RealTime Customer Feedback Aggregator

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.

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 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 project report submitted by BUCHAKKAGARI NAGENDRA BABU Regd. No.: 22781A0516, 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

In today's competitive business environment, understanding customer opinions and satisfaction levels is essential for continuous improvement and brand loyalty. However, collecting, organizing, and analyzing customer feedback manually can be time-consuming and prone to errors.

The Real-Time Customer Feedback Aggregator, developed in Java, addresses this challenge by providing an automated solution to manage and evaluate customer responses efficiently. The system allows organizations to store customer feedback in a centralized MongoDB database, enabling seamless data retrieval and real-time analysis. It supports core operations such as adding, updating, deleting, and displaying feedback, while also calculating average ratings to measure overall customer satisfaction.

By integrating a doubly linked list data structure with database operations, the project ensures efficient data management and quick access to customer records. This approach enhances both the performance and reliability of feedback handling. The application demonstrates the practical use of Java, object-oriented programming, and database connectivity in building scalable and interactive business tools.

Ultimately, the system aims to simplify the feedback management process, empower decision-making with real-time insights, and help organizations improve service quality based on authentic customer experiences.

Index

Aim:

The main aim of the Real-Time Customer Feedback Aggregator is to develop an automated system that efficiently collects, manages, and analyzes customer feedback in real-time. The project seeks to simplify the process of handling large volumes of customer responses by integrating a centralized database with an interactive Java-based application.

Algorithm:

1. Start the program.
2. Establish connection to the MongoDB database (customer_feedback_db) and access the collection (feedback).
3. Load existing feedback records from the database into a Doubly Linked List (DLL) for in-memory data management.
4. Display the main menu with the following options:
    1. Add Feedback
    2. Display All Feedback
    3. Search Feedback by I
    4. Update Feedback
    5. Delete Feedback
    6. Calculate Average Rating
    7. Exit
5. Accept the user's choice from the menu.
6. 1) Perform operations based on the user's selection:

   6.1. If choice = 1 (Add Feedback):

   1. Prompt for Feedback ID, Customer Name, Feedback Text, and Rating.
   2. Create a new feedback node and insert it at the end of the DLL.
   3. Insert the same record into the MongoDB collection.

6. If choice = 3 (Search Feedback by ID):
1. Traverse the DLL to locate the matching Feedback ID.
2. If found, display the feedback details; otherwise, show "Feedback not found."
7. If choice = 4 (Update Feedback):
1. If found, ask for new Feedback Text and Rating.
2. Update both the DLL node and the MongoDB record.
3. Display a success message.
8. If choice = 5 (Delete Feedback):
1. Search for the Feedback ID in the DLL.
2. If found, remove the corresponding node by adjusting the links.
3. Delete the same record from the MongoDB collection.
4. Display a success message.
9. If choice = 6 (Calculate Average Rating):
1. Traverse the DLL and calculate the total of all ratings.
2. Count the number of feedback entries.
3. Compute the average rating using the formula:

$$\text{Average Rating} = \frac{\text{Sum of Ratings}}{\text{Number of Entries}}$$

5.Display the calculated average.
6.7. If choice = 7 (Exit):
1.Display an exit message and terminate the program.

10. Repeat steps 4 – 6 until the user selects the Exit option.

11. Close the MongoDB connection and stop

System Requirements:

The Real-Time Customer Feedback Aggregator is a Java-based application integrated with a MongoDB database. It requires both hardware and software components to function efficiently. 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 C 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:

