iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81609

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 IV Apr 2026- Available at www.ijraset.com

Smart Faculty Workload and Timetable Management System Using Rule-Based Automation

Nimmakayala Venkata Lakshmi[1], Mamillapalli Bilvika[2], Patan Neelofar[3], Borru Rajyanka[4]
[1]*Assistant Professor, Department of Computer, Science and Engineering (AIML), Bapatla Women's Engineering College, Bapatla, India*
[2, 3, 4]*Department of Computer Science and Engineering (AIML), Bapatla Women's Engineering College, Bapatla, India*

Abstract: ***Efficient timetable generation and faculty workload management are essential for the smooth functioning of academic institutions. Manual scheduling methods are often time-consuming, prone to conflicts, and inefficient in handling dynamic changes such as faculty leave, subject reallocation, and timetable modifications. These challenges may lead to uneven workload distribution, scheduling errors, and administrative delays. To overcome these issues, this paper proposes a Smart Faculty Workload and Timetable Management System Using Rule-Based Automation, a web-based solution designed to streamline academic scheduling and workload allocation processes. The system incorporates a Role-Based Access Control (RBAC) mechanism supporting three primary users: Administrator, Faculty, and Student. Administrators can generate timetables, approve leave requests, monitor faculty workload, and manage institutional records. Faculty members can view monthly timetables and apply for leave, while students can access updated schedules in real time. A rule-based scheduling engine developed using Python and Flask ensures conflict-free subject allocation by evaluating faculty availability, workload limits, and institutional constraints. The system utilizes ReactJS for an interactive user interface and MongoDB for scalable data management within a three-tier architecture. Experimental evaluation indicates improved scheduling accuracy, balanced workload distribution, reduced manual effort, and enhanced transparency in academic management.***

Keywords: Timetable Automation, Faculty Workload Management, Rule-Based Scheduling, Academic Scheduling System, Role-Based Access Control (RBAC), Web Application Development, Flask Framework, MongoDB Database.

I. INTRODUCTION

Educational institutions operate within a structured academic framework where effective timetable preparation and faculty workload management play a crucial role in maintaining operational efficiency.

Timetable scheduling is not merely the allocation of subjects to time slots; it involves careful coordination of faculty availability, subject requirements, classroom allocation, and institutional workload policies. As institutions expand in terms of courses, departments, and student strength, the complexity of scheduling increases significantly. Managing these processes manually often leads to inefficiencies and operational challenges.

In many colleges and universities, timetable generation is still performed using spreadsheets or semi-manual approaches. These traditional methods are time-intensive and highly dependent on administrative coordination. Even minor changes, such as a faculty leave request or subject adjustment, may require substantial rework. This often results in timetable clashes, uneven distribution of teaching hours, and communication gaps between administrators, faculty members, and students. Over time, such inefficiencies can negatively impact academic planning and institutional productivity. Another major challenge in academic management is maintaining a balanced workload among faculty members. Unequal subject allocation may lead to faculty dissatisfaction and reduced teaching effectiveness. Additionally, the absence of real-time timetable visibility restricts students and faculty from accessing updated schedules promptly. Institutions require a centralized, automated system that ensures fairness, transparency, and adaptability while minimizing administrative burden. To address these challenges, this project introduces a Smart Faculty Workload and Timetable Management System Using Rule-Based Automation. The system is designed as a web-based solution that automates subject allocation and timetable generation by applying predefined institutional rules.

The rule-based engine evaluates multiple constraints such as faculty availability, maximum workload limits, subject requirements, and departmental policies to generate conflict-free and optimized schedules.The system incorporates a Role-Based Access Control (RBAC) mechanism that supports three primary stakeholders: Administrator, Faculty, and Student. The administrator has centralized control for timetable generation, leave approval, workload monitoring, and user management.

Faculty members can view monthly timetables and apply for leave, while students access real-time schedule updates through dedicated dashboards. This role-based structure ensures secure and organized communication. Built with Python and Flask for the backend, ReactJS for the frontend, and MongoDB for data storage, the system follows a three-tier architecture for scalability and security. By automating scheduling and supporting real-time updates, it reduces manual effort, improves accuracy, and maintains balanced workload distribution.

