Mr. S. Krishna Reddy, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Complaint & Grievance Redressal System with Duplicate Detection and Smart Escalation

Mr. S. Krishna Reddy, P.Harika, N. Uma Tejaswini, J. Veera Shiva Kumar, N.Kevin
Department of Electronics and Communication Engineering, Sasi Institute of Technology and Engineering

Abstract- The Complaint Grievance Redressal System (CGRS) was developed to improve the traditional method of managing student complaints by incorporating artificial intelligence for smarter complaint handling. In many institutions, the existing complaint process is manual, time-consuming, and difficult to track. To overcome these issues, the proposed system introduces semantic duplicate detection, automatic complaint routing, and priority-based escalation. The backend runs on Node.js and Express, with MongoDB handling data storage. When a student submits a complaint, the system uses Google Gemini 1.5 Flash to compare it against existing complaints in the database. If the similarity score hits 80 or above, the new submission is logged as a duplicate and the original complaint's duplicate count goes up by one. Once that count reaches three, the complaint is automatically bumped to high priority so it gets attention faster. Routing is automatic too. Based on the department and complaint type the student selects, the complaint goes straight to the right person either the Head of Department or the Training and Placement Officer. Students also get email notifications whenever their complaint status changes or gets resolved, so they're not left guessing. Access is split across four roles: Student, HOD, TPO, and Admin. Authentication uses JWT tokens, and each role only sees and does what it's supposed to nothing more. This paper presents the overall system architecture, artificial intelligence integration methodology, duplicate detection algorithm, and system performance evaluation.

Keywords— Grievance Redressal, Duplicate Detection, AI, Gemini Flash, Semantic Similarity, Escalation, Node.js, MongoDB, JWT, REST API.

I. INTRODUCTION

Colleges deal with a steady stream of student complaints academics, placements, hostel conditions, administrative issues. Handling them well matters, both for students who expect to be heard and for institutions that want to stay on top of recurring problems.

Most existing systems aren't built for that. Paper forms, email threads, basic web portals they get the complaint recorded, but not much else. No intelligent routing, no prioritization, no way to tell when fifty students are reporting the same broken thing.

That last part is a real problem. When multiple students raise the same issue, traditional systems treat each submission as its own ticket. Administrators end up with a pile of near-identical complaints, each demanding individual attention, while the actual underlying problem gets lost in the noise. The workload goes up, the focus gets scattered, and issues that genuinely need urgent attention don't always get it.

The standout feature of the system is how it handles duplicate complaints. Rather than matching keywords, Gemini 1.5 Flash looks at what a complaint actually means and compares that against existing open complaints in the same category. If something's already been reported, the system catches it, scores the similarity, and links it back to the original.

That linking does more than just clean up the database. Once three or more complaints point to

the same underlying issue, the system automatically bumps the priority to HIGH and marks it as ESCALATED. So if a dozen students are all frustrated about the same thing but describing it differently, the administration sees it as one urgent problem not a dozen separate tickets sitting in a queue.

Students get email notifications through Node Mailer when their complaint is reviewed or resolved, so there's at least some visibility into what's happening on the other end.

Taken together, the system cuts down on redundant submissions, surfaces the issues that actually need urgent attention, and keeps students in the loop which is more than most college complaint systems manage.

II. LITERATURE SURVEY

Grievance redressal systems have come a long way, with researchers tackling the problem from both technical and administrative angles — automation, transparency, system design, user experience. The common thread is finding better ways to handle the gap between users raising issues and organizations actually responding to them.

A 2024 study in the ISREM Journal, "Beyond Complaints: Strategic Analysis of College Grievance Redressal System," looked at the problem from the policy side. It made a solid case for institutional accountability and structured complaint handling processes. What it didn't get into was how to actually build any of it — no technical implementation, no system architecture, just the administrative framework.

Sanay Shah et al. (2022) proposed a smart grievance redressal system that uses Natural Language Processing (NLP) to detect and filter inappropriate language in complaints. This approach helps maintain the quality of complaint submissions and reduces misuse of the system. However, the system's performance depends heavily on the accuracy of the NLP model and its ability to support multiple languages and contexts.

