



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:

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

CampusWatch X: Autonomous Multi-Agent Campus Management System

Ashwini S P¹, Prajwal P J², Prajwal Veerappa Ujjagol², Pramodh N P², Prateek K²

¹Assistant Prof., ²UG Students, Department of Computer Science and Engineering, Jawaharlal Nehru New College of Engineering, Shivamogga, Visvesvaraya Technological University, Belagavi, Karnataka, India

Abstract: Campus services often span separate processes for room reservations, faculty consultations, incident reporting, and institutional information access. This paper presents CampusWatch X, an AI-assisted campus service framework that brings these tasks behind a common conversational interface. A Supervisor agent routes requests to specialist agents for resource booking, faculty appointment scheduling, incident management, and college-information assistance. The design combines language-model interpretation with API-mediated tools and persistent application records; the College Guide is intended to ground answers in content collected from the official institutional website. This paper describes the architecture, workflow safeguards, implementation approach, and an evaluation plan. Rather than claiming unmeasured improvements, it identifies testable measures for routing correctness, scheduling conflict prevention, incident confirmation, and answer grounding. The contribution is an integration-oriented design for campus workflows that retains explicit validation and authorization boundaries.

Keywords: Campus management, large language models, multi-agent systems, resource booking, appointment scheduling, incident management, retrieval-augmented generation.

I. INTRODUCTION

Higher-education institutions coordinate shared spaces, faculty time, student requests, and administrative information. When services are distributed across forms, email, office procedures, and independent applications, users must discover the correct channel before completing a task. A conversational interface can reduce this discovery burden, but a language model alone is insufficient for operations that change institutional records. Such operations require deterministic validation, trusted identity, and clear state transitions.

CampusWatch X is designed as a common entry point for four service domains: resource and seminar-hall booking, faculty appointment scheduling, campus incident reporting, and college-information assistance. A Supervisor agent routes a request to a specialist. Domain tools interact with structured records or retrieved website content. This design follows the broader direction of tool-using language-model agents and multi-agent systems [1], [2].

The paper makes three practical contributions: (1) a modular architecture connecting a conversational Supervisor with bounded specialist workflows; (2) workflow safeguards for state-changing tasks, including availability checks, confirmation gates, and authorization at trusted service boundaries; and (3) an evaluation plan that separates service connectivity from evidence of end-to-end correctness. This is an applied system-design paper, not a claim that the individual techniques are new.

The intended operating model is deliberately hybrid. Language-model reasoning is used for interpreting free-text requests, asking follow-up questions, and proposing structured descriptions. Deterministic application code remains responsible for checking availability, validating fields, enforcing permissions, and persisting state. This boundary reduces the risk that a plausible-sounding response is mistaken for a completed transaction. It also gives evaluators observable outcomes: whether a tool was called, whether the database changed, and whether the final message accurately describes that change.

II. RELATED WORK

A. Retrieval-Grounded Institutional Assistance

Retrieval-augmented generation (RAG) combines retrieved external passages with language generation, allowing responses to use evidence not encoded solely in model parameters [3]. This is relevant to a College Guide because institutional information can change and should be tied to authoritative sources. RAG does not guarantee factuality by itself: retrieval coverage, chunking, ranking, source freshness, and the model's use of retrieved evidence all affect answer quality. Educational RAG surveys describe growing use of this approach for domain-specific academic support [4].

For CampusWatch X, the College Guide is intended to use content collected from the official JNNCE website. The system should preserve source URLs and collection dates, retrieve passages relevant to a question, and avoid inferring policies absent from indexed material. Evaluation should include answerable questions, unsupported questions, and questions whose answers have changed on the source website.

A practical source-ingestion pipeline should preserve page title, canonical URL, fetch time, and a content hash. Text normalization should remove repeated navigation while retaining headings, tables, dates, and links that change the meaning of institutional content. Chunk boundaries should follow semantic sections where possible, because splitting a policy across unrelated fragments can remove exceptions or eligibility conditions. A refresh process should record deletions and changed pages so stale passages can be retired. These steps are design recommendations; the final paper should state which ones are implemented and which remain planned.