II. LITERATURE SURVEY

Kaur et al. developed an automated timetable generation system using genetic algorithms to minimize scheduling conflicts. Their approach optimized time-slot allocation efficiently under predefined constraints. However, the system required high computational tuning and lacked real-time update capabilities for dynamic changes such as faculty leave. In contrast, the proposed system uses rule-based automation with instant dashboard updates and simplified constraint handling. [1]

Al-Yakoob et al. proposed a university course scheduling model using constraint satisfaction techniques. The system successfully handled classroom and time-slot constraints. However, it did not focus on faculty workload balancing or role-based system interaction. In contrast, the proposed system integrates workload distribution logic along with structured role-based access control. [2]

Raghavendra et al. implemented a web-based academic scheduling platform using MySQL and PHP. Their system automated timetable generation but required manual intervention for leave management. In contrast, the proposed system incorporates automated leave approval workflows and substitution handling through rule-based logic. [3]

Bhardwaj et al. applied heuristic optimization techniques to generate conflict-free timetables. Although effective in reducing clashes, their model did not provide separate dashboards for students and faculty. In contrast, the proposed system provides dedicated dashboards with real-time visibility for all stakeholders. [4]

Santos et al. developed a workload analysis tool for faculty performance monitoring. The system evaluated teaching hours but did not integrate timetable automation. In contrast, the proposed system combines workload monitoring with automated scheduling in a single platform. [5]

Lee et al. applied artificial intelligence techniques for dynamic timetable optimization. While their approach improved scheduling efficiency, it required extensive training datasets and computational resources. In contrast, the proposed system adopts lightweight rule-based automation suitable for institutional deployment without heavy infrastructure.[6]

Mishra et al. introduced a cloud-based academic management system for timetable viewing and record storage. However, their system lacked intelligent scheduling automation and primarily functioned as a static record portal. In contrast, the proposed system actively generates timetables using predefined institutional rules. [7]

Sharma et al. developed a leave management system for academic institutions. Their work streamlined leave approvals but operated independently from timetable scheduling modules. In contrast, the proposed system integrates leave management directly with timetable regeneration logic. [8]

Patel et al. designed a faculty allocation system using linear programming to optimize subject assignments. Although it improved allocation fairness, it lacked real-time modification capabilities. In contrast, the proposed system ensures instant reflection of administrative changes across dashboards.[9]

Zhang et al. implemented a role-based academic management system with layered architecture. Their model enhanced data security but did not incorporate automated workload balancing. In contrast, the proposed system integrates workload calculation modules with scheduling automation. [10]

Kumar et al. developed a timetable generation system using a backtracking algorithm to avoid scheduling conflicts. The method effectively prevented overlapping assignments but showed increased computational complexity as institutional size grew. This limited its scalability in larger academic environments. In contrast, the proposed system uses predefined rule-based logic to ensure faster execution, simplified conflict handling, and better scalability for real-time deployment. [11]

Hassan et al. proposed a distributed scheduling framework for multi-department institutions, allowing decentralized timetable management. While flexible, the system lacked centralized workload monitoring to ensure balanced teaching hours among faculty. In contrast, the proposed system provides centralized administrative control with integrated workload tracking, improving transparency and coordination. [12] Nguyen et al. developed a MongoDB-based academic record system to enhance scalability and flexible data storage. Although efficient in handling institutional records, it did not focus on timetable automation or workload balancing. In contrast, the proposed system utilizes MongoDB not only for storage but also for real-time scheduling, leave management, and workload computation. [13]

Verma et al. implemented a ReactJS-based academic dashboard to improve user interface responsiveness. While the system enhanced user experience, it lacked backend automation for timetable generation and workload management. In contrast, the proposed system integrates a dynamic ReactJS frontend with a rule-based backend engine to deliver automated scheduling and structured academic management. [14]

