

Digital Academic Framework: A Web-Based Academic Management Platform with Automated Timetable Generation

P. Anlet Pamila Suhi¹, Charvi Reddy M², Chintamani Abhilash², Mohammed Taahir²

¹Assistant Professor, Dept. of Information Science and Engineering
Sri Sairam College of Engineering, Bangalore

²Final Year BE (Information Science and Engineering), Sri Sairam College of Engineering, Bangalore

DOI: doi.org/10.34293/iejcsa.v4i4.166

Abstract - Preparing a college timetable involves more than placing subjects into empty periods. Faculty availability, semester requirements, laboratories, rooms, working days, and repeated changes must all be considered at the same time. In many colleges, this work is still handled through spreadsheets and repeated manual checking. The Digital Academic Framework presented in this paper brings timetable generation together with basic academic management in one web application. The platform provides separate teacher and student views, Firebase-based authentication, academic data management, timetable handling, notices, notes, faculty information, and feedback or remarks. The scheduling part is treated as a constraint problem in which faculty, class or semester, room, and period conflicts must be avoided. The system follows a simple workflow: collect academic inputs, validate them, generate a timetable, check conflicts, store the approved result, and make the relevant information available to users. The work focuses on practical integration rather than claiming a new optimization algorithm. The paper describes the system architecture, workflow, scheduling model, implementation approach, testing plan, limitations, and possible future extensions.

Keywords— academic management system, college timetable, timetable generation, faculty management, constraint validation, Firebase, student portal, web application.

INTRODUCTION

Timetabling is a recurring institutional planning task: teaching events must be placed into a finite set of periods while respecting resource and participant constraints. University course timetabling is recognized as a difficult combinatorial optimization problem, with interacting hard and soft constraints and substantial variation between institutions [1], [5]. [1, 2] Hard constraints represent assignments that cannot coexist, such as a faculty member or class being allocated to two events at the same time; soft constraints represent preferences or quality goals that may be traded off when a fully preferred schedule is infeasible.

Manual scheduling can require repeated coordination among academic staff and revisions when assignments overlap or inputs change. Research on course scheduling has therefore explored constraint programming, integer programming, local search, and metaheuristics; these methods demonstrate that automation can address complex assignment problems, but their formulations and data requirements differ [1], [3]–[6].

Recent real-world benchmarks have broadened the scheduling problem beyond time assignment to include room allocation and student sectioning, emphasizing the complexity of operational university scheduling [7], [8]. [3, 4] Yet algorithmic scheduling alone does not provide the notices, notes, feedback, and role-specific access that form part of everyday academic administration.

A complementary need is timely, centralized access to academic information. Web-based academic information systems have been proposed to organize academic records and services across institutional user groups, while cloud-backed applications demonstrate patterns for authentication, database access, and file storage [9]–[11]. A web platform can make information available through distinct teacher and student workflows; however, simply digitizing data entry does not ensure that timetable assignments satisfy scheduling constraints.

The Digital Academic Framework addresses these concerns by combining timetable management and generation with academic information services in a single web-based application. The project description specifies Firebase Authentication, a cloud database, and storage where applicable, alongside administrator/teacher and student interfaces. This paper makes four contributions: (i) it formalizes the project scope without attributing unspecified technologies or results; (ii) it describes the integrated modules and data flow; (iii) it explains a constraint-aware timetable workflow while marking the unspecified algorithmic details; and (iv) it defines functional and quantitative evaluation measures that can be populated only after system testing.

