IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

Digital Complaint Box System

Jayant Vijay Bhagat[1], Sankalp Santosh Bhosle[2], Dipak Ramesh Chavan[3], Prof. P. S. Shekhar[4]

Department of Computer Engineering,

A. C. Patil College of Engineering, Affiliated to University of Mumbai, Kharghar, Navi Mumbai, India[1-3]

Guide, Department of Computer Engineering,

A. C. Patil College of Engineering, Affiliated to University of Mumbai, Kharghar, Navi Mumbai, India[4]

Abstract: Most colleges, offices, and institutions still rely on old complaint-handling methods — paper forms, suggestion boxes, and emails — that are slow, hard to track, and easy to ignore. This paper introduces the Digital Complaint Box System (DCBS), a simple web-based platform that solves these problems. Any user can open the website, fill in a short form, and submit a complaint in seconds. If they do not want to share their identity, they can do so anonymously. Admins get a secure, private dashboard where they can see every complaint, update its progress (Pending, In Progress, or Resolved), and manage everything from one place. The moment an admin changes a status, the user sees the update on their screen automatically — no refreshing needed. The system was built using HTML, CSS, and JavaScript for the front end; Node.js to handle the server logic; and Google Firebase to store data and manage logins. We tested it across eight real-world scenarios — submitting complaints, logging in as admin, going anonymous, checking on a mobile phone, and watching real-time updates — and every test passed. Compared to paper systems, emails, or expensive enterprise tools, DCBS is faster, more transparent, and free to deploy. This paper walks through why the system was needed, how it was designed and built, what the test results showed, and what we plan to add in the future.

Keywords: Complaint Management System; Web Application; Firebase Realtime Database; Node.js; Grievance Resolution; Three-Tier Architecture; Institutional Transparency.

I. INTRODUCTION

A. Background — Why This System Was Needed

Every organization needs a way for people to speak up when something is wrong. Whether it is a student with a problem in class, an employee dealing with a workplace issue, or a citizen unhappy with a public service — complaints are important. When they are handled well, trust grows and services improve. When they are ignored, people feel unheard and the same problems come back again and again.

For a long time, most institutions handled complaints by placing a physical suggestion box somewhere in the building, or by asking people to send an email or speak to someone in person. These methods have worked in the past, but today they just are not good enough. A paper complaint can be lost, ignored, or thrown away. An email gets buried. There is no way to check what happened after you submitted something. And nobody is held accountable when nothing gets done.

With the internet and free cloud tools now widely available, it is completely realistic to build a digital system that fixes all of these problems — one that works on any device, keeps complaints safe, lets users track progress, and gives admins everything they need in one dashboard. That is what the Digital Complaint Box System (DCBS) does. It uses HTML, CSS, JavaScript, Node.js, and Google Firebase — all free and open-source — so any institution can run it at no cost.

B. The Problem with Traditional Complaint Systems

Despite all the technology available today, many institutions still handle complaints in ways that cause the following problems:

- There is no central place to store complaints — they end up scattered across papers, emails, and memory.
- Users have no way to check what happened to their complaint after they submitted it.
- The whole process is hidden from view — people do not know who is handling it or what decisions are being made.
- Everything is done manually, so complaints sit unprocessed for days or weeks.
- Without written records, admins can ignore complaints and face no consequences.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

- There is no way to look back at patterns and figure out which problems keep repeating.

These issues make it clear why a proper digital system is needed — not just for speed, but for fairness and accountability.

C. What This System Aims to Do

The Digital Complaint Box System was built with these specific goals:

1. Create a website anyone can use to submit a complaint — no technical skills needed.
2. Store every complaint safely in a cloud database (Google Firebase) so nothing is ever lost.
3. Build a password-protected admin panel where staff can view, manage, and resolve complaints.
4. Let users track their complaint through three stages: Pending, In Progress, and Resolved.
5. Allow people to submit anonymously, so they are not afraid to report sensitive issues.
6. Keep everything secure so only the right people can access complaint data.

D. How This Paper Is Organized

Section II looks at existing complaint systems and what they lack. Section III explains how DCBS was designed. Section IV goes into the technical details of how it was built. Section V shows the testing results. Section VI wraps up with a conclusion, and Section VII talks about future improvements.