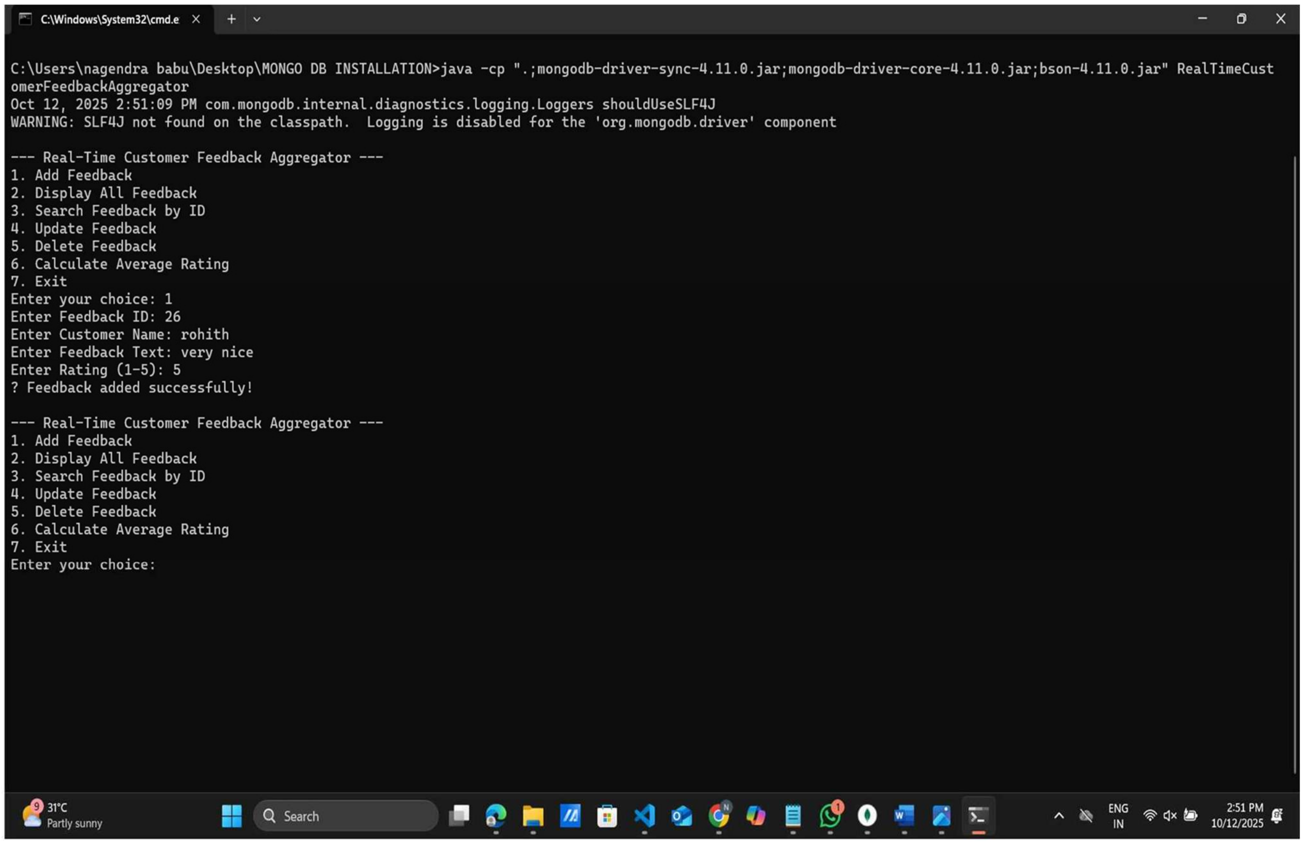
```
import com.mongodb.client.*;
import org.bson.Document;
import java.util.Scanner;
public class RealTimeCustomerFeedbackAggregator {
    // Node for Doubly Linked List
    static class FeedbackNode {
        String feedbackId;
        String customerName;
        String feedbackText;
        int rating;
        FeedbackNode prev, next;
        FeedbackNode(String feedbackId, String customerName, String feedbackText, int rating) {
            this.feedbackId = feedbackId;
            this.customerName = customerName;
            this.feedbackText = feedbackText;
            this.rating = rating;
        }
    }
    // Doubly Linked List with MongoDB Integration
    static class FeedbackDLL {
        FeedbackNode head, tail;
        MongoCollection<Document> collection;
        FeedbackDLL(MongoCollection<Document> collection) {
            this.collection = collection;
        }
        // Load feedback from database into DLL
        void loadFromDatabase() {
            FindIterable<Document> docs = collection.find();
            for (Document doc : docs) {
                String id = doc.getString("feedbackId");
                String name = doc.getString("customerName");
head = tail = newNode;
                } else {
                    tail.next = newNode;
                    newNode.prev = tail;
                    tail = newNode;
                }
            }
        }
        // Safe integer parsing
        private int parseInteger(Object obj) {
            if (obj instanceof Integer) return (Integer) obj;
            else if (obj instanceof String) return Integer.parseInt((String) obj);
            else return 0;
        }
        // Add feedback
        void addFeedback(String feedbackId, String customerName, String feedbackText, int rating) {
            FeedbackNode newNode = new FeedbackNode(feedbackId, customerName, feedbackText,
rating);
            if (head == null) {
                head = tail = newNode;
            } else {
                tail.next = newNode;
                newNode.prev = tail;
                tail = newNode;
            }
            Document doc = new Document("feedbackId", feedbackId)
                    .append("customerName", customerName)
                    .append("feedbackText", feedbackText)
                    .append("rating", rating);

            collection.insertOne(doc);
            System.out.println(" Feedback added successfully!");
        }
        // Display all feedback