LITERATURE REVIEW

A. Automated Timetable Generation and Course Scheduling

Course timetabling research has developed a range of methods because assignments combine discrete resources, periods, and competing requirements. The widely used curriculum-based course timetabling formulation assigns course events to periods and evaluates hard and soft constraints, providing a standardized basis for comparison [2], [3]. The formulation is useful for algorithm comparison, but no benchmark captures every institution's rules; the problem definition and constraint set must therefore be adapted to local academic practice [1], [3]. [1]

Constraint-programming approaches model feasible assignments explicitly and use propagation to prune invalid partial schedules. An adjustable constraint-propagation framework has compared search approaches and demonstrated the relevance of constraint propagation to course-scheduling instances [4]. This approach offers a principled representation of feasibility, although its performance and usability depend on careful model construction and institution-specific input. [5]

Optimization-based research also includes mixed-integer programming and hybrid methods. A graph-based mixed-integer model for the 2019 International Timetabling Competition incorporates class time and room assignment together with student sectioning,

using conflict-graph structure to represent constraints [7]. A related parallelized matheuristic combines multiple solution methods and was evaluated on the competition's benchmark instances [8]. These studies provide rigorous techniques and reproducible benchmark settings, but the computational model is not equivalent to a deployed academic-management portal: user authentication, notices, notes, feedback, and ongoing information maintenance are separate application concerns.[3, 4]

Metaheuristics, including local search and hyper-heuristics, have also been applied to course timetabling. A survey of university course timetabling methods notes the prominence of metaheuristics and hybrid approaches and identifies limited flexibility in transferring some published methods to real-world environments [1]. [1] A 2022 course-timetabling study applies a graph-coloring initial solution with Late Acceptance Hill Climbing in a hyper-heuristic framework, incorporating lecturer preferences and semester-specific data [5]. Such methods can optimize solutions against defined objectives; their results cannot be assumed for the present system because its implemented algorithm and objective function have not been supplied.

The 2019 competition formulation and subsequent literature review highlight a persistent translation challenge: practical instances vary, and solver generality and consistency of problem definitions remain concerns [6]–[8]. [1, 6] Benchmark performance should thus be distinguished from evidence that a particular institution's timetable can be generated reliably from its own data. For the proposed framework, that distinction motivates explicit constraint documentation, feasibility validation, and testing with authentic institutional configurations.

B. Academic Information and Web/Cloud Systems

Web-based academic systems have been studied as means of centralizing academic information and supporting interactions among administrators, teachers, and students [9], [10]. Project implementations demonstrate possible features such as publishing academic materials and notices and maintaining role-specific workflows, but reported features and architectures differ substantially; they do not establish that timetable generation is conflict-aware unless scheduling constraints are explicitly described and tested [9], [10]. [7, 8]

Firebase-related application studies describe cloud services for authentication, database synchronization, and storage, and demonstrate their use in education-oriented applications [11], [12]. These reports provide implementation precedents rather than evidence that Firebase itself performs timetable optimization. In this project, Firebase is treated as an application backend for identity and academic data, while scheduling logic belongs to the application's business layer. Cloud storage and real-time behavior must be verified against the actual configuration and security rules.[9]

Overall, the literature provides strong foundations for constraint-based scheduling and separate precedents for web-based academic information management. The

approaches solve different portions of the institutional problem: specialized timetable solvers focus on assignments and objectives, whereas academic portals focus on data access and workflows. The project's central design question is how to connect those functions while retaining explicit feasibility checks and role-appropriate access.

RESEARCH GAP

The reviewed work supports a bounded gap rather than a claim that no integrated systems exist. Timetabling studies provide increasingly rich constraint and optimization formulations, including student sectioning and room assignment [3], [6]–[8], but their principal contributions are solver models and benchmark results. Conversely, academic information system studies emphasize centralized access, administrative workflows, and digital resources [9], [10], while Firebase application reports focus on cloud-backed implementation [11], [12]. These strands do not, in the sources reviewed here, establish a single project-specific implementation that combines conflict-aware timetable generation with the listed faculty, subject, class/semester, notice, notes, and feedback functions under teacher and student roles.[6]

A second gap concerns implementation transparency. Published solver results depend on an explicit constraint set, objective function, input instance, and evaluation protocol [1], [7], [8]. The supplied project description identifies relevant scheduling dimensions but does not specify the algorithm, hard/soft constraint definitions, optimization objective, or measured performance. Consequently, the project can be framed as an integrated conflict-aware scheduling platform, but its algorithmic novelty, comparative advantage, generation quality, and scalability remain unverified. This paper addresses the architectural and methodological integration gap and identifies the evidence required to substantiate operational claims.

