



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85257>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Campus Resolve: A Role-Based College Complaint Management and Automated Escalation System Using Firebase

Mrs. K. Mutheeswari¹, P. R. Swathy², K.V. Samyuktha³

¹Assistant Professor/CSE, K.L.N. College of Engineering, Pottapalayam, Sivagangai, Tamilnadu, India

², ³UG Scholars CSE, K.L.N. College of Engineering, Pottapalayam, Sivagangai, Tamilnadu, India

Abstract: Educational institutions receive concerns related to academic services, infrastructure, administration, facilities, and student welfare. When these concerns are recorded through paper forms, informal conversations, or disconnected communication channels, it can be difficult for students to track progress and for administrators to identify delayed cases. This paper presents Campus Resolve, a web-based college complaint management system designed to organize complaint submission, assignment, tracking, deadline monitoring, escalation, feedback, and reporting within a single application. The implementation in the supplied project archive uses an HTML, CSS, JavaScript, and Bootstrap frontend, an Express and Node.js backend, Firebase Authentication for identity management, and Cloud Firestore for application data. Users are assigned role-based access according to their responsibilities, including student, staff, coordinator, Head of Department (HOD), principal, and administrator. Complaints contain structured information such as category, department, description, priority, location, submission time, status, and a server-calculated deadline. A scheduled backend process checks overdue complaints and escalates unresolved cases from the initial escalation level to the HOD and then to the principal, while generating notifications and action-history records. The application also includes dashboards, satisfaction ratings, and weekly or monthly PDF reports. The archive supports a description of these implemented functions, but it does not contain a documented benchmark dataset or measured evaluation results. Therefore, the paper discusses functional capabilities and identifies the testing required before making quantitative claims about response time, resolution improvement, or user satisfaction. Campus Resolve provides a structured digital workflow for improving visibility and accountability in institutional complaint handling.

Keywords: College complaint management, complaint tracking, automated escalation, role-based access control, Firebase Authentication, Cloud Firestore, web application.

I. INTRODUCTION

A college is a service environment in which students and staff may need to report issues involving classrooms, laboratories, infrastructure, administration, academic processes, or campus facilities. A complaint is useful only when it reaches an appropriate person, receives attention within a reasonable period, and can be followed until a clear outcome is recorded. Complaint-handling guidance emphasizes the importance of accessible reporting, consistent processing, communication, and review of outcomes [1]. In practice, however, institutions may receive concerns through several channels, including verbal communication, paper forms, email, or messaging applications. These channels can make it difficult to maintain a single history for each case.

A fragmented process creates several practical difficulties. Students may not know whether a complaint has been received or assigned. Coordinators and department heads may have limited visibility into pending cases, while administrators may find it difficult to compare complaint categories, priority levels, departments, and resolution status. When deadlines are not recorded and monitored consistently, unresolved complaints may remain unnoticed. These issues do not necessarily arise because staff are unwilling to respond; they can also result from the absence of a common tracking process and reliable records.

Campus Resolve is designed to address this coordination problem through a centralized web application. The project combines complaint submission and status tracking with role-aware access, deadline calculation, automatic escalation checks, notifications, feedback, dashboards, and periodic reports. The application uses Firebase Authentication to manage identity and Cloud Firestore to store application records, while an Express backend exposes APIs used by the browser interface. The system is rule-based rather than an artificial-intelligence classifier: the supplied source code does not implement a machine-learning model for classifying complaint text or predicting outcomes.

The main contributions of this project are:

- 1) A structured digital record for each complaint;
- 2) Access restrictions aligned with institutional roles;
- 3) Priority-based target deadlines and scheduled escalation of overdue complaints;
- 4) Notifications and an action history that improve traceability;
- 5) Analytics and report generation to support administrative review.

The remainder of this paper describes the limitations of conventional approaches, the proposed design, the implementation evidence available in the project archive, and the evaluation work needed before deployment at institutional scale.

II. EXISTING SYSTEM AND ITS DRAWBACKS

In a conventional college complaint process, students may submit concerns through written applications, direct conversations, email, or separate departmental channels. A staff member or administrator then forwards the issue to the relevant office. In some cases, the student must contact the office again to learn whether action has been taken. Records may be maintained in registers or spreadsheets, which can work for a small number of cases but require manual updates and coordination as the volume increases.