        void displayAllFeedback() {
```

```
if (head == null) {
            System.out.println("No feedback found.");
            return;
        }
        FeedbackNode current = head;
        System.out.println("\n--- All Customer Feedback ---");
        while (current != null) {
            System.out.println("ID: " + current.feedbackId +
                    " | Name: " + current.customerName +
                    " | Rating: " + current.rating +
                    "\nFeedback: " + current.feedbackText);
            System.out.println("----------------------------------------");
            current = current.next;
        }
    }
    // Search feedback by ID
    FeedbackNode searchFeedback(String feedbackId) {
        FeedbackNode current = head;
        while (current != null) {
            if (current.feedbackId.equals(feedbackId)) return current;
            current = current.next;
        }
        return null;
    }
    // Update feedback
    void updateFeedback(String feedbackId, String newText, int newRating) {
        FeedbackNode node = searchFeedback(feedbackId);
        if (node != null) {
            node.feedbackText = newText;
            node.rating = newRating;
            collection.updateOne(
                    new Document("feedbackId", feedbackId),
                    new Document("$set", new Document("feedbackText", newText)
                            .append("rating", newRating))
            );
    System.out.println(" Feedback updated successfully!");
        } else {
            System.out.println(" Feedback not found.");
        }
    }
    // Delete feedback
    void deleteFeedback(String feedbackId) {
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;
            collection.deleteOne(new Document("feedbackId", feedbackId));
            System.out.println(" Feedback deleted successfully!");
        } else {
            System.out.println(" Feedback not found.");
        }
    }
    // Calculate average rating (aggregator)
    void calculateAverageRating() {
        if (head == null) {
            System.out.println("No feedback available for aggregation.");
            return;
        }
        double sum = 0;
        int count = 0;
        FeedbackNode current = head;
        while (current != null) {
            sum += current.rating;
            count++;
            current = current.next;
        }
       double avg = sum / count;
        System.out.printf("Average Rating: %.2f (%d feedback entries)%n", avg, count);
    }
}
```