Kiran M. Jadhav et al. (2022) developed an online grievance redressal system that allows users to register complaints and track their status online. Their system reduced manual work and improved transparency in complaint handling. Although the system worked effectively for basic complaint management, it lacked features for handling large-scale data and advanced analytics.

Owen Maytrio Phratama et al. (2024) proposed a web-based grievance system developed using ASP.NET and SQL Server to replace traditional manual complaint systems. The system improved usability and transparency in complaint processing. However, the system followed the Waterfall development model, which made it less flexible when new features needed to be added or changes were required during development.

Ms. Pooja B. Nirgude et al. (2023) developed an online student grievance system with an admin dashboard for managing complaints efficiently. Their system helped centralize complaint handling and improved communication between students and administrators. However, the system faced scalability limitations and did not include advanced reporting or analytics features.

Chetana R. Shivanagi et al. (2024) introduced “ResolveU: Student Complaint Management System,” developed using Python (Flask) and MySQL. Their system included secure login functionality, a structured workflow, and proper database management. However, the research did not clearly explain how the system would perform in large-scale environments with a large number of users and complaints.

Yuvraj Narayan Mishra et al. (2024) proposed a flexible grievance redressal framework developed using PHP. Their system included features such as complaint tracking, escalation mechanisms, and structured workflows, making the system more organized and transparent. However, the system lacked real-world testing and performance validation.

Prof. R. K. Sahare et al. (2024) developed an Online Grievance Redressal System using PHP and MySQL, focusing on complaint categorization and role-based user management. The system improved data organization and transparency in complaint handling, but it did not address scalability issues or include advanced analytical features.

Indhitya R. Padiku et al. (2024) proposed a web and mobile-based grievance system using prototype development methodology along with MAUT (Multi-Attribute Utility Theory) to prioritize complaints. This approach improved decision-making and helped prioritize important complaints efficiently. However, the study did not fully explore real-time implementation and practical deployment challenges.

Overall, the literature shows that many grievance redressal systems focus on online complaint submission, tracking, and administrative management. However, most existing systems lack intelligent duplicate detection, automated escalation, and AI-based complaint analysis. These limitations motivated the development of the proposed Complaint Grievance Redressal System (CGRS), which integrates artificial intelligence for semantic duplicate detection, automatic routing, and priority-based escalation.

III. PROBLEMS IN EXISTING SYSTEMS

Traditional grievance redressal systems face several limitations that reduce their efficiency and effectiveness in handling student complaints. The major problems observed in existing systems are described below.

1. Lack of AI Integration
Most existing grievance systems do not use artificial intelligence technologies, and the complaint handling process depends mainly on manual work. Because of this, identifying duplicate complaints, assigning complaints to the correct authority, and prioritizing issues becomes time-consuming and less efficient. The absence of automation also increases the workload for administrators.

2. No Performance Monitoring of Authorities
In many existing systems, there is no proper mechanism to monitor the performance of authorities such as Heads of Departments (HODs) or higher officials. The system does not maintain clear records showing how many complaints were resolved, pending, or escalated by each authority. This lack of performance tracking reduces accountability and makes it difficult to evaluate administrative efficiency.

3. Missing Notification System
Many traditional grievance platforms do not include an automated notification system. Students are not informed when their complaint status changes or when the complaint is resolved. This creates a communication gap between the system and users, and students may need to repeatedly check the portal for updates.

4. No Proper Complaint Tracking
Existing systems often do not provide a proper real-time complaint tracking feature. Users cannot easily track the progress of their complaints, such as whether the complaint is under review, in progress, resolved, or escalated. This lack oftransparency may cause uncertainty and dissatisfaction among users.

5. Lack of Priority-Based Handling
Many grievance systems treat all complaints in the same way and do not differentiate between normal complaints and urgent complaints. As a result, critical or system-wide issues may not receive immediate attention, which can delay problem resolution and affect many students.

IV. PROPOSED SYSTEM AND ARCHITECTURE

AI-GRS is a RESTful backend built on Node.js with Express v5 and MongoDB, using Mongoose as the ODM. The architecture follows a layered MVC pattern — routes, controllers, models, middleware, and services each live in their own layer, which keeps the codebase manageable as it grows.