Chaudhary et al. proposed a smart institutional management framework integrating attendance and timetable modules. While comprehensive, it lacked structured workload balancing mechanisms and substitution logic. In contrast, the proposed system emphasizes balanced workload distribution and automated substitution handling. [15]

III. METHODOLOGY

The Smart Faculty Workload and Timetable Management System is designed as a modular rule-based framework for academic scheduling and workload management. It integrates authentication, role-based access control, automated timetable generation, workload monitoring, leave handling, and real-time updates within a web platform. The rule engine evaluates faculty availability and workload constraints to produce conflict-free schedules. Using Python and Flask for backend processing, ReactJS for frontend interaction, and MongoDB for data storage, the system provides a scalable and efficient academic management solution.

A. User Authentication and Role Identification

The system begins with secure user authentication implemented through the Flask backend. Login credentials are verified against records stored in MongoDB using encrypted password validation mechanisms. Upon successful authentication, the system identifies the user's role as Administrator, Faculty, or Student. Based on the detected role, a corresponding dashboard is dynamically rendered using ReactJS. This process ensures controlled system access, protects sensitive academic data, and prevents unauthorized modifications. It establishes the foundation for secure and role-specific interaction within the platform.

B. Role-Based Access Control (RBAC)

A Role-Based Access Control mechanism is implemented to regulate system functionalities according to user roles. Administrators are granted full privileges, including timetable generation, leave approvals, workload monitoring, and user management. Faculty members are authorized to view assigned timetables and submit leave requests, while students are provided read-only access to month-wise schedules. Access permissions are enforced at both backend API and frontend interface levels. This structured access control framework enhances system security, accountability, and operational transparency.

C. Rule-Based Timetable Generation

The timetable generation module applies predefined institutional rules to allocate subjects efficiently. The system evaluates faculty availability, workload limits, subject requirements, and time-slot constraints before final assignment. Conflict detection logic prevents overlapping schedules or double booking of faculty members. The rule-based engine ensures balanced distribution of subjects across available slots while maintaining institutional policies. Generated timetables are stored in MongoDB and reflected immediately across dashboards. This module represents the core automation engine of the system.

D. Workload Monitoring and Balancing

The workload management module continuously tracks the total teaching hours assigned to each faculty member. Before assigning a subject, the system verifies that the workload remains within predefined institutional limits. If the maximum threshold is reached, alternative faculty members are evaluated. This mechanism ensures equitable workload distribution and prevents faculty overburdening. Workload data is dynamically updated to maintain accuracy and transparency. This method supports fair academic management and improved faculty satisfaction.

E. Leave Management and Substitution Handling

Faculty members can submit leave requests through their respective dashboards. These requests are stored in the database and routed to the administrator for approval. Once approved, the system identifies suitable substitute faculty based on availability and current workload capacity. The timetable is automatically updated to reflect substitutions without causing scheduling conflicts. Notifications are displayed across relevant dashboards to maintain coordination. This method ensures uninterrupted academic operations.

F. Month-Wise Timetable Retrieval

Students and faculty members can select a specific month to view their schedules. The system retrieves filtered timetable data from MongoDB based on user credentials and selected month parameters. ReactJS dynamically renders the timetable in a structured and user-friendly format. Any administrative modifications are instantly synchronized across dashboards. This feature ensures organized schedule access, real-time visibility, and improved academic planning efficiency.

IV. SYSTEM IMPLEMENTATION

The proposed Smart Faculty Workload and Timetable Management System is implemented as a modular web-based framework integrating authentication, role-based access control, and rule-based timetable generation. Built using ReactJS, Flask, and MongoDB, the system evaluates faculty availability and workload constraints to produce conflict-free schedules. The structured workflow ensures secure access and efficient academic management.

A. User Authentication Module

The module handles secure login verification for administrators, faculty, and students. It validates credentials using encrypted password matching through the Flask backend. Upon successful authentication, role-specific dashboards are generated.