PROBLEM STATEMENT

Educational institutions need to assign teaching activities to available time slots while avoiding incompatible allocations of faculty, subjects, and classes, and to provide academic information through appropriate user interfaces. Manual or fragmented processes make it difficult to coordinate these functions consistently: timetable edits may not propagate to all users, and academic notices or resources may remain disconnected from schedule management. Specialized optimization models, in turn, require explicit local constraints and do not by themselves deliver the broader academic workflows described in this project [1], [6], [9].

The research problem is therefore to design and implement a web-based academic framework that (a) represents the institution's scheduling inputs, (b) generates or manages candidate timetables with conflict validation against configured constraints, and (c) integrates timetable publication with faculty, subject, class/semester, notice, notes, and feedback services for teacher and student roles. The system aims to reduce avoidable

scheduling conflicts and manual administrative effort. Whether it achieves these aims in practice must be established through implementation-specific testing rather than assumed from the architecture.

PROPOSED SYSTEM

The Digital Academic Framework is a centralized web application for academic data management and timetable access. Its scope, as provided, includes administrator/teacher and student roles, faculty and subject records, class/semester configuration, timetable generation and management, notices, academic notes/resources, feedback, and Firebase-based authentication and data services. The system is not described as employing machine learning or another predictive method; timetable generation is treated as a constraint-based scheduling function.

The system objectives are to: (1) centralize the academic data needed for timetable preparation; (2) support generation or management of schedules while checking configured conflicts; (3) publish the approved timetable to the relevant user groups; (4) provide teacher and student access according to role; and (5) maintain related academic information through the same application.

Teacher/administrator module: Authorized staff manage faculty, subjects, classes or semesters, timetable inputs, notices, notes, and feedback. The precise division of privileges between administrators and teachers must be determined by the deployed role policy. A publication step should distinguish a draft schedule from the timetable visible to students, preventing incomplete schedules from being treated as final.

Student module: Authenticated students access the timetable relevant to their class or semester and view published notices and available academic resources. They can submit feedback through the provided feature. Student access should be read-limited to the relevant academic data; the project description does not specify student record identifiers or the exact filtering model.

Academic data management: Faculty, subject, and class/semester records provide the entities required for schedule configuration. Their identifiers and relationships should be validated before generation. Notices and notes are associated with the appropriate audience or academic grouping where such relationships are implemented. File-based resources may use Firebase Storage; structured records belong in the configured cloud database.

Timetable management and conflict handling: A timetable consists conceptually of teaching events assigned to time slots and associated with faculty, subject, and class/semester. At minimum, validation should prevent a faculty member or class from being assigned to overlapping events and should check any subject-specific scheduling rules supplied by the institution. Room conflicts are outside the stated project context unless room entities are actually implemented. If no feasible complete schedule exists, the

interface should report unresolved assignments rather than silently publish an invalid timetable. The exact search and conflict-resolution algorithm is not specified and must be documented from the implementation.

Compared with a paper- or spreadsheet-centered workflow, the integrated platform is intended to place timetable operations and related academic resources in one access environment. That is an architectural benefit, not a measured result; improvements in workload, usability, conflict rates, and data freshness require evaluation.

SYSTEM ARCHITECTURE

The architecture comprises a browser-based user interface, authentication and role-control services, an application/business-logic layer, timetable constraint validation, and Firebase cloud services. The teacher/administrator interface exposes data-management and timetable workflows. The student interface exposes authorized timetable and academic information. Both interfaces communicate with business logic that validates requests and determines which operations are permitted for the authenticated role.

Firebase Authentication verifies user identity. After sign-in, the application resolves the user's role and applies authorization checks before permitting access to administrative or student functions. Authentication proves identity; it does not by itself enforce data-level authorization. Firestore/database security rules and server-side or trusted application checks must be configured so that users cannot bypass interface restrictions by directly querying data. The exact enforcement arrangement is an implementation detail requiring inspection and security testing.