The system has four roles. Students submit complaints and check on their status. HODs handle department-level issues, TPOs deal with placement-related ones, and the Admin has full visibility across everything — including the ability to reassign complaints when needed. Access is controlled through JWT authentication, with role-based middleware making sure each user only reaches the endpoints they're supposed to.

The core of the system is the AI duplicate detection. When a student submits a complaint, the system pulls all open complaints in the same category and sends them to Gemini 1.5 Flash along with the new one. The model comes back with a JSON object — three fields: IsDuplicate, originalComplaintId, and similarityScore on a scale of 0 to 100. If that score lands at 80 or above, the complaint is treated as a duplicate and handled accordingly.

Figure 1 represents the overall system architecture and workflow. The complaint submission process follows these steps: input validation → identify the appropriate authority → retrieve existing complaints → call the AI duplicate detection service → check duplicate result → store complaint or repeated complaint record → update escalation details → send response to the user.

STUDENT	→	**POST /API/complaints**	→	**AI Duplicate Check (Gemini)**
		↓ (duplicate detected)		↓ (unique complaint)
		Repeated Complaint saved to DB		New Complaint created
		Original: duplicate Count++		Assigned to HOD / TPO
		If count ≥ 3 → ESCALATED + HIGH		Status: IN_PROGRESS
HOD/TPO	→	**PUT / API /complaints/respond/:id**	→	**Email notification sent to Student**

Fig. 1. AI-GRS Complaint Processing Pipeline

V. SYSTEM MODULES

The system is broken into focused modules, each handling a specific piece of the overall workflow — from managing user accounts to detecting duplicates and tracking resolutions.

1. User Management

This module covers registration, login, password hashing with bcrypt, and JWT token generation. Each user record stores the basics name, email, encrypted password, role, and department. Roles can be Admin, Student, HOD, or TPO. Department is only required for HOD and TPO accounts since it doesn't apply to Admins or Students.

2. Complaint Submission

Students fill in a few fields when submitting a complaint: category, subject, description, and the target authority — either HOD or TPO. Department information is included when relevant. Once submitted, the complaint moves through the rest of the system from there. The controller validates all inputs, identifies the correct complaint assignee from the database, and then calls the AI duplicate detection service before saving the complaint in the database.

3. AI Duplicate Detection Service

This service is implemented in the aiservice.js file. It generates a structured prompt containing the new complaint along with all existing open complaints in the same category. The Gemini API processes the data and returns a similarity result in JSON format. The system safely extracts the JSON using regex and performs type conversion to avoid errors caused by incorrect AI responses. If any error occurs during AI processing, the system automatically marks the complaint as non-duplicate to ensure that valid complaints are not blocked.

4. Escalation Engine

The escalation module activates when duplicate complaints are detected. Each duplicate gets logged in the Repeated Complaints collection, and the original complaint's duplicate count goes up by one. Once that count hits three, the complaint is automatically marked as ESCALATED and bumped to HIGH priority — making it easier for admins to spot recurring issues without having to dig through everything manually.

5. Complaint Resolution

HODs and TPOs can view the complaints assigned to them through the complaints endpoint, then submit responses and update the status once they've acted on it. When a complaint is marked resolved, the student gets an email notification through Nodemailer so they know it's been dealt with.

6. Admin Module

The Admin module provides full system control. Administrators can view all complaints, reassign complaints to different authorities, and manage user accounts. The admin dashboard also displays system statistics such as total complaints, open complaints, in-progress complaints, escalated complaints, resolved complaints, and closed complaints.

7. Complaint Status Module

This module allows students to track the progress of their complaints. The complaint status follows a predefined workflow:

OPEN → IN_PROGRESS → ESCALATED → RESOLVED → CLOSED.