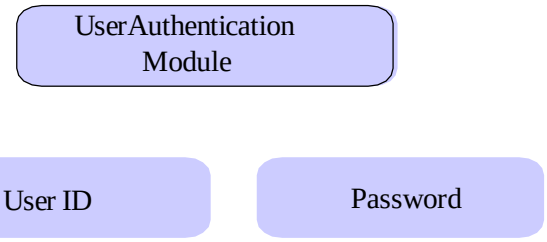


Fig.1.User Authenticationmoduleworkflow

B. Role-Based Access Control (RBAC) Module

The RBAC module defines permission levels based on user roles: Administrator, Faculty, and Student. Access restrictions are enforced at both frontend and backend levels to prevent privilege escalation. Administrators receive full system control, faculty members are limited to timetable and leave-related operations, and students are provided read-only access to schedules. This module ensures structured interaction and secure separation of responsibilities. The interaction and featuresare depicted in Figure 2.

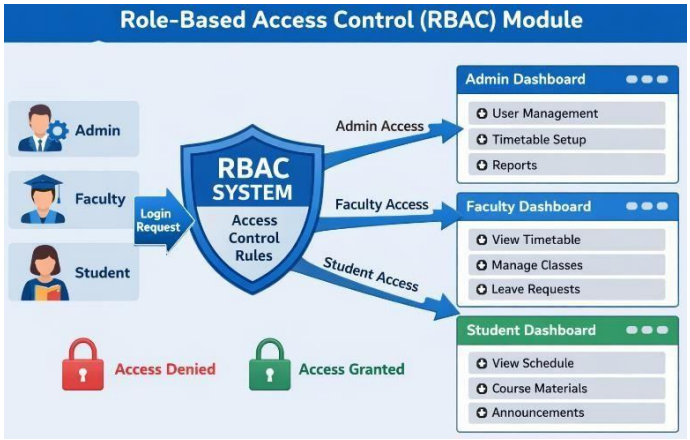


Fig.2.RBACfunctionality

C. Admin Management Module

The Admin Management Module provides centralized control over academic scheduling operations. It enables administrators to generate timetables, approve or reject leave requests, monitor faculty workload distribution, and manage user records. The module also supports adding, updating, or removing student and faculty information. All administrative actions trigger database updates and real-time synchronization across dashboards.

This module also supports workload monitoring and leave approvals. Administrators can review teaching hours, approve leave requests, and manage student and faculty records. All actions are securely processed and updated in real time across dashboards. These preprocessing steps significantly improve feature clarity and detection reliability in subsequent stages as shown in Figure 3.

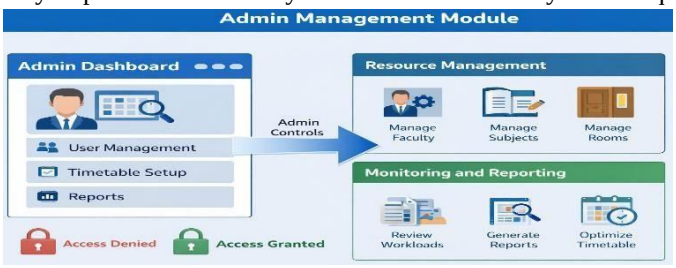


Fig.3.Admin Management moduleWorkflow.

D. Faculty Management Module

The Faculty Management Module allows faculty members to securely log in and access their personalized dashboard. Through this module, they can view assigned subjects, monitor teaching workload, and access month-wise timetables. The system retrieves schedule data from MongoDB and displays it dynamically using ReactJS, ensuring real-time visibility.The module also enables faculty to submit leave requests and track their approval status. Approved changes are reflected instantly in their timetable, ensuring clear communication and coordinated academic management, as depicted in Figure 4.

