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.2026.85515

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

Digital Event Documentation and Archive Platform

Jayalakshmi S[1], Monicka R[2], Monisha P[3]
[1, 2, 3] *Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, India*

Abstract: ***Colleges hold many events every year, such as workshops, guest lectures, seminars and cultural programs. In most colleges the permission for an event travels on paper from the faculty to the Head of the Department (HOD) and then to the Principal, and the photos and the final report are kept in different places afterwards. This causes delay, confusion about the status of a request, and lost records. This paper presents the Digital Event Documentation and Archive Platform, a web application in which a faculty member creates an event, the HOD and the Principal approve or reject it in order, and the same event record is later used to store media files and generate a report. Every step sends a notification inside the website and by e-mail. The Principal works as the administrator who manages the HODs, and each HOD manages the faculty of his or her department. The system is built with React, Node.js, Express and MySQL. We describe the design, the roles, the event life cycle, the database and the security choices. Formal testing has not been carried out yet, so this paper does not claim performance numbers; a testing plan is given instead.***

Index Terms: ***approval workflow, digital archive, event management, report generation, role-based access, web application.***

I. INTRODUCTION

Every department in a college plans events throughout the year, such as workshops, guest lectures, seminars and cultural programs. Before an event takes place, the organizing faculty needs permission from the Head of the Department (HOD) and from the Principal. In the usual practice, this is a letter or a form that is carried from one
office to another. If the HOD or the Principal is busy, the letter waits, and the faculty has no way of knowing where it is. The problem does not end with the approval. After the event, the photos stay on someone's phone, the attendance sheet stays in a file, and the report is typed again from memory. When the college needs the details of an event from two or three years ago, for example for an accreditation visit, it is hard to find them. Messaging apps, e-mail chains and shared folders are often used to reduce this work. They help, but they keep the approval, the photos and the report apart, and nothing records clearly who approved an event and when.

A. Motivation

A practical event management system for a college must satisfy several requirements at once. It must show the current status of every request, because otherwise the faculty keep walking to the offices to ask. It must follow the real chain of approval (Faculty, HOD, Principal) and allow a rejection with a reason. It must tell every person involved about each change, inside the website and by e-mail. It must keep the media files and the report of an event together with its details, so that old events can be found again. Finally, it must protect user accounts, because an approval is an official decision. These requirements motivate a design in which one event record carries the event from the first request to the final report, so the information is entered only once.

B. Contributions

The contributions of this work are: (i) the design of a web platform in which a single event record carries an event from request, through two levels of approval, to a stored archive and an automatic report; (ii) a role structure that matches a real college, where the Principal acts as the administrator over the HODs, each HOD looks after the faculty of his or her department, and each faculty member manages only his or her own events; (iii) an explicit status model and application programming interface (API) contract
with rejection and re-submission, so a rejected event can be corrected and sent again without losing its history; (iv) a searchable archive by date, department and event type; and (v) a testing plan that is stated openly, with no results claimed before the system is tested. The remainder of the paper is organized as follows. Section II surveys related work, Section III describes the proposed system, Section IV presents the implementation and experimental setup, Section V reports the evaluation, Section VI discusses limitations, and Section VII concludes with future work.

©IJRASET: All Rights are Reserved

II. LITERATURE SURVEY

Azmi and Ibrahim [1] developed USEM, a web-based event management system built with PHP and MySQL for a Malaysian university, where student clubs used to send paper applications to an office for approval. In the system, the organizer fills an online application form, the coordinator approves or rejects it, and the status is shown to the organizer. The system also handles event registration, payment and attendance by Quick Response (QR) code, and it can print pages such as the attendance list. The authors report that 99% of their test cases ran as intended. The approval is a single step by one coordinator, and the work does not describe an archive of event photos or an automatic event report. Pinto et al. [2] built an automated web-based work order system for a college using PHP, MySQL and Bootstrap. A request moves through a fixed chain of Department staff, HOD, Assistant Director and Secretary. Each type of user sees his or her own dashboard, the next approver is informed by e-mail when a request arrives, and a rejected request goes back to the department with comments. The system also keeps a log of every action. The authors report that the average processing time fell from about five days to under 24 hours. The system is made for maintenance requests, so it has no event record, media archive or report. Pireva and Kefalas [3] presented AcaDocMan, a web application that keeps academic documents, such as course syllabuses, in one central database as a step towards a paperless university. Academic staff and quality assurance officers can manage the documents and

generate them in different consistent formats. The work shows the value of a central archive and automatic document generation in education, but it is not connected to events or to an approval process. Bukhari et al. [4] developed iConnect, a web platform for academic and research events at a Malaysian university. Students, lecturers and organizers register for events, receive real-time updates, and the system generates participation certificates automatically when the event is completed. It shows that notifications and automatic output after an event are useful, but its description does not mention an approval chain or an archive of event media. From the analysis, it is observed that the event systems [1], [4] concentrate on the organizer and participant side, such as registration, attendance, payment and certificates, and USEM [1] has only one approval step by a single coordinator. The approval system [2] shows that a chain of several levels, with a notification at each level and a return path for rejected requests, works well in an institution, but it is built for maintenance requests and keeps no event record. The document system [3] shows the value of a central archive and automatic document generation, but it is separate from events and approvals. In the parts of the papers we read, none of the four describes storing the photos and media of a completed event, or producing an event report from the approved record. Table I summarizes these differences. The combination of a Faculty, HOD and Principal approval with rejection and re-submission, notifications by website and e-mail, a searchable media archive and an automatic report in one event record is therefore not covered by the papers we reviewed, and this is the gap that the proposed platform addresses. The survey is small, so this statement applies only to the papers reviewed.

TABLE I. COMPARISON WITH RELATED WORK (BASED ON THE

CITED PAPERS; NOT DESCRIBED = THE PAPER DOES NOT MENTION THE FEATURE) Work Primary platform

Approval handling Notifications Archive and report [1] USEM Web system (PHP, MySQL) One step: coordinator approves or rejects Status update to organizer Central database; prints pages such as the attendance list [2] Work order system Web system (PHP, MySQL, Bootstrap) Four levels, with rejection and comments E-mail at each status change Log of every action; no event archive or report [3] AcaDocMan Web application Not described Not described Central document archive; generates document formats [4] iConnect Web platform Not described Real-time updates

Automatic participation certificates Proposed system Web app (React, Node.js, MySQL) Faculty, HOD, then Principal; rejection and re-submission Website and e-mail at every stage Searchable media archive and automatic report

III. PROPOSED SYSTEM

A. *Architecture*

The platform follows a three-tier architecture (Fig. 1). The presentation tier is a React front end [5] that shows a different set of pages to the Faculty, the HOD and the Principal after login. The application tier is a Node.js [6] server with the Express framework [7] that exposes a Representational State Transfer (REST) interface in the style defined by Fielding [8], using the Hypertext Transfer Protocol (HTTP) semantics of RFC 9110 [9]. The data tier is a MySQL database [10]. The media files of completed events are kept in the local storage of the server, and the database stores only their names and locations. E-mails are sent from the server with Nodemailer through an external mail service (Gmail).

Fig. 1. Three-tier architecture of the platform.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

B. User Roles and Flow

The system has three roles, and the Principal is the administrator.

TABLE II. USER ROLES AND PERMISSIONS

Role What the role can do Faculty

Create events, edit or delete their own events, edit and re-submit a rejected event, upload media after completion, view and download the report of their own events HOD Manage the faculty accounts of their own department, review requests from those faculty, approve or reject with a reason, view the archive of their department Principal (Admin) Manage the HODs of all departments, give the final approval or rejection, view the archive of all departments A person can see and change only the data that belongs to his or her own level. A faculty member and an HOD can see only the events of their own department, and the Principal can see the events of all departments. After login the user sees the dashboard of his or her role. The faculty member fills the event form, and the event enters the system with the status Pending HOD. The HOD of that department reviews it and either approves it, which moves it to Pending Principal, or rejects it with a reason. The Principal then gives the final decision in the same way. Once the event is approved, it can be conducted. After the event, the faculty member marks it as completed and uploads the media files, and the report is generated from the stored record (Fig. 2). Every event has one status, and the status changes only in the order allowed in Table III.

TABLE III. EVENT STATUS AND ALLOWED TRANSITIONS

Status Meaning Next possible status Pending HOD

Faculty has submitted the event Pending Principal, or Rejected Pending Principal HOD has approved Approved, or Rejected Approved Both approvals are done; the event can be conducted Completed Rejected HOD or Principal rejected it, with a reason Edited and re-submitted by the faculty; it returns to Pending HOD Completed Faculty marked the event as finished and uploaded media Archived with report

Fig. 2. Event approval flow.

A rejected event is never lost. The faculty member can edit the details and submit the same event again, which moves it back to Pending HOD, and the earlier rejection stays in the approval history of that event.

C. Data Model

The relational schema (Fig. 3) contains eight tables (Table IV). The field names in Table IV are taken from the implemented schema.

TABLE IV. MAIN TABLES OF THE DATABASE

Table Main fields users id, name, email, password_hash, role, status, department_id, temporary_password, created_at departments id, name, code categories id, name events id, title, department_id, category_id, venue, event_date, mode, status, archive_status, organizer_id, created_by, created_at, and planning fields such as objective and budget estimate event_approvals id, event_id, approver_id, role, decision, comments, created_at event_media id, event_id, uploader_id, file_url, file_name, file_type, uploaded_at participants id, event_id, user_id, status, registered_at notifications id, user_id, event_id, message, is_read, created_by, created_at Each user belongs to one department, each event belongs to exactly

one faculty member, and each approval, notification and media file belongs to exactly one event or user. Every query that reads or changes events is filtered by the role and department of the requesting user.

Fig. 3. Entity-relationship diagram of the platform.

D. Approval Sequence and API Contract

Fig. 4 shows what happens when an event is submitted and then approved by the HOD, and Table V lists the API. When an event is created, the server checks the login, validates the fields, saves the event with the status Pending HOD, saves a notification for the HOD and sends an e-mail. An approval or rejection is accepted only from the HOD of the event's own department at the first level, or from the Principal at the second level, and only while the event is waiting at that level. Otherwise the request is refused, so no level can be skipped. The endpoints in Table V are taken from the implemented routes; the full API also has routes for departments, categories and event participants.

Fig. 4. Sequence of event submission and HOD approval.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

E. Safety and Security Design Decisions

Access control. The role and the department of the user are checked on the server for every request, not only in the front end, so a user cannot reach another role's data by changing the web address. Every request must carry a valid JSON Web Token (JWT) issued at login. A request for data outside the user's scope is either filtered out of the result or refused with a forbidden response. Credential protection. Passwords are never stored in plain text. Following the guidance that password storage must use salted, deliberately slow hashing [11], [12], the system stores only a salted hash (bcryptjs, with a generated salt and a cost factor of 10). The temporary password given when an account is created is handled in the same way, and the user must change it at the first login. Injection resistance. All database queries use bound parameters, so

text typed into a form cannot change the meaning of a query. File upload checks. Uploaded files are checked for type and size before they are saved. Honest degradation. If the mail service is not available, the inwebsite notification is still saved, so the user is not left uninformed. An approval is a recorded action of a logged-in user; it is not a digital signature, and the paper does not claim that it is.

TABLE V. REST API OF THE PLATFORM (REPRESENTATIVE)

Method and path Purpose Allowed role Success POST /api/auth/login Verify credentials All 200 POST /api/auth/register Create an account with a temporary password (HOD accounts by the Principal, faculty accounts by an HOD) Principal, HOD 201 POST /api/auth/change-password Change own password All

200 POST /api/events Create an event (Pending HOD) Faculty 201 GET /api/events List the events the caller may see All 200 PUT /api/events/:id Edit an event; a rejected faculty event goes back to Pending HOD Faculty, HOD, Principal 200 DELETE /api/events/:id Delete an event Faculty, HOD, Principal 200 POST /api/events/:id/approve Approve at the caller's level HOD, Principal 200 POST /api/events/:id/reject

Reject with a reason HOD, Principal 200 POST /api/events/:id/complete Mark the event completed and archive it Faculty, HOD, Principal 200 POST /api/media/upload Upload media files Faculty, HOD, Principal 201 GET /api/reports/event/:eventId/export Download the event data as a CSV file Faculty, HOD, Principal 200 GET /api/auth/notifications List the caller's notifications All 200

F. Notification, Archive and Report Modules

Notifications. A notification is created at every stage: when the event is submitted, approved by the HOD, approved or rejected by the Principal, rejected by the HOD, and when a report is ready. The message is shown on the notification page with an unread count, and the same message is sent to the person's registered e-mail address. Archive. When the faculty marks an event as completed, the system asks for the media files, such as photos, the attendance sheet or the brochure, and stores them against the event. Completed events form the archive, which can be searched by date, department or event type. Each user sees only what his or her role allows. Report. The report is built from the data already saved for the event: title, type, date, venue, organizer, resource person, approval details and the uploaded photos. Because nothing is typed again, the report is ready as soon as the event is completed. It is produced as a printable HTML page in the browser that the user saves as a PDF file through the print dialog of the browser, with no separate PDF library; a CSV export of the event data can also be downloaded from the server.

IV. IMPLEMENTATION AND EXPERIMENTAL SETUP

The front end is written in React, the back end uses Node.js with Express, and the data is stored in MySQL. Accounts are created by the level above: the Principal creates each HOD account, and each HOD creates the accounts of the faculty of his or her department, in both cases with a temporary password. The user is required to change this temporary password at the first login. Login is handled with JWT. The prototype was built and run on Windows, using Express 5.2.1, mysql2 3.23.2, Nodemailer 10.0.1, jsonwebtoken 9.0.3, Multer 2.2.0 and React 19.2.8 (versions declared in package.json), together with Node.js v24.18.1, MySQL 8.0.46 and the Google Chrome browser, version 154.0.8037.98. The main modules are listed below.

1) Login and account management: the Principal manages the HOD accounts, and each HOD manages the faculty accounts of the department.
2) Event module: faculty can create, view, edit and delete their events (full create, read, update and delete (CRUD) operations) while the event is still editable, and can edit and re-submit a rejected event.
3) Approval module: the HOD and Principal dashboards show the pending requests with approve and reject buttons.
4) Notification module: website notifications and e-mails at every stage.
5) Archive module: media upload, storage and search by date, department or event type.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

6) Report module: report generation and download.

The time needed to obtain both approvals can be modeled as Ttotal = tcreate + tHOD + tPrincipal + tnotify (1) where Ttotal is the total time to obtain both approvals, tcreate is the time to create the event, tHOD and tPrincipal are the waiting times at the two levels, and tnotify is the time to deliver the notifications. The platform removes the time spent carrying paper and asking for the status, but it cannot shorten the time an approver takes to decide. Measuring tHOD and tPrincipal for the manual and the online process is part of the planned evaluation in Section V. The sample records used for testing are synthetic (Table VI). No real names, e-mail addresses or events are used.

TABLE VI. SYNTHETIC SAMPLE DATA PLANNED FOR TESTING

Table Sample record (synthetic) users id=1, name=Faculty A, email=facultya@example.com, role=Faculty, department_id=1 departments

id=1, name=Computer Science and Engineering events id=1, organizer_id=1, title=Sample guest lecture, category_id=1, status=Pending HOD event_approvals id=1, event_id=1, approver_id=2, role=hod, decision=approved

V. RESULTS AND DISCUSSION

The system has been developed as a working prototype, but formal testing has not been carried out so far. For this reason, no response times, user counts or satisfaction scores are reported in this version, and none should be added unless they are actually measured. The tables below show how the results will be reported.

A. Functional Correctness

The functional test plan groups the test cases as in Table VII. Each case is run with sample data from Table VI, and the results are to be filled after testing. The access cases check that a faculty member, an HOD and the Principal cannot open pages or data outside their role.

TABLE VII. FUNCTIONAL TEST RESULTS ([ ] = TO BE FILLED

AFTER TESTING) Test group Cases Passed Login and role access [ ] [ ] Event creation, edit and delete

[ ] [ ] Approval, rejection and re-submission [ ] [ ] Notifications (website and e-mail) [ ] [ ] Media upload and archive search [ ] [ ] Report generation [ ] [ ] Total [ ] [ ]

B. Approval Time

The time to obtain both approvals is compared for the manual and the online process using the model in Section IV. From the current practice in the college, the manual process takes about four days, and the online platform takes about one day after the event is planned (Table VIII). These are approximate figures given by the authors and not measured values; measuring the times over real events is left as future work.

TABLE VIII. APPROXIMATE APPROVAL TIME, MANUAL AND

ONLINE (DAYS) Event Manual process (days) Online platform (days) Typical event about 4 about 1

C. User Feedback

A short questionnaire is to be given to 10 to 20 faculty members and staff after they try the platform, with questions such as ease of use, clarity of the status, and usefulness of the notifications and the report, rated from 1 to 5. [Results to be added after the survey.]

D. Discussion

Based on the design, the main benefits are that the status of every request is visible, the waiting time depends only on the approvers and not on carrying paper, and the report does not have to be written again. These are expected benefits of the design, and they must be confirmed by the tests above. The platform records and relays decisions made by people; it does not make any decision itself.

VI. LIMITATIONS AND THREATS TO VALIDITY

(1) The system has not been tested formally, and no measured results are available. (2) It has not been tried with a large number of users or with large media files. (3) An approval is a recorded action of a logged-in user, and digital signatures are not included. (4) E-mail delivery depends on the mail service and the internet connection. (5) The system is designed for one college and its departments, and it has not been checked at other institutions. (6)

The security measures in Section III-E are design decisions and should be checked against the final implementation. (7) The related-work comparison will be based on published descriptions, not on running the cited systems.

VII. CONCLUSION AND FUTURE WORK

This paper presented the Digital Event Documentation and Archive Platform, which moves the approval of college events from paper to a web system and keeps the approval, media files and report of an event in one record. The Faculty, HOD and Principal roles follow the real structure of the college, rejection and re-submission are supported, and every step is notified inside the website and by email. The next step is to test the system with real users and report the results. Future work includes digital signatures for approvals, QR-code attendance, an analytics dashboard showing events per department and per year, a mobile application, pagination of long archive lists, and short summaries of event reports using artificial intelligence (AI).

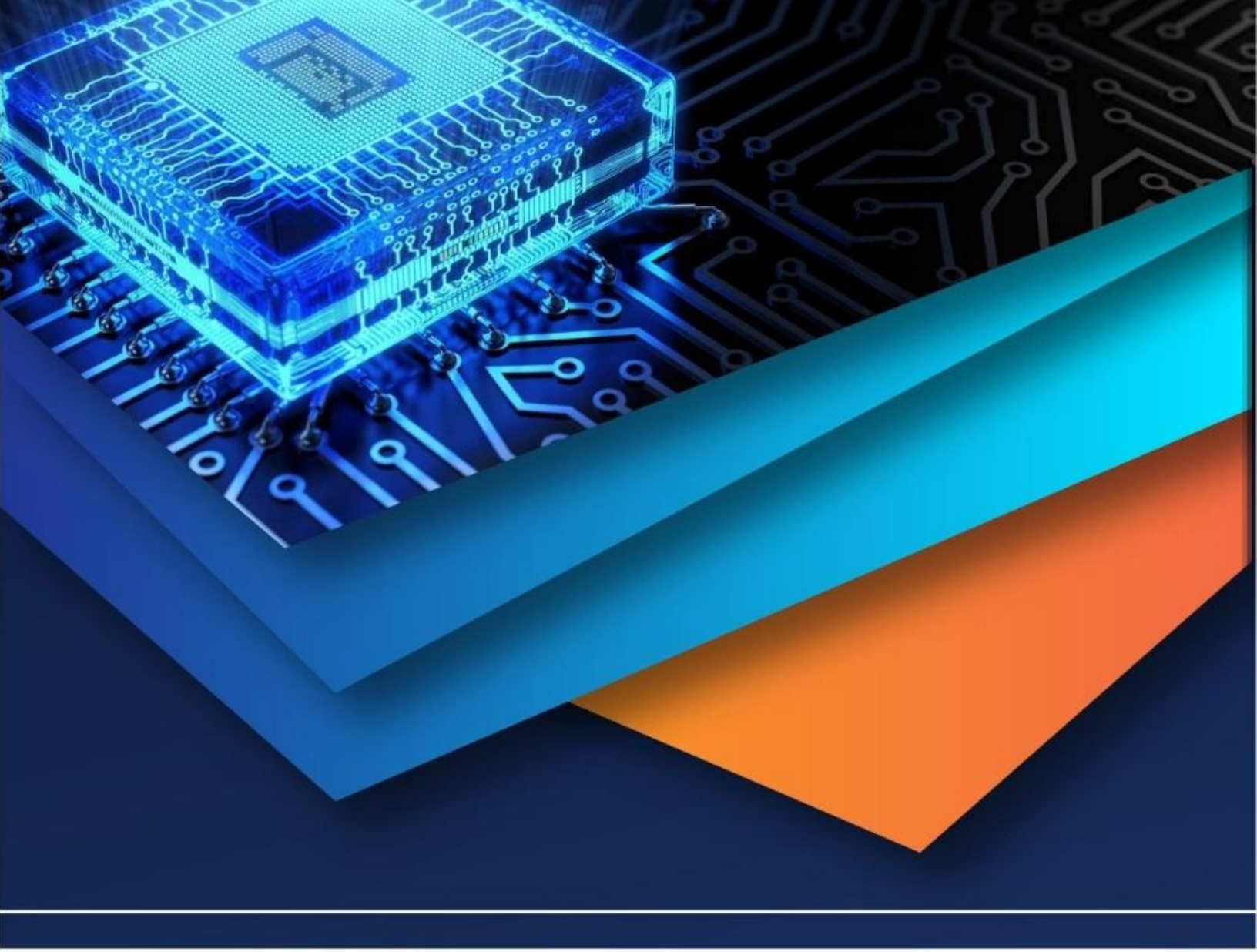
VIII. ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, and their guide A. Lavanya for the guidance and support. This work was carried out as the Final Year, 7th Semester Mini Project.

REFERENCES

[1] P. A. S. U. Azmi and N. Ibrahim, “USEM: UTHM student’s event management system,” Appl. Inf. Technol. Comput. Sci., vol. 2, no. 2, pp. 697–716, Nov. 2021, doi: 10.30880/aitcs.2021.02.02.046.
[2] J. Pinto, P. Priyadarshini, and B. Hareesh, “Automated work order management system for educational institutions,” Int. J. Eng. Res. Technol., vol. 14, no. 1, Techprints 9.0, Mar. 2026, doi: 10.17577/IJERTCONV14IS010028.
[3] K. Pireva and P. Kefalas, “An innovative web application for managing academic documents,” Int. J. Bus. Technol., vol. 1, no. 2, pp. 39–46, 2013, doi: 10.33107/ijbte.2013.1.2.04.
[4] A. N. A. Bukhari, A. Adly, I. S. Haidzir, S. S. Nathan, and N. Saji, “iConnect: A web-based digital platform for integrated management of large-scale events,” Emerg. Adv. Integr. Technol., vol. 7, no. 1, pp. 41–50, Jun. 2026.
[5] Meta Platforms, “React documentation.” [Online]. Available: https://react.dev
[6] OpenJS Foundation, “Node.js documentation.” [Online]. Available: https://nodejs.org/docs
[7] “Express documentation.” [Online]. Available: https:// expressjs.com
[8] R. T. Fielding, “Architectural styles and the design of networkbased software architectures,” Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.
[9] R. Fielding, M. Nottingham, and J. Reschke, Eds., “HTTP semantics,” IETF, RFC 9110 (STD 97), Jun. 2022.
[10] Oracle Corporation, “MySQL reference manual.” [Online]. Available: https://dev.mysql.com/doc/
[11] N. Provos and D. Mazières, “A future-adaptable password scheme,” in Proc. USENIX Annu. Tech. Conf., Monterey, CA, USA, Jun. 1999, pp. 81–92.
[12] OWASP Foundation, “Password storage cheat sheet,” OWASP Cheat Sheet Series. [Online]. Available: https:// cheatsheetseries.owasp.org/cheatsheets/ Password_Storage_Cheat_Sheet.html

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



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)