B. Scheduling and Campus Resources

Resource booking and faculty appointments share a temporal constraint: a requested interval must not conflict with an existing allocation.

Course-timetabling research formalizes such problems through hard constraints and optimization objectives [5]. CampusWatch X does not claim to solve institution-wide timetable optimization; its scope is request-level availability checking and workflow coordination. Research on student-lecturer appointment scheduling using multi-agent concepts is a relevant precedent for decomposing scheduling responsibilities [6].

A reliable booking workflow distinguishes a request from an approved reservation. Similarly, an available faculty slot should not be presented as a confirmed appointment if faculty approval is required. Concurrent requests are a key edge case: a check followed by a later write can allow two users to claim the same slot unless persistence or transaction logic protects against conflicts.

The booking record should preserve the requested interval, resource identifier, requester, purpose, creation time, and current decision state. The service should define whether time intervals are half-open, such as [start, end), to avoid ambiguous treatment of bookings that touch at a boundary. All timestamps should be normalized consistently and displayed in the campus time zone. Rejected or cancelled requests should not be treated as active conflicts, but the historical record should remain available to authorized reviewers. These details are important because user-interface validation alone cannot guarantee that a stored reservation is valid.

C. Incident and Multi-Agent Systems

Incident-management systems require structured reports, classification, status tracking, and an auditable resolution path. AI-assisted grievance systems can help transform free text into categories or priorities, but generated fields should remain reviewable and should not replace human accountability for consequential decisions [7]. Duplicate detection is best treated as an advisory signal rather than an automatic reason to discard a report.

LLM multi-agent surveys discuss agent roles, coordination, communication, tools, and evaluation challenges [1], [2]. A Supervisor/specialist arrangement can separate concerns, but it introduces routing errors and context-transfer risks. Testing must therefore cover each specialist independently and the end-to-end route from user request to the resulting record or answer.

III. SYSTEM ARCHITECTURE

A. Layered Design

The logical design comprises a user interface, a Node.js/Express application API, a Python FastAPI agent service, specialist agents and tools, and a MongoDB data store. The application API forwards requests to the agent service. The Supervisor selects a domain specialist, which gathers missing fields and invokes tools. Tools should return structured results so generated prose does not become the sole evidence of a state change. The College Guide follows a separate content path: institutional pages are collected, normalized, segmented, and made available to retrieval.

The development configuration described for the project uses Python agents with Google ADK and Gemini 3.1 Flash Lite, FastAPI on port 8010, Node.js/Express on port 3000, and MongoDB. Firecrawl is used for institutional website content collection. These implementation details should be checked against the final release configuration before publication.

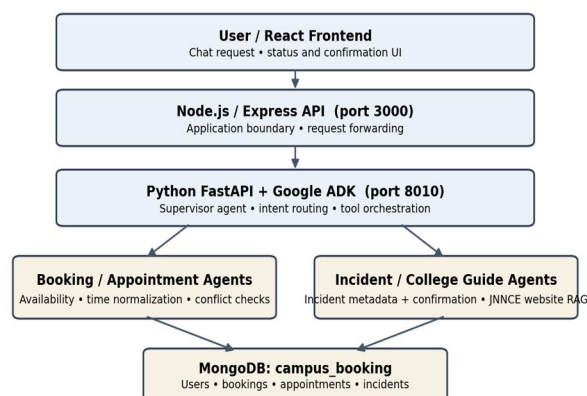


Fig. 1. Logical architecture of CampusWatch X and its service boundaries.

Table I. Core Service Modules And Responsibilities

Module	Responsibility	Control
Booking	Resource/hall request	Availability + approval
Appointment	Faculty slot request	Conflict validation
Incident	Classify/submit Report	Confirm metadata first
College Guide	JNNCE information	Retrieved source evidence
Supervisor	Route intent	Clarify ambiguity

Note: Module responsibilities describe the architecture; they do not imply all production security controls are complete.