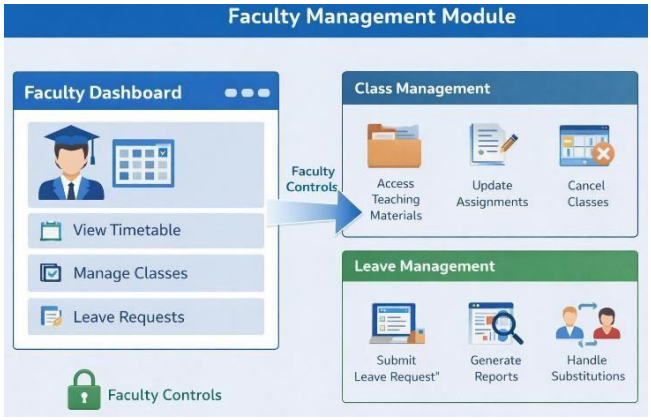


Fig.4.Faculty managementModule workflow

The structured data retrieved and displayed through this module forms the foundation for accurate workload tracking and efficient schedule management under dynamic academic conditions.

E. Student Management Module

The Student Management Module provides secure and structured access to academic timetables. Students can log in through authenticated credentials and access their personalized dashboard. The system retrieves relevant schedule data from MongoDB and ensures read-only access to maintain data integrity.

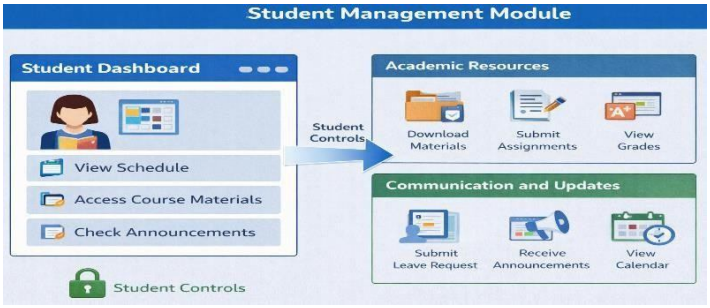


Fig.5.Student managementmodule

F. Rule-Based Timetable GenerationModule

TheRule-Based Timetable Generation Module functions as the core automation engine of the system. It generates schedules by evaluating faculty availability, subject requirements, workload limits, and time-slot constraints. Conflict detection logic prevents overlapping classes or double booking of faculty members. If violations are detected, alternative allocations are considered. The finalized timetable is stored in MongoDB and synchronized across dashboards, ensuring conflict-free and balanced scheduling.

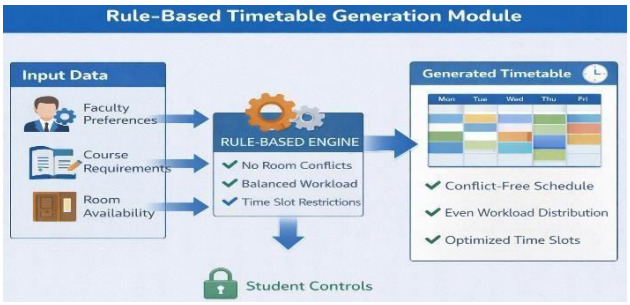


Fig.6.Rule-based timetable generationmodule

G. Severity Assessment Module

The Workload Monitoring Module tracks the total teaching hours assigned to each faculty member. Before confirming subject allocation, the system verifies that workload limits are not exceeded. If thresholds are reached, alternative faculty members are evaluated. The module provides administrators with workload summaries for monitoring and adjustments. This mechanism ensures fair workload distribution and efficient academic management.

H. Leave Management and Substitution Module

The Leave Management Module enables faculty members to submit leave requests through their dashboards. Requests are stored in the database and reviewed by the administrator. Upon approval, the system activates substitution logic to assign eligible faculty based on availability and workload capacity. Timetables are updated automatically, ensuring uninterrupted academic sessions and real-time schedule consistency.

V. RESULTS AND DISCUSSION

The developed Smart Faculty Workload and Timetable Management System was evaluated using simulated faculty, subject, and student data. The workflow, including authentication, rule-based timetable generation, workload balancing, and leave management, was tested across all modules. Performance was assessed based on scheduling accuracy, conflict prevention, and system responsiveness.