II. LITERATURE REVIEW

A. Old-Fashioned Paper Complaint Boxes

The most common way institutions have handled complaints is the physical suggestion box — a locked box in a corridor where people drop handwritten notes. Patr and Viaene [1] found that while these are easy to set up, they fail at almost everything else. Notes get lost or thrown away. There is no way to follow up. Nobody knows whether their complaint was ever read. And since there are no records, it is impossible to spot patterns or understand which problems keep happening.

B. Email-Based Complaint Handling

Sending complaints by email is better than paper because at least there is a written record. But Gronroos [2] points out that email was never designed for this kind of work. When complaints pile up in an inbox, there is no structure — no categories, no priorities, no status tracking. Multiple people might be cc'd on the same complaint, or nobody might be. From the user's side, they send an email and then just wait, often never hearing back at all.

C. Online Complaint Portals

A step further, some organizations and governments have built web portals specifically for complaints. Saxena et al. [3] studied one such portal used by a city authority and found that going digital reduced average resolution time by around 40% compared to paper systems — a significant gain. However, many of these early portals did not update in real time. Data refreshed in batches, so users might wait hours before seeing a status change. DCBS fixes this by using Firebase's live database, where updates appear the instant they happen.

Kumbhar and Lokhande [4] built a college complaint system using PHP and MySQL. It worked, but it needed a dedicated web server and someone with database skills to manage it. That is a real barrier for smaller institutions.
DCBS avoids that entirely — Firebase is managed by Google in the cloud, so there is nothing to install or maintain.

D. Enterprise Tools Like Salesforce and Zendesk

At the high end, tools like Salesforce Service Cloud and Zendesk offer advanced complaint management with automatic ticket routing, AI analysis, and detailed reports [5]. These are powerful, but they are also expensive and complex. Singh and Sharma [6] compared several such platforms and found that the total cost of running them is often far beyond what a typical college or small office can afford. DCBS delivers the most important features — submission, tracking, dashboard, real-time updates — at zero cost.

E. Mobile Complaint Apps

More recently, researchers have started building complaint apps for smartphones. Malhotra and Gupta [7] created a mobile app for a college using React Native and Firebase, and found that students were more willing to report issues when they could

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

do it from their phone. Easy access matters. DCBS does not have a dedicated mobile app yet, but the website is fully responsive — it works well on phones and tablets through a browser. A proper app is on the roadmap.

F. The Three Gaps That DCBS Fills

After reviewing all of these systems, three problems kept coming up: (1) cheap solutions rarely offer real-time updates; (2) solutions with real-time features are usually expensive or complicated to set up; and (3) almost no system supports anonymous submissions in a structured way, even though anonymity is often the only thing that makes users feel safe enough to speak up. DCBS was specifically designed to solve all three of these problems at once.

Table I compares the different types of complaint systems side by side.

Table I: Comparison of Complaint Management Approaches

Feature	Manual System	Email-Based	Web Forms	Our System (DCBS)
Complaint Tracking	Not Available	Limited	Not Available	Full Real-Time
Processing Speed	Very Slow	Moderate	Fast	Very Fast
Transparency	Very Low	Medium	Low	High
Data Storage	Paper / None	Unstructured	Moderate	Cloud (Firebase)
Accessibility	On-Site Only	Moderate	High	Any Device, Anytime
Admin Dashboard	Not Available	Not Available	Limited	Comprehensive
Anonymous Submission	Not Available	Not Available	Rarely	Fully Supported
Deployment Cost	Very Low	Low	Medium	Free (Firebase Tier)
Scalability	Very Low	Low	Moderate	High

III. SYSTEM DESIGN

A. The Big Picture

DCBS is split into three layers: the front end (what you see on screen), the back end (the logic running behind the scenes), and the database (where everything is stored). Keeping these three layers separate makes the system easier to understand, test, and update over time.

There are two kinds of people who use the system: regular users who submit complaints, and admins who manage them. A user fills out a form and submits it. The complaint goes straight into Firebase and appears on the admin's dashboard immediately. The admin reviews it, takes action, and updates the status. As soon as they do, the user's tracking page reflects the change — no page refresh, no waiting.

B. The Three Layers Explained

Front End (what users interact with): Built with HTML, CSS, and JavaScript. This is the complaint form, the status tracker, and the admin dashboard. CSS makes sure it looks right on any screen size — phone, tablet, or desktop.

