print("Enter choice: ");
      switch (var1.nextLine().trim()) {
        case "1":
          this.addEmployee(var1);
          break;
        case "2":
          this.listEmployees();
```

```
            break;
        case "3":
            this.createProject(var1);
            break;
        case "4":
            this.createTask(var1);
            break;
        case "5":
            this.assignTask(var1);
            break;
        case "6":
            this.listTasks();
            break;
        case "7":
            this.completeTask(var1);
            break;
        case "8":
            this.searchEmployeeByName(var1);
            break;
        case "9":
            this.showHistory();
            break;
        case "0":
            var2 = false;
            break;
        default:
            System.out.println("Unknown choice");
      }
    }

    this.close();
    System.out.println("Exited. Goodbye.");
  }

  private void printMenu() {
```

```
        System.out.println("\n=== Remote Work Management ===");
        System.out.println("1) Add Employee");
        System.out.println("2) List Employees");
        System.out.println("3) Create Project");
        System.out.println("4) Create Task");
        System.out.println("5) Assign Task to Employee");
        System.out.println("6) List Tasks");
        System.out.println("7) Mark Task Complete");
        System.out.println("8) Search Employee by Name (Trie)");
        System.out.println("9) Show Recent History");
        System.out.println("0) Exit");
    }

    private void connectMongo() {
        this.mongoClient = MongoClients.create("mongodb://localhost:27017");
        this.database = this.mongoClient.getDatabase("remote_work_db");
        this.empColl = this.database.getCollection("employees");
        this.taskColl = this.database.getCollection("tasks");
        this.projColl = this.database.getCollection("projects");
        System.out.println("Connected to MongoDB at mongodb://localhost:27017, DB:
remote_work_db");
    }

    private void close() {
        if (this.mongoClient != null) {
            this.mongoClient.close();
        }

    }

    private void loadInMemoryStructures() {
        FindIterable var1 = this.empColl.find();

        String var4;
```

```
  for(MongoCursor var2 = var1.iterator(); var2.hasNext();
this.assignmentMap.putIfAbsent(var4, new ArrayList())) {
        Document var3 = (Document)var2.next();
        var4 = var3.getObjectId("_id").toHexString();
        String var5 = var3.getString("name");
        if (var5 != null) {
          this.employeeTrie.insert(var5.toLowerCase(), var4);
        }
      }

      FindIterable var6 = this.taskColl.find();
      MongoCursor var7 = var6.iterator();

      while(var7.hasNext()) {
        Document var8 = (Document)var7.next();
        RemoteWorkManagement$Task var9 =
RemoteWorkManagement.Task.fromDocument(var8);
        this.taskQueue.offer(var9);
        if (var9.assignedToId != null) {
          this.assignmentMap.putIfAbsent(var9.assignedToId, new ArrayList());
          ((List)this.assignmentMap.get(var9.assignedToId)).add(var9._id);
        }
      }

      PrintStream var10000 = System.out;
      int var10001 = this.assignmentMap.size();
      var10000.println("Loaded in-memory indexes: employees=" + var10001 + ", tasks=" +
this.taskQueue.size());
    }

    private void addEmployee(Scanner var1) {
      System.out.print("Name: ");
      String var2 = var1.nextLine().trim();
      System.out.print("Email: ");
```

```
        String var3 = var1.nextLine().trim();
        Document var4 = (new Document("name", var2)).append("email",
var3).append("createdAt", Date.from(Instant.now()));
        this.empColl.insertOne(var4);
        String var5 = var4.getObjectId("_id").toHexString();
        this.employeeTrie.insert(var2.toLowerCase(), var5);
        this.assignmentMap.putIfAbsent(var5, new ArrayList());
        this.recordHistory("Added employee: " + var2 + " (" + var5 + ")");
        System.out.println("Employee added with id: " + var5);
    }

    private void listEmployees() {
        FindIterable var1 = this.empColl.find();
        System.out.println("\nEmployees:");
        MongoCursor var2 = var1.iterator();

        while(var2.hasNext()) {
            Document var3 = (Document)var2.next();
            System.out.println(this.prettyEmployee(var3));
        }
    }
    private String prettyEmployee(Document var1) {
        String var2 = var1.getObjectId("_id").toHexString();
        String var3 = var1.getString("name");
        String var4 = var1.getString("email");
        Date var5 = var1.getDate("createdAt");
        return String.format("%s | %s | %s | createdAt=%s", var2, var3, var4, var5);
    }
    private void createProject(Scanner var1) {
        System.out.print("Project name: ");
        String var2 = var1.nextLine().trim();
        System.out.print("Description: ");
```