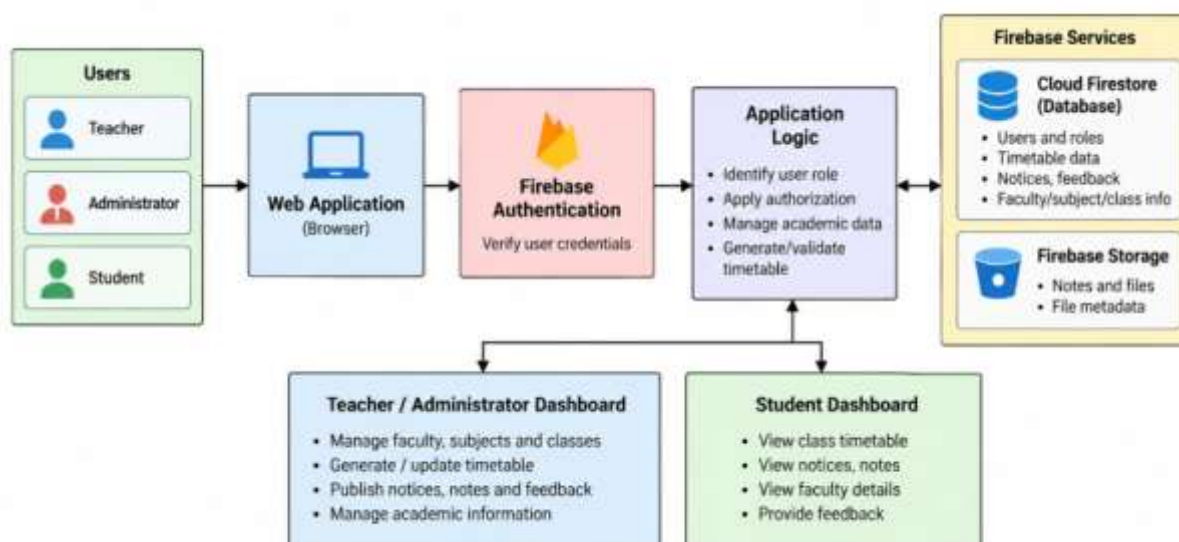


Figure 1 System Architecture

The application logic reads configured faculty, subject, class/semester, working-day, and time-slot data. It constructs or receives candidate timetable assignments and applies

conflict checks according to the institution's constraints. The accepted schedule and relevant metadata are written to the cloud database. Where notes or other files are stored, the application stores file objects in Firebase Storage and maintains their metadata and access references in the database. Notices, feedback, and structured academic records are persisted in the database. After publication, the student interface retrieves the timetable and associated academic information allowed for that role.

Block-diagram description: Draw two client blocks—"Teacher/Administrator Web Interface" and "Student Web Interface"—connected to a "Web Application / Business Logic" block. Connect both interfaces through an "Authentication and Role Authorization" block to "Firebase Authentication." Within the business-logic block show "Academic Data Validation," "Timetable Generation/Management," and "Conflict Validation." Connect these functions bidirectionally to "Cloud Firestore / Academic Database." Connect "Notes and Uploaded Resources" to "Firebase Storage," with metadata links to Firestore. Show arrows from teacher administration inputs to validation and timetable generation, from generated schedule to database publication, and from the database to student and teacher read views. Add a feedback path from student interface to the database and a notice/resource publication path from teacher interface to authorized users. Mark security rules and role checks at database access boundaries. This diagram represents the project's stated components; it should be adjusted to match the actual deployment topology.

METHODOLOGY

The proposed development and scheduling methodology consists of the following stages.