A. User Interface

The interface combines the login panel and Admin Dashboard of the Smart Faculty Workload and Timetable Management System. The left section provides role selection (Student, Faculty, Admin) with secure email and password authentication. Upon admin login, the right section displays a structured dashboard with navigation modules such as Faculty, Student, Workload, Timetable, Notifications, and Leaves. The dashboard overview presents key statistics including total faculty, students, and holidays for quick monitoring. The clean and responsive design ensures secure access, real-time visibility, and efficient academic managementas shown in Figure 6.

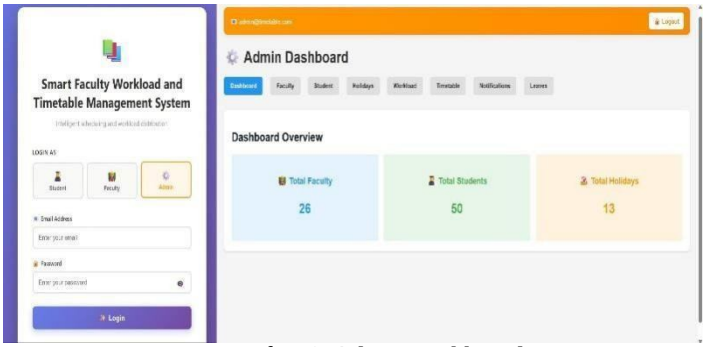


Fig.7.UserInterface& Admin Dashboard.

B. Workload View

Thisinterface represents the Faculty Workload Management section within the Admin Dashboard. It allows the administrator to select a specific month and monitor the total teaching hours assigned to each faculty member. The table displays faculty names, subjects handled, and corresponding workload hours for clear comparison. This module helps ensure balanced workload distribution and supports efficient academic planning.

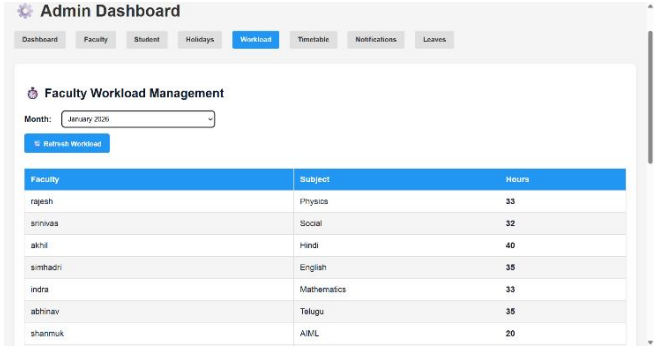


Fig.8.Faculty Workload management.

C. Timetable Management & Timetable View

The images show the Timetable Management module allows administrators to generate and update monthly schedules by selecting the month, year, and holidays. The Weekly Timetable View displays detailed class-wise schedules with subjects, faculty, and time slots. Users can navigate between weeks and filter by class for organized viewing. These features ensure structured planning and real-time timetable updates.

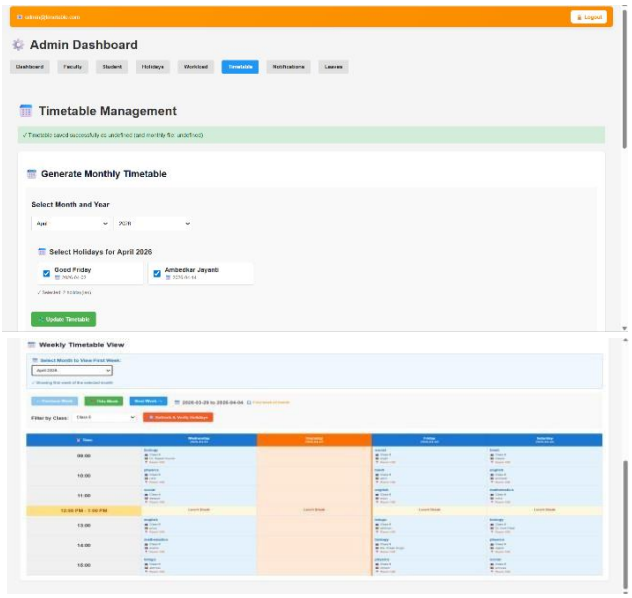


Fig. 9.Timetable Management & Timetable View.