B. Supervisor and Specialist Agents

The Supervisor acts as a router rather than an unrestricted database operator. The booking specialist handles resource selection, requested time, availability checks, and booking status. The appointment specialist resolves faculty and time-slot requests. The incident specialist structures descriptions into a proposed title, category, and priority, checks for similar reports, and requests confirmation before submission. The College Guide answers information queries using retrieved institutional content. Ambiguous requests should trigger clarification instead of a potentially incorrect state-changing action.

Routing can be evaluated as a classification problem. Each test request should have a ground-truth domain label, including an out-of-scope class and, where appropriate, a clarification-required class. The confusion matrix can reveal whether booking requests are mistaken for appointments or whether general information queries are routed to a transactional agent. Mixed-intent prompts should be tested separately because they may require sequential handling or a clarification step. A safe routing policy should prefer clarification over a state-changing action when the intent or required details remain uncertain.

C. Workflow Controls

Three controls are central to the design. First, availability and conflict checks must use authoritative records before a booking or appointment is confirmed. Second, incident metadata generated by an LLM should be presented for user review before submission. Third, authentication and authorization must be enforced by trusted backend services. A development-only actor email or user-supplied identifier is not a production identity mechanism. The authenticated user's identity should be propagated through the service boundary using a verified session or token, and every protected operation should check role and scope.

IV. IMPLEMENTATION

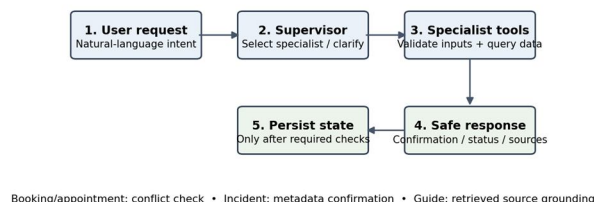


Fig. 2. Typical request-processing flow with domain-specific safeguards.

The implementation separates web-facing request handling from agent orchestration. The Express service exposes an application endpoint and forwards chat requests to FastAPI. The Python service invokes the Supervisor and relevant specialist. MongoDB stores application records, while the College Guide relies on content scraped from the official college website. This separation makes it possible to test the HTTP boundary, agent routing, and domain tools independently.

The API boundary should validate the request schema and return predictable error responses for malformed input, timeouts, unavailable downstream services, and failed persistence. Each request should have a correlation identifier that is carried across the Express service, agent service, and tools. Structured logs should record the selected specialist, tool outcomes, and error category without exposing passwords, tokens, or sensitive incident descriptions. These observability measures support debugging and make it possible to reconstruct a failed workflow without relying only on a user-visible chat transcript.

A resource-booking interaction collects a resource, date, start and end time, and purpose; validates the interval; checks for conflicts; and records a request with a status reflecting whether approval is pending or complete. Appointment scheduling compares the requested slot with faculty availability and existing appointments. The incident workflow proposes structured metadata, presents it to the user for confirmation, and only then creates the record. These intended behaviors must be checked against the final code and UI.

The College Guide content pipeline begins with web collection. Collected text should be cleaned to remove repeated navigation elements, divided into coherent passages, and indexed with source metadata. At query time, retrieved passages are supplied to the language model. A production-quality version should expose sources, refresh content periodically, and handle missing evidence explicitly. A fluent response is not sufficient proof that an answer is grounded.

V. DEVELOPMENT RESULTS AND EVALUATION

Evaluation should measure observable outcomes rather than subjective impressions. The first level is service integration: health checks and representative requests verify that Express and FastAPI communicate. The second level is specialist workflow correctness. The third level tests Supervisor routing. The fourth tests grounding and authorization. The test set should be defined before execution and include positive, negative, incomplete, and ambiguous requests.

The protocol should be repeatable. Before running tests, the team should freeze the application version, define test data, specify expected outcomes, and identify which tests are manual and which are automated. For each test, record the input, expected route, expected state transition, observed response, database evidence, execution time, and pass/fail decision. Failures should remain in the reported results rather than being silently removed. For any aggregate metric, include the numerator, denominator, and test-set composition so that a reader can judge how much evidence supports the reported value.