This status flow ensures proper tracking and management of complaints throughout their lifecycle

Table1. Complaint Schema Field Reference

Field	Type	Description
Student Id	ObjectId (ref User)	Complaint submitter
category	String	Complaint category (fees, hostel, etc.)
subject	String (required)	Brief complaint title
description	String (required)	Full complaint text
target	Enum: HOD | TPO	Intended recipient authority
Assigned To	ObjectId (ref User)	Auto-resolved HOD or TPO
status	Enum (5 states)	OPEN / IN_PROGRESS / ESCALATED / RESOLVED / CLOSED
priority	LOW | MEDIUM | HIGH	Default MEDIUM; set HIGH on escalation
Duplicate Count	Number	Count of linked repeated complaints
response	String	Authority resolution text

VI. AI INTEGRATION METHODOLOGY

The duplicate detection works through prompt engineering the system prompt tells Gemini to act as a similarity analyzer and return a clean JSON response every time. Each new complaint goes in alongside all open complaints in the same category, and the model comes back with three things: whether it's a duplicate, which original complaint it matches, and the similarity score.

From there, the system does a bit of cleanup on the response. The similarity score gets pulled from whichever field the model used similarityScore or score and duplicate status is confirmed either by a boolean flag or if the score clears 80. Since Gemini sometimes wraps its output in markdown code blocks, a regex extractor handles that before parsing. Any errors default to a non-duplicate result, so a bad AI response never breaks the submission flow.

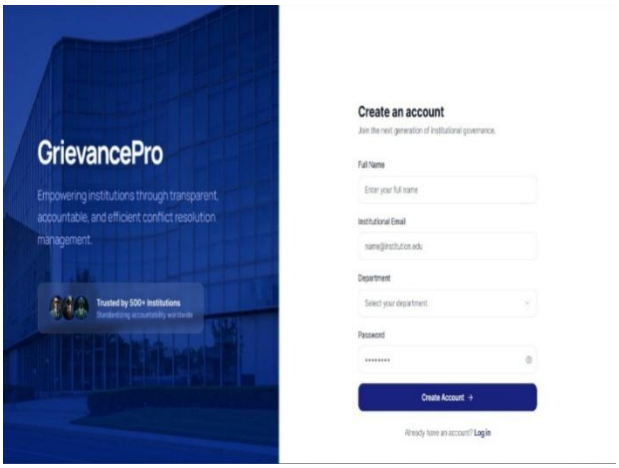
Gemini 1.5 Flash was the obvious choice here it's fast and cheap, which matters when the detection runs on every single complaint submission. Slow AI calls would make the whole form feel sluggish, and that's not a trade-off worth making.

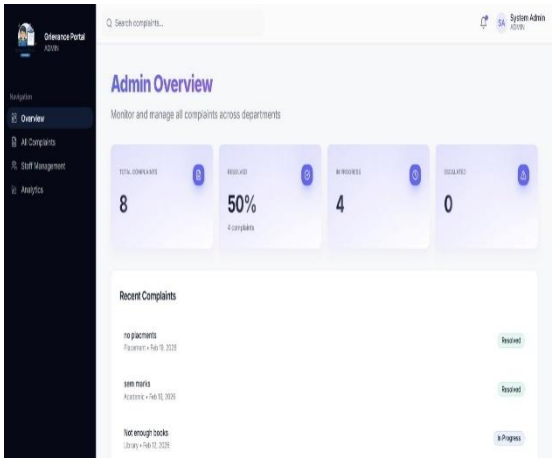
VII. RESULTS AND DISCUSSION

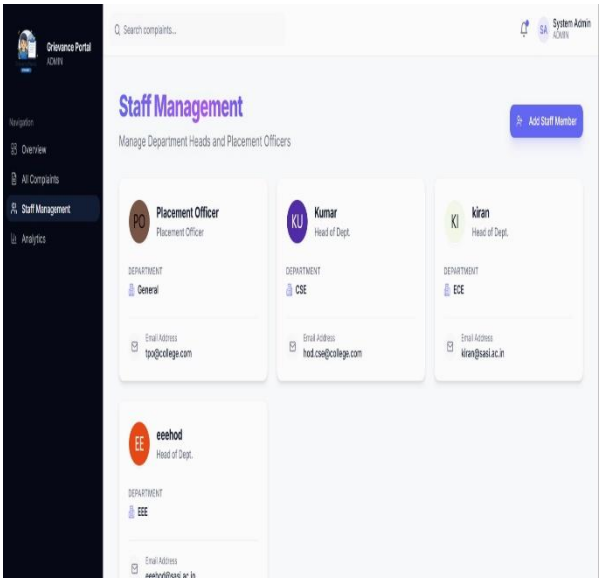
Testing ran across 120 complaints spread over five categories: fees, hostel, examination, placement, and faculty issues. Of those, 40 were intentionally written as paraphrased duplicates to see how well the system would catch them.