Back End (the logic layer): Built with Node.js and Express.js. This part receives data from the front end, checks that everything is valid, and passes it to Firebase. It also enforces security — making sure only logged-in admins can access sensitive data.

Database (where data lives): Google Firebase Realtime Database. All complaints are stored here in a simple, organised structure. When anything changes, Firebase instantly notifies all connected screens — that is how realtime updates work.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

C. The Four Modules

User Module: Everything the regular user sees and does — filling in the form, choosing anonymous, getting a confirmation message.

Admin Module: The password-protected dashboard. Admins can see all complaints, filter them, and change their status step by step.

Database Module: Handles all reading and writing to Firebase — saving new complaints, fetching them for the admin, updating statuses.

Tracking Module: Uses Firebase's live listener feature to push any status change to the user's screen the moment the admin makes it.

D. How a Complaint Travels Through the System — Step by Step

7. The user opens the website, fills in the form (name, category, description), and optionally ticks 'Submit Anonymously'.
8. JavaScript checks the form immediately. If anything is wrong or missing, an error message appears right next to that field.
9. Once everything is valid, the data is sent to the Node.js server via a secure API call.
10. The server does its own checks, then writes the complaint to Firebase — automatically adding a unique ID and the submission timestamp.
11. The admin's dashboard updates instantly, showing the new complaint without any reload.
12. The admin reviews the complaint, and moves its status from Pending → In Progress → Resolved.
13. The user's tracking page updates in real time — they can see the change as soon as it happens.

IV. IMPLEMENTATION DETAILS

A. Technologies Used and Why Each Was Chosen

HTML5: The skeleton of every page. It creates the structure — forms, buttons, headings, layout containers. Semantic HTML tags were used throughout to make the site accessible to screen readers.

CSS3: Handles all the visual design — colours, spacing, fonts, and responsive layout. CSS Flexbox and Grid make sure the interface works correctly on any screen, from a 375px phone to a 1440px desktop monitor.

JavaScript (ES6+): Adds interactivity. It validates forms before submission, connects to Firebase to listen for live updates, and handles everything that changes on screen without needing a page reload.

Node.js with Express.js: Runs the server. It handles three main tasks: accepting new complaints (POST), returning all complaints to the admin (GET), and updating a complaint's status (PATCH). It also blocks any request that does not come with a valid admin login token.

Google Firebase: Two jobs in one platform. The Realtime Database stores all complaint records and pushes live updates to connected clients. Firebase Authentication manages admin logins securely, and Firebase Security Rules make sure users can only see their own data.

npm: Manages the project's libraries — Firebase Admin SDK, Express, dotenv (for keeping secret keys out of the code), and cors.

B. How the Front End Was Built

The front end has two main pages: the Complaint Submission Page and the Complaint Tracking Page.

On the Submission Page, users see a clean form with three fields: their name (which disappears if they choose to go anonymous), a dropdown to pick the complaint type (Academic, Infrastructure, Administrative, or Other), and a text box where they describe the issue in detail. A live character counter shows how much of the 1000-character limit they have used. Before anything is sent to the server, JavaScript checks the form on the spot. If the description is empty, a clear error message appears right below it. If an email address is given but does not look right, that gets flagged too. Errors are shown in a friendly, non-alarming way that follows standard web accessibility guidelines.

On the Tracking Page, users who gave their email when submitting can see the live status of their complaint. This page connects to Firebase through a listener — meaning the status shown there updates automatically whenever the admin makes a change.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

C. How the Back End Was Built

The Node.js server handles three kinds of requests:

POST /api/complaints — receiving new complaints: When a user submits the form, the data arrives here. The server validates it, saves it to Firebase with an auto-generated ID, and sends back a confirmation.

GET /api/complaints — fetching all complaints for the admin: Only accessible with a valid Firebase login token. Returns all complaints, newest first.

PATCH /api/complaints/:id — updating a complaint's status: The admin sends the new status, the server checks it is a valid option, and saves the change to Firebase.

Any request to the admin endpoints without a proper login token is immediately rejected with a '401 Unauthorized' error. The token is generated by Firebase Authentication when the admin logs in, and is checked on every single request.

D. How the Database Is Structured

Every complaint is stored as a record in Firebase. Table II below shows exactly what each record contains.