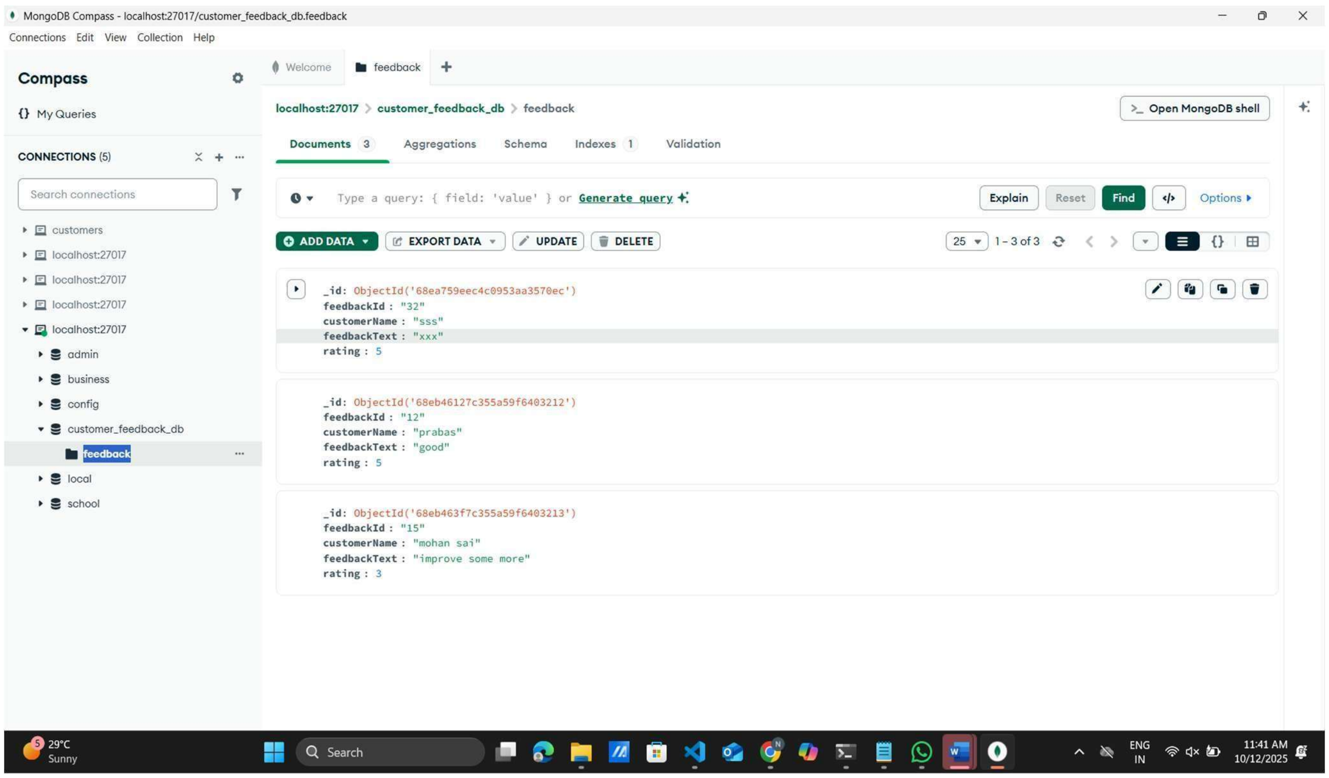
```
  // Main method
   public static void main(String[] args) {
       MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
       MongoDatabase database = mongoClient.getDatabase("customer_feedback_db");
       MongoCollection<Document> collection = database.getCollection("feedback");
       FeedbackDLL feedbackList = new FeedbackDLL(collection);
       feedbackList.loadFromDatabase();
       Scanner sc = new Scanner(System.in);
       int choice;
 do {
           System.out.println("\n--- Real-Time Customer Feedback Aggregator ---");
           System.out.println("1. Add Feedback");
           System.out.println("2. Display All Feedback");
           System.out.println("3. Search Feedback by ID");
           System.out.println("4. Update Feedback");
           System.out.println("5. Delete Feedback");
           System.out.println("6. Calculate Average Rating");
           System.out.println("7. Exit");
           System.out.print("Enter your choice: ");
           while (!sc.hasNextInt()) {
               System.out.print("Invalid input. Enter a number: ");
               sc.next();
           }
           choice = sc.nextInt();
           sc.nextLine();
switch (choice)
 case 1 -> {
                   System.out.print("Enter Feedback ID: ");
                   String id = sc.nextLine();
                   System.out.print("Enter Customer Name: ");
                   String name = sc.nextLine();
                   System.out.print("Enter Feedback Text: ");
                   String feedback = sc.nextLine();
                   System.out.print("Enter Rating (1-5): ");
                   int rating = sc.nextInt();
```

```
    sc.nextLine();
                    feedbackList.addFeedback(id, name, feedback, rating);
                }
                case 2 -> feedbackList.displayAllFeedback();
                case 3 -> {
                    System.out.print("Enter Feedback ID to search: ");
                    String id = sc.nextLine();
                    FeedbackNode f = feedbackList.searchFeedback(id);
                    if (f != null)
                        System.out.println(" Found Feedback:\nCustomer: " + f.customerName +
                                "\nRating: " + f.rating + "\nFeedback: " + f.feedbackText);
                    else
                        System.out.println(" Feedback not found.");
                }
   case 4 -> {
                    System.out.print("Enter Feedback ID to update: ");
                    String id = sc.nextLine();
                    System.out.print("Enter new feedback text: ");
    String newText = sc.nextLine();
                    System.out.print("Enter new rating (1-5): ");
                    int newRating = sc.nextInt();
                    sc.nextLine();
                    feedbackList.updateFeedback(id, newText, newRating);
                }
   case 5 -> {
                    System.out.print("Enter Feedback ID to delete: ");
                    String id = sc.nextLine();
                    feedbackList.deleteFeedback(id);
                }
  case 6 -> feedbackList.calculateAverageRating();
                case 7 -> System.out.println(" Exiting...");
                default -> System.out.println("Invalid choice. Try again!");
            }
        } while (choice != 7);
        mongoClient.close();
        }}
```