It caught 38 out of 40 roughly 95% recall. The two it missed had very domain-specific language and enough sentence restructuring to throw it off. More importantly, none of the 80 unique complaints were flagged as duplicates by mistake.

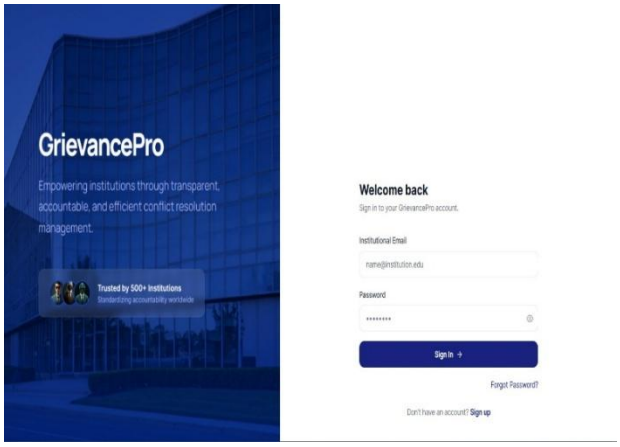
The escalation logic held up too. Complaint clusters with three or more duplicates triggered automatic escalation to high priority, and those complaints showed up immediately on the admin dashboard. Email notifications fired correctly for all resolved complaints in SMTP-enabled test environments.