Table II: Structure of Each Complaint Record

Field	Type	Rule	What It Means in Plain English
complaintId	String (auto-ID)	Unique, auto-generated	Every complaint gets its own ID the moment it is saved — no two are the same
userName	String	Required	The name of the person who filed the complaint (hidden if anonymous)
userEmail	String	Optional	An email address the admin can use to follow up, if the user provided one
Field	**Type**	**Rule**	**What It Means in Plain English**
category	String	Required	Which type of complaint it is — Academic, Infrastructure, Administrative, or Other
description	Text	Required, max 1000 chars	The full details of the complaint, written by the user in their own words
status	String	Default: Pending	Where the complaint currently stands — Pending, In Progress, or Resolved
submittedAt	Timestamp	Auto-generated	Exact date and time when the complaint was submitted — set automatically by the system
resolvedAt	Timestamp	Optional	Filled in when the admin marks the complaint as resolved
isAnonymous	Boolean	Default: false	Set to true when the user chose to hide their name and identity

Firebase Security Rules make sure that regular users can only write complaints and read their own records. Admins can read and update everything. These rules run on Firebase's own servers, so they cannot be bypassed even if someone tries to manipulate the front end.

E. Security

Security was handled carefully at every level. Admin accounts are protected by Firebase Authentication, which issues short-lived tokens that automatically expire. All data travelling between the browser and the server is encrypted over HTTPS. Input from users is cleaned on both the client side and the server side to prevent any code from being injected into the database. And Firebase's own security rules act as a final firewall — even if someone found a gap in the front end, they still could not read other people's complaints.

V. RESULTS AND DISCUSSION

A. How Testing Was Done

The system went through three rounds of testing. First, each part was tested on its own to make sure it worked correctly in isolation (unit testing). Then all parts were connected and tested together to check that data flowed correctly from the user's

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154247

browser all the way to Firebase and back (integration testing). Finally, the full system was tested using realistic end-to-end scenarios — the kinds of things real users would actually do (system testing).
Eight test cases were defined. Each one had a clear goal, a specific input, a known expected result, and a recorded outcome. Table III shows the complete list.

Table III: Test Cases and Results

ID	What Was Tested	Input	Expected Result	Outcome
TC-01	Submitting a valid complaint	User fills name, category, description	Saved to Firebase; success message shown	PASS
ID	**What Was Tested**	**Input**	**Expected Result**	**Outcome**
TC-02	Submitting with a blank description	User leaves description empty and clicks Submit	Red error: 'Description cannot be empty'	PASS
TC-03	Admin logging in	Admin enters correct email + password	Redirected to admin dashboard	PASS
TC-04	Admin viewing all complaints	Admin opens dashboard	All complaints load in real time	PASS
TC-05	Admin changing a complaint's status	Admin clicks 'Mark as Resolved'	Status updates in Firebase and on user's screen	PASS
TC-06	Submitting anonymously	User ticks 'Anonymous' box and submits	Complaint saved — no name or identity attached	PASS
TC-07	Using the site on a phone screen (375px)	Website opened on mobile browser	Layout adjusts cleanly — nothing overflows	PASS
TC-08	Real-time status update	Admin resolves while user watches tracking page	Status changes instantly — no page refresh needed	PASS

B. What the Results Showed

All eight tests passed. Beyond pass/fail, some useful performance numbers were also recorded:

- After a user submits a complaint, it is saved to Firebase in about 340 milliseconds on a standard broadband connection. That is less than half a second — users will not notice any delay.
- When an admin updates a status, it appears on the user's tracking page in about 180 milliseconds. This is possible because Firebase keeps an open WebSocket connection instead of making repeated server requests. • The system was tested with 20 users active at the same time. There was no slowdown and no data errors.
- The layout was checked at three screen widths — 375px (phone), 768px (tablet), and 1440px (desktop). It looked and worked correctly on all three.

C. How DCBS Compares to Traditional Approaches

Transparency: Users always know what stage their complaint is at. Traditional systems give users nothing after submission.
Accountability: Every complaint has a timestamp and a permanent record. Admins cannot quietly ignore something because it is documented the moment it is submitted.
Accessibility: Anyone with a browser and an internet connection can use it — no office visit, no business hours, no installation.
Cost: It runs entirely on Firebase's free tier. There are no licensing fees, no server bills, and no maintenance contracts.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154247

Anonymity: Users who are worried about speaking up can still do so without revealing who they are. No traditional system handles this in a structured, consistent way.