A. Limitations of the Existing Approach

- 1) Limited visibility: Students may not have a consistent way to check the current status, assigned person, or expected resolution date.
- 2) Fragmented records: When information is spread across paper records, email, and local files, reviewing the complete history of a complaint can take additional effort.
- 3) Manual deadline monitoring: A person must remember to revisit pending cases. Overdue complaints can be missed when responsibility is unclear or workloads are high.
- 4) Inconsistent prioritization: Without a common priority field and defined target periods, similar complaints may be handled differently across departments.
- 5) Weak accountability trail: If updates and decisions are not recorded in a shared history, it becomes harder to understand who changed a complaint's status and why.
- 6) Limited institutional analysis: Manual records make it difficult to prepare timely summaries by department, category, priority, status, resolution time, or feedback rating.

These limitations motivate a common digital workflow. Nevertheless, digitization alone does not guarantee faster or better resolutions. The result also depends on accurate records, appropriate role assignment, staff participation, usable interfaces, and clear institutional procedures. Campus Resolve therefore focuses on workflow visibility and administrative support rather than claiming that software by itself resolves every institutional concern.

III. PROPOSED SYSTEM

Campus Resolve centralizes complaint handling in a web application with a browser-based frontend and a server-side API. The browser interface is built with HTML, CSS, JavaScript, Bootstrap, and Chart.js for dashboard visualizations. The backend runs on Node.js and Express. Firebase Authentication supports account identity, and the backend uses the Firebase Admin SDK to access Cloud Firestore. The application separates users, complaints, notifications, complaint actions, ratings, departments, categories, and report metadata into logical collections.

A. System Architecture

The system follows a layered flow. First, the user interacts with a role-specific page in the browser. The frontend sends requests to the Express API and includes the current Firebase ID token in the Authorization header. Authentication middleware verifies the token and attaches the user's profile to the request. Authorization middleware restricts protected endpoints to permitted roles. Controllers validate the request, read or update Firestore records, and call helper functions for deadlines, notifications, action logging, and reports. The API returns structured responses to the frontend, which updates the displayed information.

Proposed system flow: User Interface → Firebase Authentication Token → Express API and Authorization Middleware → Complaint/Deadline/Notification Services → Cloud Firestore → Dashboard and Reports.

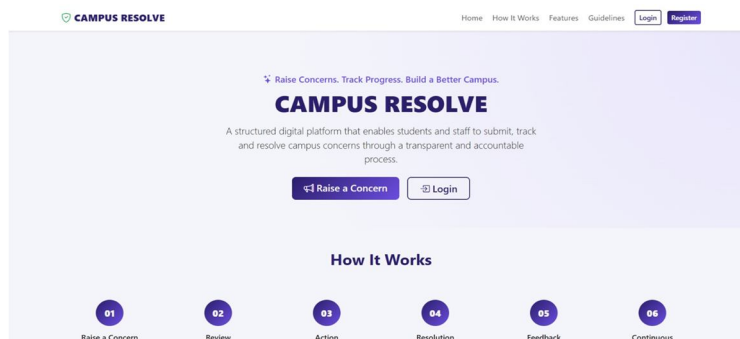


Figure.1 :Home page

B. User Roles and Access Control

The application defines six roles: student, staff, coordinator, HOD, principal, and administrator. Students and staff can register themselves, while coordinator, HOD, principal, and administrator accounts are intended to be created by an authorized administrator. Registration checks required fields, email format, password length, password confirmation, and permitted self-registration roles. Student registration also requires a register number. This reduces the chance of a user selecting an elevated role during public registration.

Role-aware filtering is applied when complaints are listed. Students and staff see complaints linked to their own user ID; coordinators see complaints assigned to them; HODs are limited to their department; and principals or administrators can review broader complaint records. Complaint detail access also checks ownership or department/assignment scope. These controls follow the general principle of role-based access control, in which permissions are associated with roles and users are assigned appropriate roles [2]. The backend is the principal enforcement point because the frontend alone cannot safely protect privileged operations.