1. **Requirement identification:** Elicit institutional requirements from the administrative and teaching workflows and establish the expected student-facing information. Convert each function—faculty, subject, class/semester, timetable, notices, notes, feedback, authentication—into testable requirements. The project description supplies the feature scope but not a completed requirements specification.
2. **Role definition:** Define administrator, teacher/faculty, and student permissions. Specify which roles may create, edit, publish, view, or remove each data type. Validate authorization both at the application layer and, where applicable, through Firebase security rules.
3. **Academic data configuration:** Enter and validate faculty, subjects, classes/semesters, working days, and available time slots. Each teaching event must have enough linked data to determine its eligible assignments. The exact data schema, identifiers, and mandatory fields must be documented by the implementation team.
4. **Constraint specification:** Express institutional rules as explicit predicates over assignments. Examples supported by the project scope include faculty unavailability,

overlapping allocations for the same faculty member, overlapping events for the same class, and any required subject/class assignment constraints. Distinguish hard constraints, which define feasibility, from optional preferences if these are implemented. Do not infer room constraints, student preferences, or optimization objectives unless the system actually captures them.

5. **Timetable generation:** Generate a candidate assignment of teaching events to working-day/time-slot pairs using the implemented procedure. The project information does not identify whether generation uses backtracking, graph coloring, integer programming, a heuristic, or another method; no such algorithm is asserted here. A reproducible implementation description should state input representation, candidate-selection strategy, ordering, termination conditions, and how infeasibility is reported.
6. **Conflict detection and resolution:** Validate each candidate against the specified hard constraints, including faculty and class overlaps. Resolve a detected conflict by reassigning the event or returning it to an unresolved state for authorized staff intervention. If the implemented generator does not automatically resolve every conflict, describe it as conflict detection/assisted scheduling rather than fully automatic optimization. Record validation outcomes to support audit and evaluation.
7. **Persistence and publication:** Save validated timetable data in the cloud database with term/class association and publication status. Staff review and publish the schedule. Draft changes should not overwrite the student-visible schedule until approved, if versioning or staged publication is implemented.
8. **Role-based access and academic services:** Teachers manage authorized academic information and students retrieve published information. Notices, notes/resources, and feedback follow the same authentication and authorization controls. File upload/download operations use Firebase Storage only where the application implements them.
9. **Feedback and maintenance:** Review feedback, correct academic data, and regenerate or revise timetables when inputs change. Preserve the distinction between data correction and schedule optimization; every revision should be revalidated before publication.

This is a constraint-based process description, not a claim about a specific solver. Its effectiveness depends on the completeness of institutional rules and the correctness of their implementation.

IMPLEMENTATION & EVALUATION

The documented technology choices are a web application and Firebase services for authentication, database management, and storage where applicable. The project

description does not specify the front-end framework, programming language, exact Firebase database product, hosting service, database schema, or deployment configuration. These details must be added from the source code and deployment records before this paper is represented as a completed implementation report.

The teacher/administrator workflow begins with authentication, then configuration or maintenance of faculty, subjects, classes/semesters, working days, and slots. The staff member initiates timetable generation or management, reviews conflict-validation feedback, corrects unresolved inputs or assignments, and publishes a validated schedule. The student workflow begins with authentication, retrieves the published timetable associated with the relevant class or semester, and accesses authorized notices and notes; feedback is submitted through the designated module. These workflows reflect the specified project features, but their implementation status and usability have not been independently verified.