VI. CONCLUSION

This paper presented the design and implementation of a Smart Faculty Workload and Timetable Management System using rule-based automation to streamline academic scheduling and workload distribution. The system integrates automated timetable generation, workload balancing, leave management, and real-time dashboard updates to ensure transparent and efficient institutional management. By applying predefined rules that evaluate faculty availability, workload limits, and subject constraints, the system generates conflict-free schedules while maintaining fair workload allocation. Developed using Python and Flask for backend processing, ReactJS for frontend interaction, and MongoDB for scalable data storage, the system follows a modular three-tier architecture that supports maintainability and security.

Experimental evaluation demonstrated reduced manual administrative effort, improved scheduling accuracy, and enhanced coordination among administrators, faculty, and students. Although large-scale deployment may require performance optimization strategies such as database indexing and load balancing, the proposed framework provides a scalable, cost-effective, and practical solution for modern educational institutions. Future enhancements will focus on integrating advanced optimization techniques, intelligent scheduling algorithms, and real-time notification systems to further improve automation efficiency and adaptability.

.

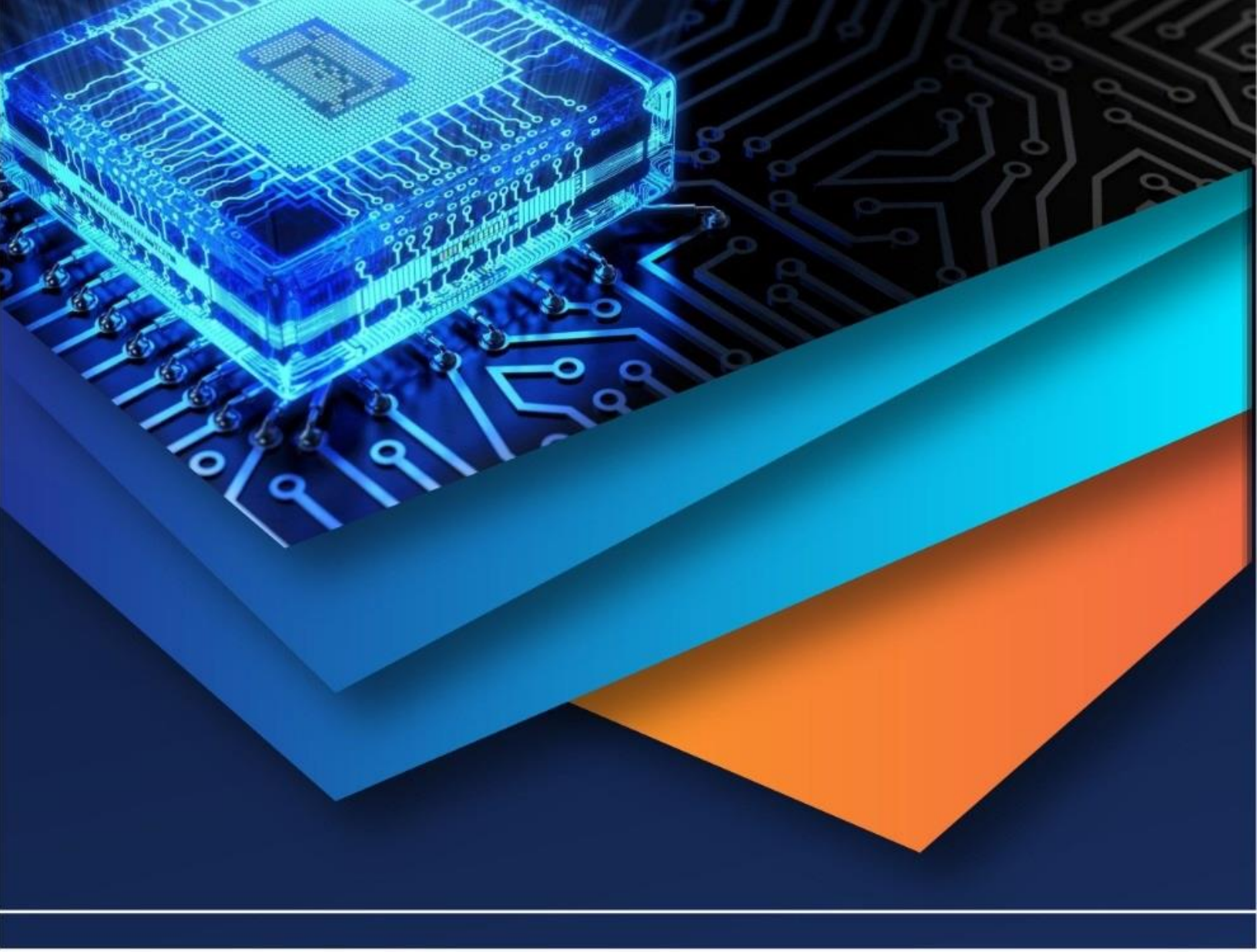
VII. ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the faculty mentors and academic coordinators for their valuable guidance and support during the design and implementation of this project. Their suggestions were instrumental in refining the system architecture and ensuring its practical applicability in academic institutions.