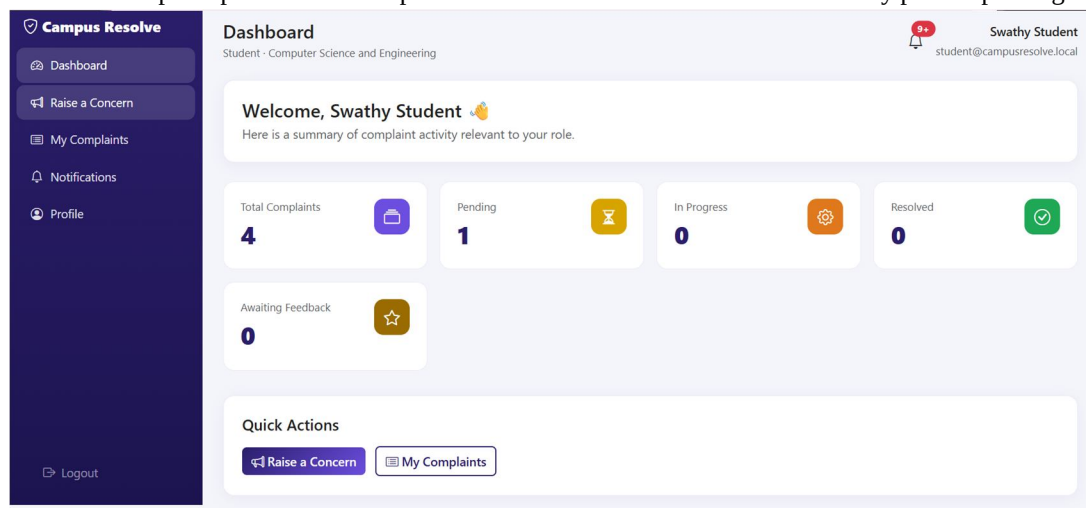


Figure.2 :Student dashboard

C. Complaint Submission and Record Creation

The complaint form collects category, department, title, detailed description, location, priority, and an optional attachment. The backend requires a title, description, and valid priority value. Supported priority levels are Low, Medium, High, and Critical. A complaint receives a generated complaint code and a status of Submitted. The record also stores the submitter's identity, department and category details where available, submission timestamp, attachment path, coordinator assignment fields, resolution fields, escalation level, and timestamps for later updates. A history entry and a confirmation notification are created after submission.

The backend calculates the target deadline instead of trusting a date supplied by the browser. The current implementation assigns 2 days for Critical priority, 7 days for High, 10 days for Medium, and 14 days for Low. The deadline is used to display the remaining time or whether the deadline has passed. These periods are implementation settings and should be reviewed and approved by the institution before operational use.

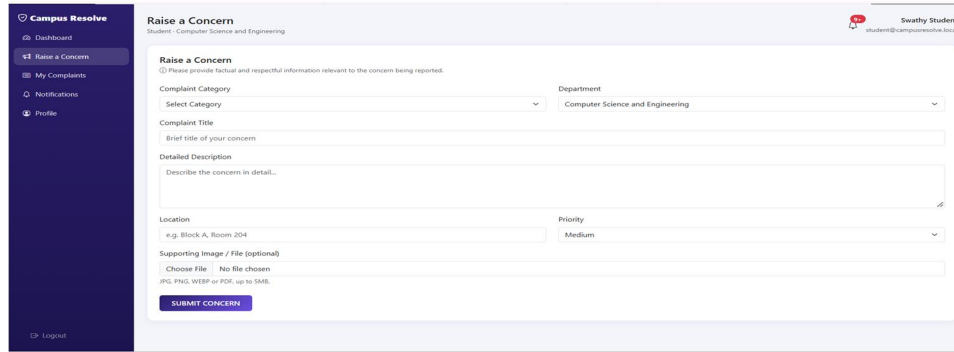
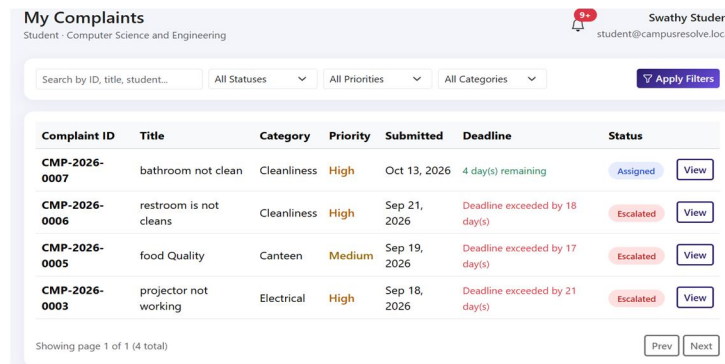