Development checks reported for the project included successful service health checks, a request routed through the Supervisor, and passing specialist tests for booking, appointment and incident workflows. These are development-stage checks, not a complete benchmark: the available evidence does not provide a frozen test-set size, test logs, latency distribution, or measured task-success percentage. Accordingly, this paper reports no numerical accuracy or performance claim.

A. Representative User-Query Results and Observations

To make the evaluation user-facing, representative prompts were mapped to the intended specialist and the observable response expected from the implemented workflow. These are qualitative scenario observations based on the documented project behavior, not a claim that every prompt below was executed in a controlled test session.

1) Booking query — “Book the seminar hall tomorrow from 10:00 to 11:00.”

Intended route: Booking specialist. Observation: the workflow collects resource/time details, checks availability and conflicts, and preserves the approval state; a request should not be described as approved before authorization. Evidence: workflow described.

2) Faculty appointment query — “Can I meet Professor Ravi on Tuesday afternoon?”

Intended route: Appointment specialist. Observation: the faculty identity and a usable time window are needed, and availability must be checked before confirmation. Evidence: workflow described.

3) Incident query — “Report a broken light near Block B.”

Intended route: Incident specialist. Observation: the agent drafts incident title/category/priority and asks for confirmation before submission; similar incidents may be surfaced for duplicate awareness. Evidence: behavior reported in project context.

4) College information query — “What documents are needed for admission?”

Intended route: College Guide. Observation: the answer should be grounded in retrieved official JNNCE website content; if evidence is missing, the system should clarify or state the limitation instead of inventing policy. Evidence: RAG design described.

5) Conflict query — “Reserve the same room for two overlapping time slots.”

Intended route: Booking specialist. Observation: the conflict check should reject or hold the overlapping request; near-simultaneous submissions require a concurrency test. Evidence: required negative test.

Table II. Observed Development Evidence And Open Measurements

Area	Evidence available	Not Established
Integration	Successful health checks reported during development	Sustained-load availability and reliability
Supervisor routing	Representative request routing reported	Routing accuracy on a labeled test set
Specialist agents	Specialist tests reported as passing	Test coverage, sample size, and reproducible test logs
Booking safety	Date normalization and conflict-checking logic implemented	Concurrent-booking test results
College Guide	Official JNNCE website selected as the knowledge source	Grounded-answer rate and retrieval quality
Security	College-managed accounts selected as the authentication model	Completion and validation of production authentication and authorization

Table II separates reported development checks from measurements still required. No performance values are inferred from passing tests.

- Routing accuracy: correctly routed requests divided by the number of labeled routing cases; retain the intent label, predicted specialist, and decision.
- Task completion: workflows completed with the correct persisted state divided by workflow attempts; preserve the conversation and database evidence.
- Conflict prevention: overlapping requests correctly rejected divided by conflict cases; include sequential and near-simultaneous submissions.
- Grounded-answer rate: College Guide answers supported by retrieved passages divided by answers reviewed by an evaluator.
- Unsupported-query handling: questions safely declined or redirected divided by questions not answered by indexed source content.
- Authorization correctness: protected operations correctly allowed or denied according to the approved role matrix.

Table III. Recommended Reproducible Evaluation

Metric	Definition	Evidence to Retain
Routing accuracy	Correct routed requests / total labelled requests	User intent, predicted agent, expected agent, and review outcome
Task completion	Successfully completed tasks/ total task attempts	Conversation transcript and corresponding database state
Conflict prevention	Conflicts correctly rejected / total conflict test cases	Overlapping reservations and concurrent-request test results
Grounding	Answers supported by retrieved evidence / total answers reviewed	User question, retrieved passages, citations, and reviewer judgement
Authorization	Correct allow / deny decisions / total authorization test cases	Role-permission matrix, test inputs, and observed outcomes

Scheduling tests should include overlapping reservations, invalid intervals, cancellations, pending approvals, and two requests submitted nearly simultaneously. Incident tests should verify that no record is created before required confirmation, similar incidents are surfaced without destructive auto-merging, and malformed model output is rejected. For the College Guide, each answer should be reviewed against retrieved evidence, including questions whose answers are absent or stale.