The authors also acknowledge the support of departmental staff and the use of open-source technologies such as Python, Flask, ReactJS, and MongoDB, which significantly contributed to the successful development and evaluation of the proposed system.

REFERENCES

[1] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning forimage-basedplant diseasedetection,” Frontiersin PlantScience,vol. 7, p.1419,2016.
[2] H. Al-Yakoob and H. D. Sherali, “A mixed-integer programming approach to a class timetabling problem,” Computers & Operations Research, vol. 33, no. 5, pp. 1276–1305, 2006.
[3] S. Abdullah, E. K. Burke, and B. McCollum, “A hybrid evolutionary approach to the university course timetabling problem,” IEEE Transactions on Evolutionary Computation, vol. 14, no. 6, pp. 1000–1011, 2010.
[4] A. Schaerf, “A survey of automated timetabling,” Artificial Intelligence Review, vol. 13, no. 2, pp. 87–127, 1999.
[5] K. Murray, “Automated timetable generation and workload allocation in higher education,” International Journal of Computer Applications, vol. 42, no. 15, pp. 12–18, 2012.
[6] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner’s Approach, 8th ed. New York, NY, USA: McGraw-Hill, 2015.
[7] M. Sandhu and P. Samarati, “Access control: Principles and practice,” IEEE Communications Magazine, vol. 32, no. 9, pp. 40–48, 1994.
[8] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, “Proposed NIST standard for role-based access control,” ACM Transactions on Information and System Security, vol. 4, no. 3, pp. 224–274, 2001.
[9] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2002.
[10] T. Müller, “ITC2007 solver description: A hybrid approach to university course timetabling,” Annals of Operations Research, vol. 172, no. 1, pp. 429–446, 2009.
[11] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, “A survey of search methodologies and automated system development for examination timetabling,” Journal of Scheduling, vol. 12, no. 1, pp. 55–89, 2009.
[12] S. R. Sahu and P. K. Mishra, “Automated academic timetable generation using genetic algorithm,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 5, pp. 271–276, 2017.
[13] A. A. El-Moursy, “Web-based academic scheduling system using constraint-based optimization,” International Journal of Computer Applications, vol. 179, no. 20, pp. 1–6, 2018. ComputersandElectronicsinAgriculture, vol.147,pp.70–90,2018.
[14] M. Wren, “Scheduling, timetabling and rostering—A special relationship?,” Lecture Notes in Computer Science, vol. 1153, pp. 46–75, 1996.
[15] A. Gunawan, K. M. Ng, and K. P. Wong, “A hybrid metaheuristic approach for solving the university course timetabling problem,” Applied Soft Computing, vol. 13, no. 1, pp. 308–316, 2013.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY