



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.

Project Name: Developer Onboarding portal

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

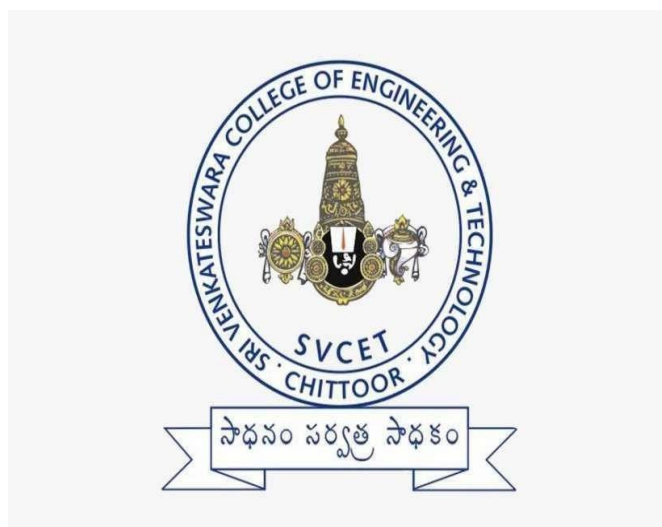
R.V.S.Nagar, Chittoor-517127.(A.P)

(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 “ Internship report ” submitted by **SANGATI REDDY BABU (Regd.No.:22781A3142)** 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 ENGINEERING (ARTIFICIAL INTELLIGENCE)**, at The **HCL GUVI**.

P. Ragavan

Technical Trainer

DR. M. Lavanya

Head of the Department

Computer Science Engineering (Artificial Intelligence)

Index

S.No	Content	Page No
1	Company Profile	1
2	Certificate	2
3	Index	3
4	Abstract	4
5	Aim	5
6	Algorithm	6
7	System Requirements	7
8	Program Code	8
9	Output Screenshots / Output	10
10	Conclusion	12

Abstract

- The Developer Onboarding Portal is a centralized platform designed to help new developers quickly integrate into the development team and become productive.
- It provides easy access to essential resources like documentation, code samples, API references, and onboarding guides.
- The portal offers automated workflows for setting up development environments, accessing tools, and managing permissions.
- Interactive features such as API explorers and sandbox environments enable developers to experiment and learn with minimal friction.
- It supports clear role expectations through scorecards and personalized onboarding plans.
- The portal fosters faster ramp-up, reduces manual onboarding effort, and improves developer engagement and retention.
- Ongoing support through community forums, feedback mechanisms, and regular updates ensures continuous improvement and developer satisfaction.

Aim:

Developer Onboarding Portal is to provide a centralized, user-friendly platform that accelerates the integration of new developers into an organization. It empowers developers with quick and structured access to essential tools, documentation, workflows, and self-service capabilities needed to become productive rapidly. The portal seeks to reduce onboarding time, minimize manual support requests, maintain consistent engineering standards, and enhance developer satisfaction and retention. By unifying disparate systems and automating routine tasks, it fosters a seamless onboarding experience that supports continuous learning and collaboration within development teams.

Algorithm:

- **Start the portal system** and initialize all necessary services.
- **Create and maintain a centralized repository** of onboarding resources such as documentation, code repositories, API references, and environment setup guides.
- **Register new developers** by capturing their personal and role-specific information.
- **Assign personalized onboarding plans** based on the developer's role, experience, and team.
- **Automate environment setup workflows**, including access provisioning to tools, repositories, cloud resources, and internal systems.
- **Collect feedback from developers** on onboarding experience for continuous improvement.
- **Generate reports and dashboards** for managers and HR on developer onboarding status and bottlenecks.
- **Update onboarding content and workflows** regularly based on feedback, technology changes, and team needs.
- **Repeat the process for each new developer** ensuring consistent and efficient onboarding.
- This structure ensures a comprehensive, personalized, and efficient onboarding process for developers.

System Requirements:

Software Requirements:

- Operating System:
Windows 10/11, Linux distributions (Ubuntu, CentOS), or macOS
- Programming Languages & Frameworks:
Backend: Java, Python, Node.js, or preferred stack
Frontend: React, Angular, or Vue.js for user interface
- Database:
MongoDB, PostgreSQL, or any scalable database for storing developer profiles, onboarding tasks, and resources
- Web Server:
Apache Tomcat, Nginx, or any Java-supportive or Node.js web server
- IDEs and Tools:
Visual Studio Code, IntelliJ IDEA, Eclipse, or equivalent
- Browsers (for web access):
Chrome, Firefox, Edge, Safari (latest versions recommended)
- Additional Libraries or Services:
Authentication libraries (OAuth, LDAP)
Chatbot or support integration SDKs
Analytics and reporting tools for onboarding metrics

Hardware Requirements:

- Processor:
Minimum: Intel Core i3 or equivalent
Recommended: Intel Core i5 or better
- RAM:
Minimum: 4 GB
Recommended: 8 GB or more for development and testing
- Storage:
Minimum: 50 GB free space
Recommended: SSD for faster application and database performance
- Network:
Reliable internet connection for accessing cloud services and remote repositories
- Development Environment:
Workstations or laptops capable of running required IDEs, databases, and local servers efficiently

Source code:

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

import java.time.Instant;
import java.util.*;
import static com.mongodb.client.model.Filters.eq;

/**
 * Developer Onboarding Portal
 * Java + MongoDB mini project
 */
public class DeveloperOnboardingPortal {

    private static final String MONGO_URI = "mongodb://localhost:27017";
    private static final String DB_NAME = "developerOnboardingDB";

    public static void main(String[] args) {

        try (MongoClient mongoClient = MongoClient.create(MONGO_URI)) {

            MongoDBDatabase db = mongoClient.getDatabase(DB_NAME);
            MongoCollection<Document> devs = db.getCollection("developers");
            MongoCollection<Document> tasks = db.getCollection("tasks");

            Scanner sc = new Scanner(System.in);
            boolean running = true;

            System.out.println("🔗 Developer Onboarding Portal");

            while (running) {
                System.out.println("\nMenu:");
                System.out.println("1. Add Developer");
                System.out.println("2. Add Onboarding Task");
                System.out.println("3. Assign Task to Developer");
                System.out.println("4. Mark Task as Completed");
                System.out.println("5. Show Recommended Tasks");
            }
        }
    }
}
```

```

        System.out.println("6. Exit");
        System.out.print("Enter choice: ");
        int choice = Integer.parseInt(sc.nextLine());

        switch (choice) {
            case 1 -> addDeveloper(sc, devs);
            case 2 -> addTask(sc, tasks);
            case 3 -> assignTask(sc, devs, tasks);
            case 4 -> markTaskCompleted(sc, devs);
            case 5 -> showRecommendations(sc, devs, tasks);
            case 6 -> {
                System.out.println("Exiting Portal...");
                running = false;
            }
            default -> System.out.println("Invalid choice!");
        }
    }

} catch (Exception e) {
    e.printStackTrace();
}

// -----
private static void addDeveloper(Scanner sc, MongoCollection<Document> devs) {
    System.out.print("Enter developer name: ");
    String name = sc.nextLine();
    System.out.print("Enter email: ");
    String email = sc.nextLine();
    System.out.print("Enter role (e.g., Backend, Frontend, DevOps): ");
    String role = sc.nextLine();

    Document dev = new Document("name", name)
        .append("email", email)
        .append("role", role)
        .append("joinedAt", Instant.now())
        .append("assignedTasks", new ArrayList<String>())
        .append("completedTasks", new ArrayList<String>());
}

```

```

    devs.insertOne(dev);
    System.out.println(" Developer added successfully!");
}

```

```

private static void addTask(Scanner sc, MongoCollection<Document> tasks) {
    System.out.print("Enter task ID: ");
    String id = sc.nextLine();
    System.out.print("Enter task title: ");
    String title = sc.nextLine();
    System.out.print("Enter description: ");
    String desc = sc.nextLine();
    System.out.print("Enter role (or leave blank for all): ");
    String role = sc.nextLine();
    System.out.print("Enter priority (1-10): ");
    int priority = Integer.parseInt(sc.nextLine());

```

```

    Document task = new Document("_id", id)
        .append("title", title)
        .append("description", desc)
        .append("role", role.isBlank() ? "All" : role)
        .append("priority", priority)
        .append("createdAt", Instant.now());

```

```

    tasks.insertOne(task);
    System.out.println("Task added successfully!");
}

```

```

private static void assignTask(Scanner sc, MongoCollection<Document> devs,
MongoCollection<Document> tasks) {
    System.out.print("Enter developer email: ");
    String email = sc.nextLine();
    System.out.print("Enter task ID to assign: ");
    String taskId = sc.nextLine();

```

```

    Document dev = devs.find(eq("email", email)).first();
    Document task = tasks.find(eq("_id", taskId)).first();

```

```

    if (dev == null) {
        System.out.println(" Developer not found!");
    }

```

```

        return;
    }
    if (task == null) {
        System.out.println(" Task not found!");
        return;
    }

    List<String> assigned = dev.getList("assignedTasks", String.class, new
ArrayList<>());
    if (!assigned.contains(taskId)) {
        assigned.add(taskId);
        devs.updateOne(eq("email", email), new Document("$set", new
Document("assignedTasks", assigned)));
        System.out.println(" Task assigned successfully!");
    } else {
        System.out.println(" Task already assigned.");
    }
}

private static void markTaskCompleted(Scanner sc, MongoCollection<Document>
devs) {
    System.out.print("Enter developer email: ");
    String email = sc.nextLine();
    System.out.print("Enter completed task ID: ");
    String taskId = sc.nextLine();

    Document dev = devs.find(eq("email", email)).first();
    if (dev == null) {
        System.out.println(" Developer not found!");
        return;
    }

    List<String> completed = dev.getList("completedTasks", String.class, new
ArrayList<>());
    List<String> assigned = dev.getList("assignedTasks", String.class, new
ArrayList<>());

    if (assigned.contains(taskId) && !completed.contains(taskId)) {
        completed.add(taskId);
    }
}

```

```

        devs.updateOne(eq("email", email), new Document("$set", new
Document("completedTasks", completed)));
        System.out.println(" Task marked as completed!");
    } else {
        System.out.println("Task not assigned or already completed.");
    }
}

private static void showRecommendations(Scanner sc,
        MongoCollection<Document> devs,
        MongoCollection<Document> tasks) {
    System.out.print("Enter developer email: ");
    String email = sc.nextLine();

    Document dev = devs.find(eq("email", email)).first();
    if (dev == null) {
        System.out.println(" Developer not found!");
        return;
    }

    String role = dev.getString("role");
    List<String> completed = dev.getList("completedTasks", String.class, new
ArrayList<>());

    List<Document> allTasks = tasks.find().into(new ArrayList<>());
    List<Document> available = new ArrayList<>();

    for (Document t : allTasks) {
        String taskRole = t.getString("role");
        String taskId = t.getString("_id");
        if ((taskRole.equalsIgnoreCase(role) || taskRole.equalsIgnoreCase("All"))
            && !completed.contains(taskId)) {
            available.add(t);
        }
    }

    available.sort((a, b) -> b.getInteger("priority") - a.getInteger("priority"));

    System.out.println("\n📋 Recommended Tasks for " + dev.getString("name") + " ("

```

```
+ role + "):");
    if (available.isEmpty()) {
        System.out.println("🍪 All tasks are completed or no role-specific tasks available.");
    } else {
        for (Document t : available) {
            System.out.printf("%s (Priority %d)\n", t.getString("title"), t.getInteger("priority"));
        }
    }
}
```

Output :

```
C:\Windows\System32\cmd.e  +  v
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" DeveloperOnboardingPortal
Oct 12, 2025 1:53:53 PM com.mongodb.internal.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component
? Developer Onboarding Portal

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 1
Enter developer name: reddybabu
Enter email: reddybabu02@gmail.com
Enter role (e.g., Backend, Frontend, DevOps): frontend
? Developer added successfully!

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 1
Enter developer name: venu
Enter email: venu004@gmail.com
Enter role (e.g., Backend, Frontend, DevOps): backend
? Developer added successfully!

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 1
Enter developer name: sumanth
Enter email: sumanth03@gmail.com
Enter role (e.g., Backend, Frontend, DevOps): devops
? Developer added successfully!
```

```
Enter email: sumanth03@gmail.com
Enter role (e.g., Backend, Frontend, DevOps): devops
? Developer added successfully!

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 5
Enter developer email: venu004@gmail.com

? Recommended Tasks for venu (backend):
? All tasks are completed or no role-specific tasks available.

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 2
Enter task ID: 02
Enter task title: task02
Enter description: compling
Enter role (or leave blank for all): developer
Enter priority (1?10): 2
? Task added successfully!

Menu:
1. Add Developer
2. Add Onboarding Task
3. Assign Task to Developer
4. Mark Task as Completed
5. Show Recommended Tasks
6. Exit
Enter choice: 5
Enter developer email: sumanth03@gmail.com

? Recommended Tasks for sumanth (devops):
? All tasks are completed or no role-specific tasks available.
```

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.