VI. DISCUSSION

The principal design benefit is separation of natural-language interaction from domain-specific execution. Users can express goals conversationally, while tools perform availability checks and database writes. Specialist agents provide clearer boundaries for testing and maintenance. The architecture is extensible: a new campus service could be introduced as a specialist, provided routing logic, permissions, and tests are updated.

The design also has limitations. LLM-based routing may be inconsistent on ambiguous or mixed-intent requests. Retrieval quality depends on coverage and freshness of scraped pages. A semantic duplicate check cannot establish that two reports refer to the same event. Multi-agent orchestration adds latency and failure points. College-managed authentication and server-side role enforcement must be completed and audited before real student or incident data is used.

Privacy and governance are also material constraints. Incident reports may contain personal information or sensitive allegations, so access should be limited by role and institutional scope. Retention periods, audit access, data export, and deletion procedures should be decided with the institution before deployment. Model providers, scraped content, and third-party crawling services should be assessed under the institution's data-handling requirements. The paper does not claim that these governance questions have been fully resolved; they form part of deployment readiness.

This work should be understood as an integration-oriented system paper, not evidence that conversational AI outperforms existing campus systems. The claims can be strengthened by a measured evaluation: a documented test corpus, routing confusion matrix, booking concurrency tests, grounded-answer review, latency distribution, and role-based authorization matrix. User studies could assess usability and task completion time, subject to institutional approval and privacy safeguards.

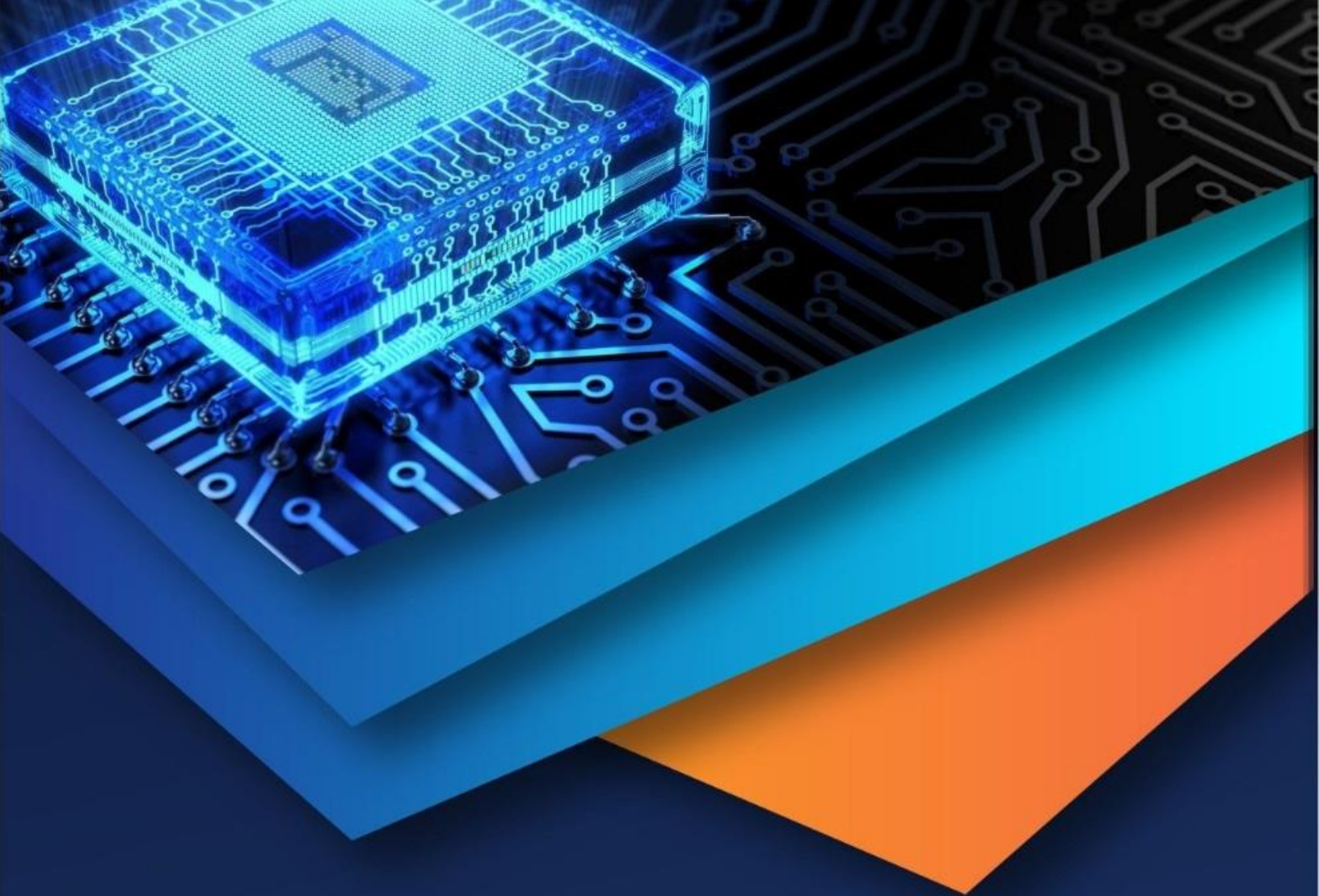
VII. CONCLUSION AND FUTURE WORK

CampusWatch X presents a multi-agent approach to unifying resource booking, faculty appointment scheduling, incident reporting, and institutional information access. A Supervisor routes requests to specialist agents, while API services and structured persistence support operational workflows. The College Guide is designed to ground answers in official institutional web content. The design emphasizes that model-generated text must not replace deterministic validation, trusted identity, user confirmation, or auditable state changes.

Future work includes validating college-managed authentication, enforcing role-based permissions across services, making booking and appointment writes concurrency-safe, refreshing website content with traceable metadata, and adding source-linked answers. A formal evaluation should measure routing accuracy, task completion, conflict prevention, grounded-answer rate, latency, and authorization correctness. Only results obtained from that evaluation should support quantitative claims.

REFERENCES

- [1] T. Guo et al., "Large Language Model Based Multi-agents: A Survey of Progress and Challenges," Proc. 33rd Int. Joint Conf. Artificial Intelligence (IJCAI), 2024, pp. 8048–8057, doi: 10.24963/ijcai.2024/890.
- [2] Y. Wang et al., "A Survey on Large Language Model Based Autonomous Agents," Frontiers of Computer Science, vol. 18, 2024, doi: 10.1007/s11704-024-40231-1.
- [3] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.
- [4] J. Swacha and M. Gracel, "Retrieval-Augmented Generation (RAG) Chatbots for Education: A Survey of Applications," Applied Sciences, vol. 15, no. 8, art. 4234, 2025.
- [5] M. Chen, F. Werner, and M. Shokouhifar, "Mathematical Modeling and Exact Optimizing of University Course Scheduling Considering Preferences of Professors," Axioms, vol. 12, no. 5, art. 498, 2023, doi: 10.3390/axioms12050498.
- [6] B. Anthony Jnr., G. P. Maureen, and M. A. Majid, "A Web Deployed Multi-Agent Based Approach for Student-Lecturer Appointment Scheduling in Institutions of Higher Learning," Journal of Physics: Conference Series, vol. 1830, 012007, 2021, doi: 10.1088/1742-6596/1830/1/012007.
- [7] "Automation Tool for Grievances Using Artificial Intelligence," Procedia Computer Science, vol. 283, pp. 2770–2779, 2026, doi: 10.1016/j.procs.2026.06.346.



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)