Output :



```
C:\Users\nagendra babu\Desktop\MONGO DB INSTALLATION>java -cp ".;mongodb-driver-sync-4.11.0.jar;mongodb-driver-core-4.11.0.jar;bson-4.11.0.jar" RealTimeCust
omerFeedbackAggregator
Oct 12, 2025 2:51:09 PM com.mongodb.internal.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component

--- Real-Time Customer Feedback Aggregator ---
1. Add Feedback
2. Display All Feedback
3. Search Feedback by ID
4. Update Feedback
5. Delete Feedback
6. Calculate Average Rating
7. Exit
Enter your choice: 1
Enter Feedback ID: 26
Enter Customer Name: rohith
Enter Feedback Text: very nice
Enter Rating (1-5): 5
? Feedback added successfully!

--- Real-Time Customer Feedback Aggregator ---
1. Add Feedback
2. Display All Feedback
3. Search Feedback by ID
4. Update Feedback
5. Delete Feedback
6. Calculate Average Rating
7. Exit
Enter your choice:
```



```
_id: ObjectId('68ea759eec4c0953aa3570ec')
feedbackId : "32"
customerName : "sss"
feedbackText : "xxx"
rating : 5

_id: ObjectId('68eb46127c355a59f6403212')
feedbackId : "12"
customerName : "prabas"
feedbackText : "good"
rating : 5

_id: ObjectId('68eb463f7c355a59f6403213')
feedbackId : "15"
customerName : "mohan sai"
feedbackText : "improve some more"
rating : 3
```

Conclusion:

The Real-Time Customer Feedback Aggregator project successfully demonstrates how automation can enhance the efficiency of feedback management in modern organizations. By integrating Java with MongoDB, the system provides a reliable, real-time platform for collecting, storing, and analyzing customer feedback.

The project eliminates the need for manual feedback handling through spreadsheets or paper-based methods, ensuring faster data processing and improved accuracy. It allows administrators to easily perform key operations such as adding, updating, deleting, and viewing feedback, while also calculating the average customer rating to assess overall satisfaction levels.

Using a Doubly Linked List for in-memory data management ensures efficient traversal and quick access to feedback records, making the system both structured and high-performing. The project also highlights the importance of object-oriented programming principles, database connectivity, and modular design in building scalable software solutions.

Overall, this system provides a practical and effective approach for businesses to monitor customer experiences in real time, make informed decisions, and continuously improve their products or services based on genuine customer insights.