Figure.3: Raise concern



Complaint ID	Title	Category	Priority	Submitted	Deadline	Status
CMP-2026-0007	bathroom not clean	Cleanliness	High	Oct 13, 2026	4 day(s) remaining	Assigned
CMP-2026-0006	restroom is not cleans	Cleanliness	High	Sep 21, 2026	Deadline exceeded by 18 day(s)	Escalated
CMP-2026-0005	food Quality	Canteen	Medium	Sep 19, 2026	Deadline exceeded by 17 day(s)	Escalated
CMP-2026-0003	projector not working	Electrical	High	Sep 18, 2026	Deadline exceeded by 21 day(s)	Escalated

Figure.3.1: Complaints page

D. Assign and Complaint

The workflow supports states such as Submitted, Assigned, Under Review, In Progress, Resolved, Feedback, Closed, Escalated, and Reopened. Administrators and HODs can assign a coordinator to a complaint. Authorized coordinator and supervisory roles can update status, provide remarks, and escalate a case when further attention is needed. Complaint details include the current status, priority, deadline, assigned coordinator, resolution remarks, and a timeline of actions. The action history records the action, actor and role where available, timestamp, and remarks, supporting later review.

The status lifecycle represents the administrative progress of a complaint; it should not be confused with an automated judgment about the complaint's truth or severity. Priority is selected through the form, and the code validates the permitted priority values. The supplied implementation does not show a natural-language processing or machine-learning component that infers priority from the complaint description.

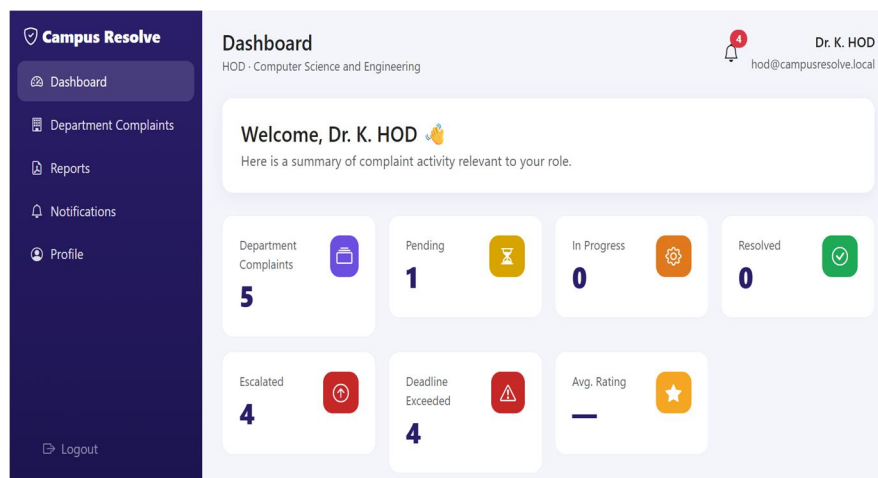


Figure.4: Hod dashboard

E. Deadline Monitoring and Automatic Escalation

A scheduled task checks complaints whose deadlines have passed. The scheduler is configured to run the overdue check hourly. An unresolved complaint whose escalation level is initially none is escalated to the HOD level. A subsequent overdue check can escalate it to the principal level, unless it has already reached that level. The system updates the complaint status to Escalated, records an escalation timestamp, and creates an action-history entry. Notifications are sent to the relevant HOD or principal and to the person who submitted the complaint.

A separate daily task checks for deadlines falling within the next two days and notifies the assigned coordinator when one exists. Weekly and monthly report-generation jobs are also scheduled. Because the overdue check is hourly, the implementation does not promise escalation at the exact moment a deadline passes; the delay may be up to the next scheduled check, depending on server operation. Reliable execution also depends on the backend process remaining active and having valid Firebase credentials.

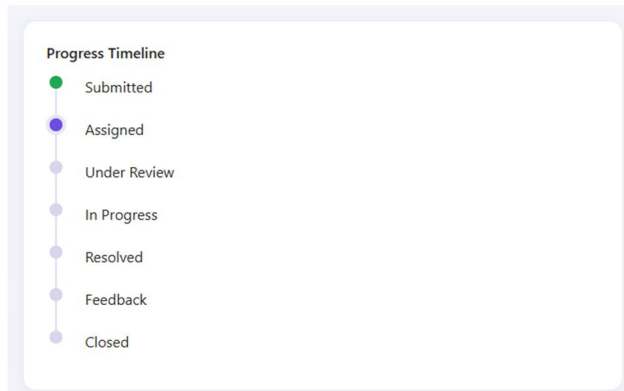


Figure.5:Deadline

F. Notifications, Feedback, Analytics, and Reports

Notifications inform users about events such as successful submission, approaching deadlines, escalation, and report availability. Users can view notifications and mark them as read. After a complaint reaches Resolved, the submitting student or staff member can provide a rating from one to five stars and optional written feedback. This creates a feedback channel, although ratings alone should not be treated as a complete measure of service quality.

The analytics API calculates totals and summaries such as resolved and pending complaints, escalated cases, resolution rate, average resolution time when timestamps are available, and average feedback rating. It can group records by department, category, priority, and status, and return weekly or monthly trends. Report generation creates weekly or monthly PDF summaries for the principal or department HODs. The reports include complaint counts, resolution-related indicators, priority distribution, escalation information, and feedback summaries when records are available.

	Complaint status updated Your complaint CMP-2026-0001 is now "Feedback". View complaint	Sep 17, 2026, 11:35 AM
	Complaint status updated Your complaint CMP-2026-0001 is now "Resolved". Please share your feedback. View complaint	Sep 17, 2026, 11:35 AM
	Complaint status updated Your complaint CMP-2026-0001 is now "In Progress". View complaint	Sep 17, 2026, 11:35 AM
	Complaint assigned Your complaint CMP-2026-0001 has been assigned to a coordinator. View complaint	Sep 17, 2026, 11:33 AM
	Complaint assigned Your complaint CMP-2026-0001 has been assigned to a coordinator. View complaint	Sep 17, 2026, 11:33 AM
	Concern submitted successfully Your concern "WERTG" has been submitted with ID CMP-2026-0001. Expected resolution by 24/9/2026. View complaint	Sep 17, 2026, 11:31 AM

Figure.6:Notification board

G. Technology Stack

The Campus Resolve system is developed using a combination of web technologies and cloud-based services. The front end is designed using HTML, CSS, JavaScript, and Bootstrap to provide a responsive and user-friendly interface. Chart.js is used to represent complaint statistics and analytical information through charts. The backend is developed using Node.js and Express.js to handle application logic, API requests, and communication between the front end and the database. Firebase Authentication is used to manage user authentication, while Firebase Admin SDK supports server-side operations. Cloud Firestore is used as the database to store complaint details, user information, status updates, feedback, and related records. The node-cron library is used to schedule background tasks, including deadline monitoring, overdue complaint checks, reminder notifications, and periodic report generation. These technologies work together to provide a centralized complaint management system with role-based access and automated escalation.

H. Methodology

The methodology adopted for Campus Resolve focuses on developing a centralized web-based system to simplify complaint management within a college environment. The process begins with identifying the difficulties associated with manual complaint handling, such as delays in communication, lack of transparency, and difficulty tracking unresolved issues. Based on these requirements, a system is designed to enable students to submit complaints, monitor their progress, and view their status through a user-friendly interface. The application follows a client-server architecture developed using HTML, CSS, JavaScript, and Bootstrap for the front end, with Node.js and Express.js handling backend operations. Firebase Authentication is used to manage user login, while Cloud Firestore stores user details, complaint records, status updates, and related information. The Firebase Admin SDK supports server-side operations, and the node-cron library is used to schedule background tasks.

After authentication, users can access the features permitted by their assigned roles. Students can register complaints by providing the required information, while authorized staff members can review and process the submitted complaints. The system maintains complaint status information throughout the workflow, allowing students to track progress from submission to resolution. Role-based access helps ensure that complaint-related actions are performed by the appropriate users.

To support timely complaint handling, the system incorporates priority-based deadlines and automatic escalation mechanisms. Critical complaints are assigned a two-day deadline, High-priority complaints a seven-day deadline, Medium-priority complaints a ten-day deadline, and Low-priority complaints a fourteen-day deadline. A scheduled background task checks for overdue complaints hourly and supports escalation from the HOD level to the principal when complaints remain unresolved. Daily reminders are also generated for complaints approaching their deadlines within two days. The application additionally supports notifications, complaint action records, feedback ratings, analytical summaries, and PDF report generation. Weekly and monthly reports help administrators review complaint-related information and monitor the overall workflow. The implemented functionalities can be verified by following the complete complaint lifecycle, including login, complaint submission, status tracking, staff review, resolution, and confirmation of the updated status. Scheduled reminders and escalation features can also be checked under their configured conditions. Since measured experimental results were not available, no quantitative performance claims are made.

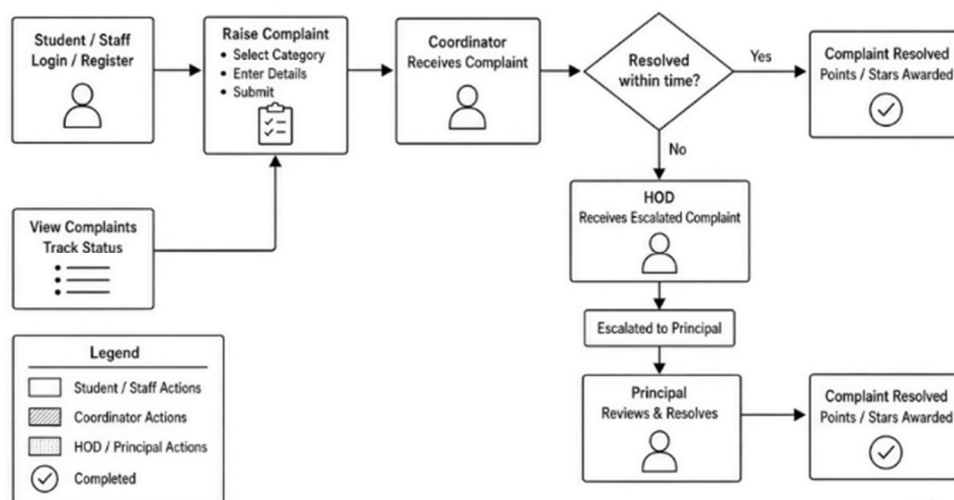


Figure.7: System architecture

I. Flow chart

The process starts when a student or staff member logs into or registers on the Campus Resolve application. After successful login, the user can raise a new complaint by selecting the appropriate category, entering the complaint details, attaching any necessary documents, and submitting it. Once submitted, the system automatically assigns the complaint to the concerned department coordinator, who reviews and begins working on it. The system then checks whether the complaint is resolved within the allotted time. If it is resolved on time, the complaint is marked as resolved, the student is awarded points or stars, and the process ends successfully. However, if the complaint is not resolved within the deadline, it is automatically escalated to the Head of the Department (HOD). The HOD reviews the escalated complaint and takes necessary action. If the issue still remains unresolved, it is further escalated to the Principal, who carefully examines the matter and provides the final resolution. Once the Principal resolves the complaint, it is marked as completed and the student receives points or stars. Throughout the process, students and staff can view their complaints and track the current status—whether submitted, in progress, escalated, or resolved—from their dashboard. This complete cycle ensures timely handling of campus issues with proper escalation and transparency.

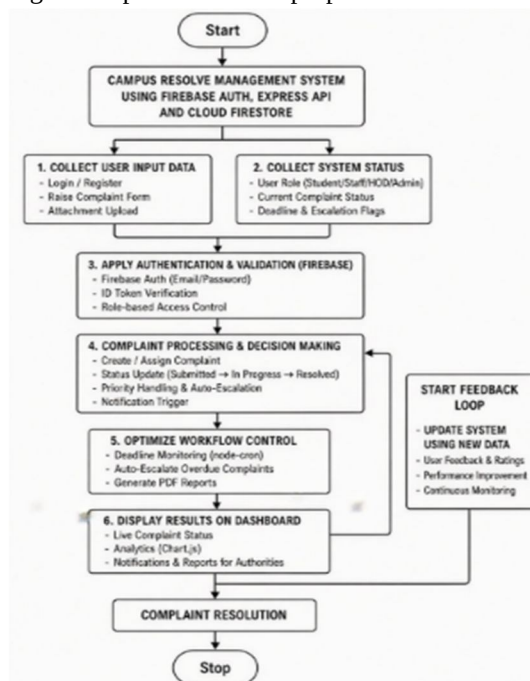


Figure.8:Flow chart

IV. RESULTS

Inspection of the supplied source archive confirms that the project contains frontend pages, API routes, controllers, middleware, Firestore integration, deadline utilities, a scheduler, notification helpers, analytics endpoints, and a report-generation utility. These components provide implementation evidence for the workflow described in this paper. The archive does not include a completed test report, a benchmark dataset, quantified user feedback, or recorded performance measurements. Consequently, this section reports verified software capabilities rather than claiming measured reductions in waiting time or improvements in satisfaction.

A. Functional Coverage Identified in the Source

The implemented system provides several functions to support the complaint management process within a college. Students can submit complaints and track their progress, while authorized staff members can access and manage complaints according to their assigned roles. The system supports role-based access for students, staff, department coordinators, Heads of Department (HODs), principals, and administrators. Complaint priority levels are associated with predefined resolution deadlines: Critical complaints have a two-day deadline, High-priority complaints have a seven-day deadline, Medium-priority complaints have a ten-day deadline, and Low-priority complaints have a fourteen-day deadline.

The system includes scheduled monitoring of overdue complaints. An hourly background task checks complaint deadlines and supports escalation from the HOD level to the principal when complaints remain unresolved. Daily reminder notifications are generated for complaints approaching their deadlines within two days. In addition, the application supports notifications, complaint action logs, feedback ratings, analytical summaries, and PDF report generation. Weekly and monthly reports help administrators review complaint-related information and monitor the overall functioning of the system.

These functionalities are identified from the project source code. However, measured performance results and experimental evaluation data were not provided, so the actual performance improvements of the system have not been quantified.

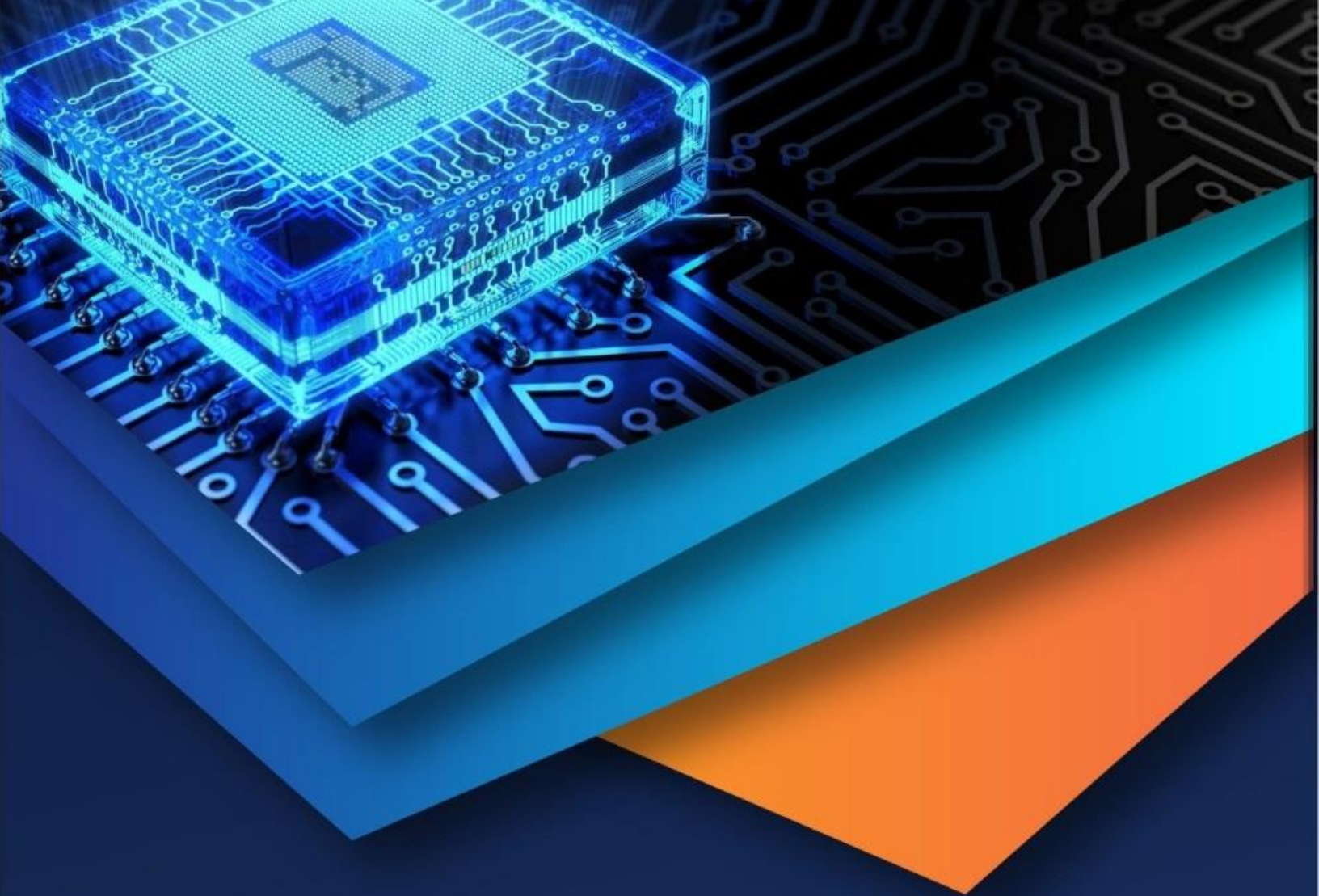
V. CONCLUSION AND FUTURE WORK

This paper presented Campus Resolve, a role-based college complaint management application that organizes complaint submission, assignment, status tracking, deadline monitoring, escalation, notifications, feedback, analytics, and periodic reporting. The supplied implementation uses a JavaScript-based web interface, an Express and Node.js backend, Firebase Authentication, and Cloud Firestore. Its scheduled workflow checks overdue complaints and escalates them through HOD and principal levels, while notifications and action records support communication and traceability. The project therefore provides a concrete software structure for managing institutional concerns in a centralized way.

The available source code supports the description of these functions, but it does not establish quantitative effectiveness or production readiness. Formal functional, security, usability, and load testing should be completed before the system is used for official institutional records. Future work can include automated test coverage, more efficient deadline queries, stronger attachment and report access controls, configurable service-level targets, notification-delivery monitoring, audit-log review tools, accessibility improvements, and integration with institutional identity systems. After collecting real usage data with appropriate privacy safeguards, the system can be evaluated for its effect on resolution time, overdue cases, and user satisfaction. Any intelligent complaint categorization or priority prediction should be treated as a future enhancement unless such a model is actually implemented and validated.

REFERENCES

- [1] International Organization for Standardization, ISO 10002:2018, Quality Management—Customer Satisfaction—Guidelines for Complaints Handling in Organizations. Geneva, Switzerland: ISO, 2018.
- [2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, doi: 10.1109/2.485845.
- [3] Google, "Firebase Authentication," Firebase Documentation. [Online]. Available: <https://firebase.google.com/docs/auth>. Accessed: Oct. 15, 2026.
- [4] Google, "Cloud Firestore," Firebase Documentation. [Online]. Available: <https://firebase.google.com/docs/firestore>. Accessed: Oct. 15, 2026.
- [5] OpenJS Foundation, "Node.js Documentation." [Online]. Available: <https://nodejs.org/docs/latest/api/>. Accessed: Oct. 15, 2026.
- [6] OpenJS Foundation, "Express—Node.js Web Application Framework." [Online]. Available: <https://expressjs.com/>. Accessed: Oct. 15, 2026.
- [7] OWASP Foundation, "Application Security Verification Standard (ASVS)," OWASP. [Online]. Available: <https://owasp.org/www-project-application-security-verification-standard/>. Accessed: Oct. 15, 2026.
- [8] Google, "Firebase Admin SDK," Firebase Documentation. [Online]. Available: <https://firebase.google.com/docs/admin/setup>. Accessed: Oct. 15, 2026.
- [9] Google, "Cloud Firestore Security Rules," Firebase Documentation. [Online]. Available: <https://firebase.google.com/docs/firestore/security/get-started>. Accessed: Oct. 15, 2026.
- [10] npm, Inc., "node-cron: A Task Scheduler in Node.js," npm package documentation. [Online]. Available: <https://www.npmjs.com/package/node-cron>. Accessed: Oct. 15, 2026.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)