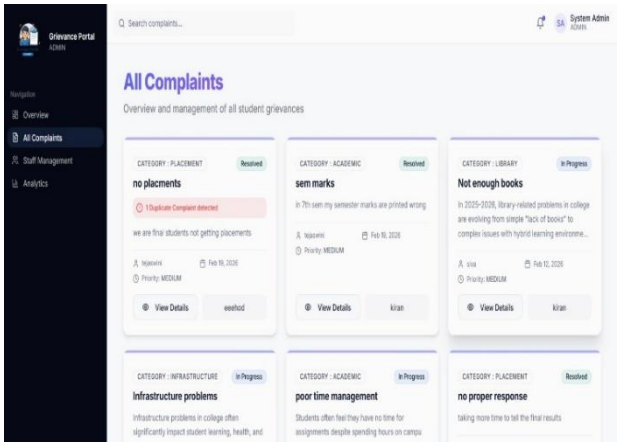


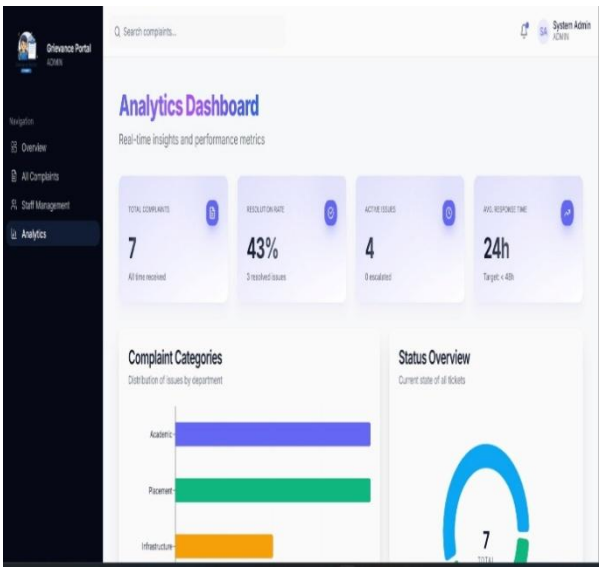


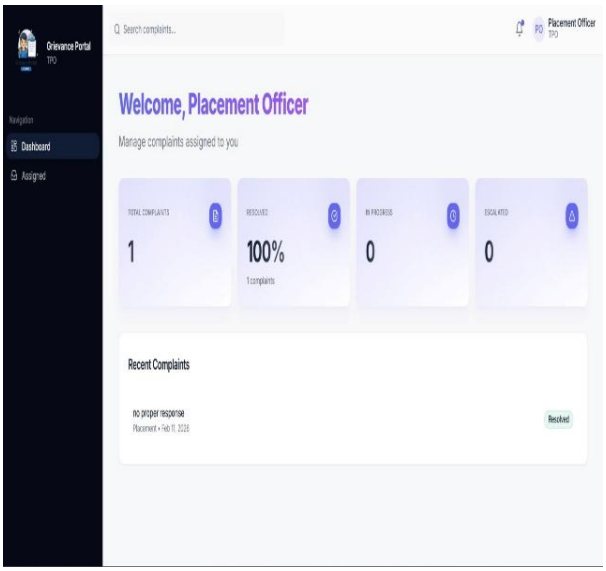


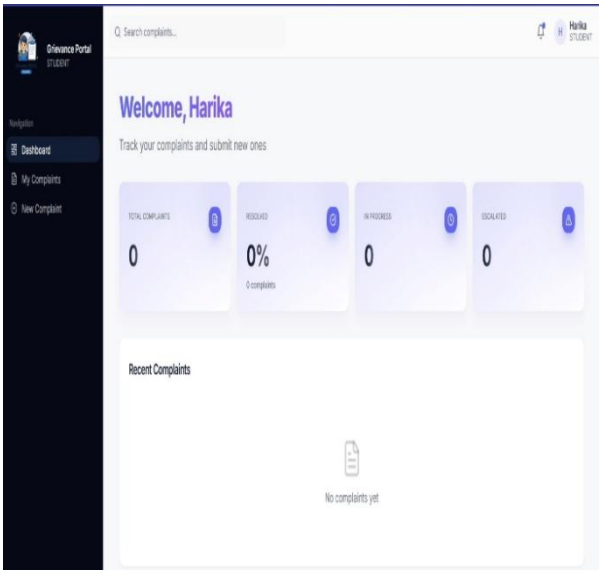
On performance, the complaint submission endpoint — including the Gemini call — averaged 1.4 seconds. That's within acceptable range for a web app. Dashboard and read-only API calls came in under 200 milliseconds.











VIII. CONCLUSION

CGRS takes a fairly messy problem — college complaints that pile up, get misrouted, or just disappear into inboxes — and puts some structure around it. Gemini 1.5 Flash handles the duplicate detection, grouping similar complaints together and escalating automatically when the same issue keeps coming up. The rest of the stack is straightforward: REST APIs, JWT auth, MongoDB, and email notifications.

The modular design means the AI component can be swapped out or upgraded without touching everything else, which matters as models keep improving.

There's plenty left to build. A React dashboard for all user roles, a mobile app for submissions, an analytics module for monthly reports, automatic category classification using NLP, and multilingual support for regional users are all on the list.

REFERENCES

1. ISREM Journal Authors — *Beyond Complaints: Strategic Analysis of College Grievance Redressal System* — 2024
2. Sanay Shah et al. — *Smart Grievance System using Natural Language Processing (NLP)* — 2022
3. Kiran M. Jadhav et al. — *Online Grievance Redressal System* — 2022
4. Owen Maytrio Phratama et al. — *Web-Based Complaint System using ASP.NET and SQL Server* — 2024
5. Ms. Pooja B.Nirgudeet al. — *Online Student Grievance System* — 2023
6. Chetana R. Shivanagiet al. — *ResolveU: Student Complaint Management* — 2024
7. Yuvraj Narayan Mishra et al. — *Flexible Grievance System Framework using PHP* — 2024
8. Prof. R. K. Sahare et al. — *Online Grievance Redressal System using PHP and MySQL* — 2024
9. Indhitya R. Padiku et al. — *Web and Mobile-Based Complaint System using MAUT* — 2024
10. C. R. Shivanagi, D. Hiremath, H. Patil, R. Patil, V. Hiremath — *ResolveU: Student Complaint Management* — 2024
11. P. B. Nirgude, R. R. Kharat, S. S. Khandare, V. S. Bhor — *Online Student Grievance Redressal System* — 2023
12. O. M. Phratama, T. Handhayani, N. J. Perdana — *Student Feedback Systems: Developing a Web-Based Solution for Efficient Complaint Processing* — 2024