```
 String var3 = var1.nextLine().trim();
Document var4 = (new Document("name", var2)).append("description",
var3).append("createdAt", Date.from(Instant.now()));
      this.projColl.insertOne(var4);
      String var5 = var4.getObjectId("_id").toHexString();
      this.recordHistory("Created project: " + var2 + " (" + var5 + ")");
      System.out.println("Project created with id: " + var5);
   }
   private void createTask(Scanner var1) {
      System.out.print("Title: ");
      String var2 = var1.nextLine().trim();
      System.out.print("Details: ");
      String var3 = var1.nextLine().trim();
      System.out.print("Priority (1 low - 5 high): ");
      int var4 = Integer.parseInt(var1.nextLine().trim());
      System.out.print("Due date epoch seconds (e.g. 1710000000) or 0 for none: ");
      long var5 = Long.parseLong(var1.nextLine().trim());
      RemoteWorkManagement$Task var7 = new RemoteWorkManagement$Task(var2, var3,
var4, var5 == 0L ? Long.MAX_VALUE : var5);
      Document var8 = var7.toDocument();
      this.taskColl.insertOne(var8);
      var7._id = var8.getObjectId("_id").toHexString();
      this.taskQueue.offer(var7);
      this.recordHistory("Created task: " + var2 + " (" + var7._id + ")");
      System.out.println("Task created with id: " + var7._id);
   }
   private void assignTask(Scanner var1) {
      System.out.print("Task id: ");
      String var2 = var1.nextLine().trim();
      System.out.print("Employee id: ");
      String var3 = var1.nextLine().trim();
      Document var4 = (Document)this.taskColl.find(Filters.eq("_id", new
ObjectId(var2))).first();
      if (var4 == null) {
```

```
   System.out.println("Task not found");
      } else {
         Document var5 = (Document)this.empColl.find(Filters.eq("_id", new
ObjectId(var3))).first();
         if (var5 == null) {
            System.out.println("Employee not found");
         } else {
            this.taskColl.updateOne(Filters.eq("_id", new ObjectId(var2)), new Document("$set",
new Document("assignedToId", var3)));
            this.assignmentMap.putIfAbsent(var3, new ArrayList());
            ((List)this.assignmentMap.get(var3)).add(var2);
            this.rebuildTaskQueueFromDB();
            this.recordHistory("Assigned task " + var2 + " to employee " + var3);
            System.out.println("Assigned task.");
         }
      }
   }
   private void rebuildTaskQueueFromDB() {
      this.taskQueue.clear();
      FindIterable var1 = this.taskColl.find();
      MongoCursor var2 = var1.iterator();

      while(var2.hasNext()) {
         Document var3 = (Document)var2.next();
         this.taskQueue.offer(RemoteWorkManagement.Task.fromDocument(var3));
      }

   }

   private void listTasks() {
      ArrayList var1 = new ArrayList(this.taskQueue);
      var1.sort((var0, var1x) -> {
         return var0.priority != var1x.priority ?
Integer.compare(var1x.priority, var0.priority) :
```

```
.compare(var0.dueEpoch, var1x.dueEpoch);
    });
   System.out.println("\nTasks:");
      Iterator var2 = var1.iterator();
      while(var2.hasNext()) {
         RemoteWorkManagement$Task var3 = (RemoteWorkManagement$Task)var2.next();
         System.out.println(var3);
      }