D. Current Limitations

DCBS does not yet have a mobile app, so users cannot receive push notifications — they need to check the website manually. The admin panel does not include charts or trend analysis yet, so spotting patterns requires manual review. The system is also English-only at this stage, which could be a barrier for some users. All of these are addressed in the next section.

VI. CONCLUSION

The Digital Complaint Box System was built to solve a problem that affects almost every institution — complaints get lost, ignored, or forgotten because the tools used to handle them simply are not good enough. By building a simple, free-to-deploy web platform, DCBS makes it easy for anyone to speak up and easy for admins to actually do something about it.

The system uses entirely free tools — HTML, CSS, JavaScript, Node.js, and Firebase — so any institution can adopt it without spending money. The three-layer structure keeps the code clean and easy to update. Firebase's realtime database means that status changes appear on users' screens the moment they happen, not after a delay.

Testing confirmed that the system works reliably in every scenario we tried, handles multiple users at once, works well on mobile screens, and keeps data secure. Compared to paper systems, email, or expensive enterprise software, DCBS is faster, more honest, and more accountable.

More than anything, this project shows that a well-thought-out digital tool does not have to be expensive or complicated to make a real difference in how institutions listen to and respond to the people they serve.

VII. FUTURE SCOPE

DCBS is a solid, working system — but there is a lot more it can become. Here are the most important improvements planned:

Mobile App: A proper Android and iOS app would allow push notifications, so users get an instant alert when their complaint status changes without needing to open the website.

AI-Based Complaint Sorting: By connecting an NLP (Natural Language Processing) model, the system could automatically read incoming complaints and sort them by type and urgency — saving admins a lot of manual work.

Analytics Dashboard for Admins: Charts showing which complaint types are most common, how long resolutions take on average, and whether things are improving would help institutions make better decisions.

Multi-Language Support: Adding Hindi, Marathi, and other regional languages would make DCBS accessible to far more users across India.

Automatic Escalation: If a complaint is not resolved within a set time limit, the system could automatically alert a higher authority — so nothing stays stuck indefinitely.

Blockchain Audit Trail: For high-stakes use cases (like government or legal settings), recording every action on a blockchain would create a tamper-proof history that no one can alter.

REFERENCES

[1]. A. Patr and S. Viaene, "Complaint management: A theoretical perspective on handling customer dissatisfaction in service organizations," Journal of Service Management, vol. 21, no. 4, pp. 502–522, 2010.

[2]. C. Gronroos, "Service recovery: The case of complaint management," International Journal of Service Industry Management, vol. 3, no. 3, pp. 37–51, 1992.

[3]. S. Saxena, A. Kumar, and R. Mishra, "Design and implementation of a web-based citizen grievance redressal portal for municipal corporations," International Journal of Computer Applications, vol. 150, no. 11, pp. 14–18, 2016.

[4]. V. Kumbhar and S. Lokhande, "Online complaint management system for educational institutions using PHP and MySQL," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 4, pp. 215–220, 2017.

[5]. Salesforce Inc., "Salesforce Service Cloud: Complaint management and customer service automation," Salesforce Technical Documentation, San Francisco, CA, 2023. [Online]. Available: https://www.salesforce.com/products/servicecloud/

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154247

[6]. R. Singh and A. Sharma, "Comparative evaluation of enterprise complaint management systems for higher education institutions," Journal of Educational Technology and Society, vol. 20, no. 2, pp. 88–101, 2017.
[7]. P. Malhotra and S. Gupta, "A hybrid mobile application for institutional grievance management using React Native and Firebase," in Proc. IEEE International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 487–493.
[8]. Google Firebase, "Firebase Realtime Database documentation," Google LLC, 2024. [Online]. Available: https://firebase.google.com/docs/database
[9]. OpenJS Foundation, "Node.js documentation," 2024. [Online]. Available: https://nodejs.org/en/docs **[10]** I. Sommerville, Software Engineering, 10th ed. Pearson Education, 2015.
[10]. R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.
[11]. World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.1," W3C Recommendation, June 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/
[12]. Mozilla Developer Network (MDN), "Web development documentation," Mozilla Foundation, 2024. [Online]. Available: https://developer.mozilla.org/

