**Case Study: Feedback Submission and Management System (CLA-2)**

**1. Objective**

To develop a feedback system for a website where users can submit their feedback and the admin team can manage and review submissions efficiently.

**Enable Real-Time Monitoring**

Provide the admin team with tools to monitor feedback in real-time, allowing faster response to critical issues or bugs.

**Improve User Engagement**

Encourage users to provide feedback by making the process smooth, quick, and user-friendly, showing that their input is valued.

**Categorize Feedback Automatically**

Include categorization (e.g., Bug, Suggestion, Compliment) to help prioritize and sort feedback efficiently.

**Provide Actionable Insights**

Offer basic analytics such as most mentioned keywords, average ratings, or trends over time to help drive business and design decisions.

**Track Feedback Lifecycle**

Allow feedback entries to be marked with statuses like New, In Progress, Resolved, or Ignored, enabling clear tracking from submission to resolution.

**2. Problem Statement**

Websites often lack an organized way to collect and manage user feedback. This leads to:

* Missed opportunities for improvement.
* Poor user experience.
* Delays in issue resolution.

In the digital age, a website serves as a critical point of interaction between businesses and users. While websites are designed to provide content, services, or products, many lack an efficient mechanism to collect, manage, and act on user feedback. This results in a missed opportunity to improve user satisfaction, optimize functionality, and drive continuous improvement.

Most existing systems for feedback collection are either too basic (simple email forms) or overly complex, lacking intuitive design, structured storage, and actionable insights. As a result:

* Users are **hesitant to provide feedback** due to clunky interfaces, lack of acknowledgment, or slow response times.
* Businesses are **unable to filter, prioritize, or respond** to valuable feedback effectively.
* There is **no structured tracking** of reported issues or suggestions, causing them to fall through the cracks.
* Feedback is often **scattered across different platforms** (email, social media, support tickets), making centralized management difficult.
* There is **limited visibility** into recurring user complaints or improvement opportunities, resulting in reactive rather than proactive decision-making.

Without a proper system in place, feedback may become disorganized or ignored entirely, causing user dissatisfaction, loss of trust, and even negative brand perception.

**Real-world Examples of the Problem**

* A customer reports a broken checkout button, but it gets buried in emails and remains unresolved for weeks.
* A user provides a brilliant feature suggestion that never reaches the product team.
* A visitor has a bad experience but finds no easy way to share it — and instead leaves a negative review online.

**Core Problems to Address**

**Lack of a structured feedback form with meaningful input fields.**

**No centralized storage or dashboard to view and manage submissions.**

**No feedback status tracking (e.g., resolved, pending).**

**No analytical tools to make sense of feedback trends.**

**No real-time notifications to alert relevant stakeholders.**

**3. Goals**

* Allow users to submit feedback easily.
* Store feedback in a structured way.
* Enable admins to view, sort, and respond to feedback.

**4. System Architecture**

**Frontend (Client-side):**

* Feedback Form: HTML5 + JavaScript
* Admin Dashboard: HTML5 + JavaScript + optional framework (React, Vue, etc.)

**Backend (Server-side):**

* RESTful API: Handles feedback submission, retrieval, and admin actions.
* Database: Stores feedback entries.

**Optional Enhancements:**

* Email notifications for new feedback.
* Authentication for admin access.
* Export to CSV or Excel.

**5. Tech Stack**

| **Component** | **Technology** |
| --- | --- |
| Frontend | HTML5, CSS3, JavaScript |
| Backend | Node.js + Express or PHP or Python (Flask/Django) |
| Database | MongoDB or MySQL/PostgreSQL |
| Hosting | Firebase, Netlify (frontend); Heroku, Vercel, or AWS (backend) |

**6. Key Features**

**User Side:**

* Form to submit feedback (name, email, rating, comments).
* Optional file upload (screenshot).
* Success message after submission.

**Admin Side:**

* Login system for admin.
* List of all feedback entries with timestamps.
* Filter by rating or status.
* Mark feedback as resolved or important.

**7. Frontend Sample Code (HTML5 Form)**

<form id="feedbackForm" method="POST" action="/submit-feedback">

<label>Name:</label>

<input type="text" name="name" required>

<label>Email:</label>

<input type="email" name="email" required>

<label>Rating:</label>

<input type="range" min="1" max="5" name="rating">

<label>Comments:</label>

<textarea name="comments" required></textarea>

<button type="submit">Submit Feedback</button>

</form>

**8. Backend API Endpoints (Example in Node.js/Express)**

// POST: Submit feedback

app.post("/submit-feedback", async (req, res) => {

const { name, email, rating, comments } = req.body;

await db.collection("feedback").insertOne({

name, email, rating, comments, timestamp: new Date()

});

res.send("Feedback submitted successfully!");

});

// GET: Admin fetch feedback

app.get("/admin/feedback", async (req, res) => {

const feedback = await db.collection("feedback").find().toArray();

res.json(feedback);

});

9. Database Schema (MongoDB Example)

{

"name": "Jane Doe",

"email": "jane@example.com",

"rating": 4,

"comments": "Great experience!",

"timestamp": "2025-04-14T12:00:00Z",

"status": "unresolved"

}

**10. Security and Validation**

* Input validation (client and server side).
* Rate limiting to prevent spam.
* CAPTCHA (e.g., Google reCAPTCHA).
* Authentication for admin dashboard.

**11. Optional Features**

* Email notification to admin.
* Visual analytics on feedback.
* Admin reply system with email response.
* Progressive Web App version of the system.

**12. Benefits**

* Improves user trust and satisfaction.
* Helps identify bugs or UX issues.
* Supports data-driven website improvements.

**13. Future Scope**

* AI-driven feedback analysis (sentiment analysis).
* Integration with project management tools like Trello/Jira.
* Slack/Discord notifications for team alerts.