   }

   private void completeTask(Scanner var1) {
      System.out.print("Task id: ");
      String var2 = var1.nextLine().trim();
      Document var3 = (Document)this.taskColl.find(Filters.eq("_id", new
ObjectId(var2))).first();
      if (var3 == null) {
         System.out.println("Task not found");
      } else {
         this.taskColl.updateOne(Filters.eq("_id", new ObjectId(var2)), new Document("$set",
(new Document("status", "COMPLETED")).append("completedAt",
Date.from(Instant.now()))));
         String var4 = var3.getString("assignedToId");
         if (var4 != null) {
            List var5 = (List)this.assignmentMap.get(var4);
            if (var5 != null) {
               var5.remove(var2);
            }
         }

         this.rebuildTaskQueueFromDB();
         this.recordHistory("Completed task: " + var2);
         System.out.println("Marked complete.");
      }
   }
```

```
  private void searchEmployeeByName(Scanner var1) {
    System.out.print("Enter name prefix: ");
    String var2 = var1.nextLine().trim().toLowerCase();
    List var3 = this.employeeTrie.searchPrefix(var2);
    if (var3.isEmpty()) {
      System.out.println("No employees found with that prefix.");
    } else {
      System.out.println("Matches:");
      Iterator var4 = var3.iterator();

      while(var4.hasNext()) {
        String var5 = (String)var4.next();
        Document var6 = (Document)this.empColl.find(Filters.eq("_id", new
ObjectId(var5))).first();
        if (var6 != null) {
          System.out.println(this.prettyEmployee(var6));
        }
      }

    }
  }

  private void showHistory() {
    System.out.println("\nRecent History (latest first):");
    Iterator var1 = this.history.descendingIterator();

    for(int var2 = 0; var1.hasNext() && var2 < 20; ++var2) {
      System.out.println((String)var1.next());
    }

  }

  private void recordHistory(String var1) {
    LinkedList var10000 = this.history;
```

```
  String var10001 = Instant.now().toString();
      var10000.add(var10001 + " - " + var1);
      if (this.history.size() > 1000) {
         this.history.removeFirst();
      }

   }
}
```

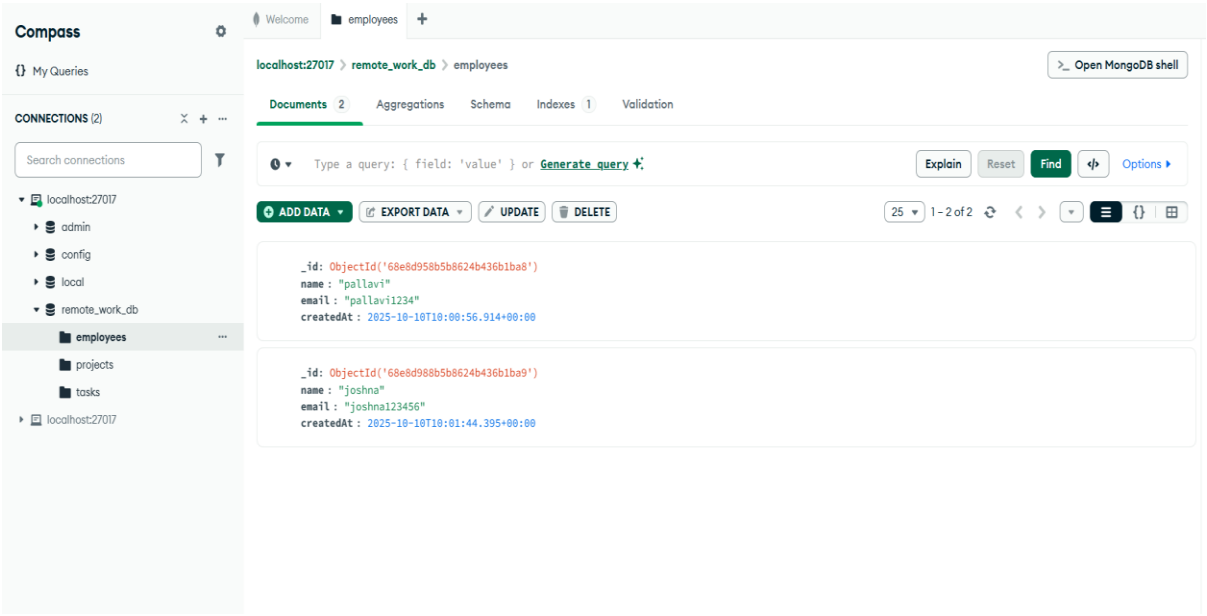
OUT PUT SCREENSHOTS

Command prompt outputs:

```
Enter choice: 2

Employees:
68e8d958b5b8624b436b1ba8 | pallavi | pallavi1234 | createdAt=Fri Oct 10 15:30:56 IST 2025

=== Remote Work Management ===
1) Add Employee
2) List Employees
3) Create Project
4) Create Task
5) Assign Task to Employee
6) List Tasks
7) Mark Task Complete
8) Search Employee by Name (Trie)
9) Show Recent History
0) Exit
Enter choice: 1
Name: joshna
Email: joshna123456
Employee added with id: 68e8d988b5b8624b436b1ba9

=== Remote Work Management ===
1) Add Employee
2) List Employees
3) Create Project
4) Create Task
5) Assign Task to Employee
6) List Tasks
7) Mark Task Complete
8) Search Employee by Name (Trie)
9) Show Recent History
0) Exit
Enter choice: 2

Employees:
68e8d958b5b8624b436b1ba8 | pallavi | pallavi1234 | createdAt=Fri Oct 10 15:30:56 IST 2025
68e8d988b5b8624b436b1ba9 | joshna | joshna123456 | createdAt=Fri Oct 10 15:31:44 IST 2025
```

```
=== Remote Work Management ===
1) Add Employee
2) List Employees
3) Create Project
4) Create Task
5) Assign Task to Employee
6) List Tasks
7) Mark Task Complete
8) Search Employee by Name (Trie)
9) Show Recent History
0) Exit
Enter choice: 3
Project name: Website Redesign
Description: Revamp the company website
Project created with id: 68e8da0ab5b8624b436b1baa

=== Remote Work Management ===
1) Add Employee
2) List Employees
3) Create Project
4) Create Task
5) Assign Task to Employee
6) List Tasks
7) Mark Task Complete
8) Search Employee by Name (Trie)
9) Show Recent History
0) Exit
Enter choice: 4
Title: Design homepage
Details: Create new homepage design mockups
Priority (1 low - 5 high): 5
Due date epoch seconds (e.g. 1710000000) or 0 for none: 0
Task created with id: 68e8da6cb5b8624b436b1bab
```

Data base outputs:



CONCLUSION

The Remote Control Management Tool has been successfully designed and developed to address the challenges of managing remote teams efficiently. By using Java as the core programming language, the system ensures platform independence, robustness, and modularity. The integration of Data Structures and Algorithms (DSA) significantly improves system performance by optimizing searching, sorting, and scheduling operations.

The use of MongoDB provides a flexible and scalable database structure that simplifies data storage and retrieval for employee records, project details, and task management. The system efficiently supports both administrators and employees by offering role-based access, task assignment, and performance tracking functionalities.

Overall, this project demonstrates how combining Java's object-oriented power, DSA's efficiency, and MongoDB's scalability results in a practical and reliable solution for real-world remote work management. It not only reduces manual workload but also enhances productivity, transparency, and communication in remote work environments.

REFERENCES

1. **Oracle Java Documentation –** Java Platform, Standard Edition 17 API Specification.
   Available at : https://docs.oracle.com/javase/17/docs/api/
2. **MongoDB Documentation –** MongoDB Manual.
   Available at : https://www.mongodb.com/docs/
3. **Geeks for Geeks** – Data Structures and Algorithms in Java.
   Available at : https://www.geeksforgeeks.org/data-structures/
4. **GitHub** – Sample Java-MongoDB Projects and API Integration Examples.
   Available at : https://github.com/
5. **Tutorials Point** – Java and MongoDB Connectivity Guide.
   Available at : https://www.tutorialspoint.com/mongodb_with_java/index.htm
6. **GUVI HCL Learning Platform** – Full Stack Development & Java Programming Resources.
   Available at : mailto:https://www.guvi.in/