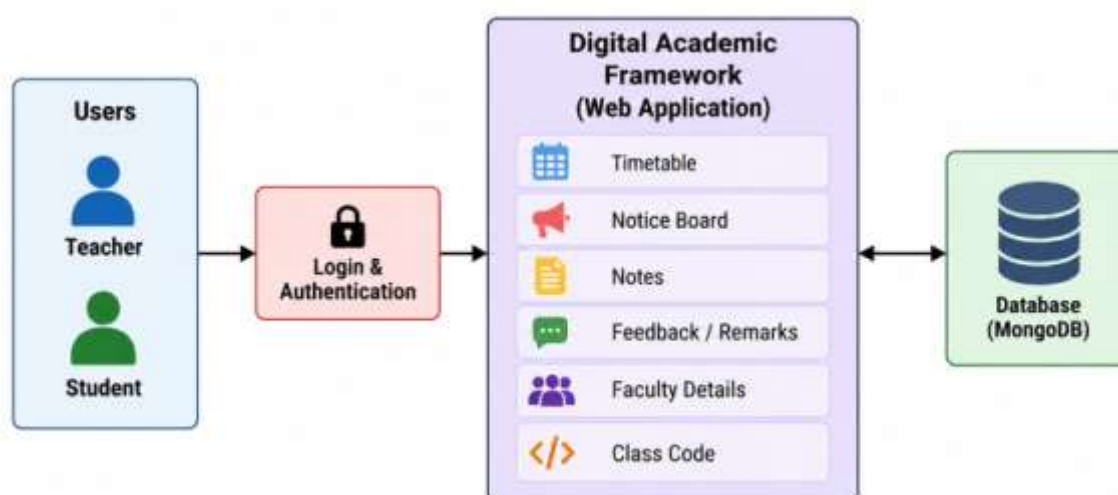


Figure 2 Simple Block Diagram

Evaluation should separate functional correctness, schedule feasibility, operational performance, security, and user experience. Functional testing should exercise successful and invalid authentication, role-specific access, record creation and update, schedule generation, conflict detection, timetable publication, notice/resource access, and feedback submission. Conflict tests should include deliberately overlapping faculty assignments and class assignments, unavailable faculty, insufficient slots, missing required data, and schedule edits after publication. Security testing should attempt unauthorized reads and writes, including direct database access. Usability evaluation requires representative teachers and students and a documented instrument; no such study or result was supplied.

The following are **proposed measurements, not results**:

Evaluation dimension	Measure to collect	Measurement procedure	Actual result
Generation time	Elapsed time from valid input submission to generated candidate	Repeat across documented timetable sizes and configurations; report median and spread	Not provided
Conflict count	Number of hard-constraint violations in generated and published schedules	Independent validator checks faculty/class overlaps and configured constraints	Not provided
Successful generation rate	Proportion of valid input instances yielding a complete feasible schedule	Test a documented suite including feasible and infeasible cases	Not provided
Conflict-resolution behavior	Correct detection and handling of each seeded conflict	Functional test cases with expected outcomes	Not provided
Response time	End-to-end latency for login, timetable retrieval, and academic-data operations	Measure under stated network, user load, and deployment conditions	Not provided
Functional coverage	Passed, failed, and blocked cases by module	Maintain traceable requirements-to-test-case matrix	Not provided
Usability	Task completion, observed errors, and participant feedback	Conduct a defined usability study with documented participants and instrument	Not provided
Access control	Unauthorized access attempts blocked by application and database rules	Role-based positive and negative security tests	Not provided

No performance values, user-study data, dataset description, or measured conflict count are available in the project information. Therefore, system effectiveness cannot yet be quantified, and no comparison with manual scheduling or other solvers is claimed. A strong evaluation would disclose the instance sizes and constraint definitions, baseline process, hardware/network conditions, repeated-run protocol, and whether any manual changes were required after generation. Algorithmic comparison should be considered only after the implemented algorithm and a reproducible test set are identified.

CONCLUSION

This paper has presented the Digital Academic Framework as a web-based platform intended to integrate conflict-aware timetable generation with academic management for teacher/administrator and student users. The described scope combines faculty, subject, class/semester, timetable, notices, notes/resources, feedback, and Firebase-backed authentication and data services. Its contribution is the system-level integration of

scheduling and academic information access, supported by a methodology that makes scheduling constraints, conflict checks, authorization, persistence, and publication explicit.

The available project description does not establish a particular scheduling algorithm, detailed constraint model, deployed security configuration, measured timetable quality, or user evaluation. The system should therefore be regarded as a project framework whose practical benefits remain to be demonstrated through reproducible functional, conflict, performance, security, and usability testing. Future implementation work should document the exact generation procedure, distinguish hard constraints from preferences, establish schedule versioning and infeasibility handling, and report results on real institutional instances. These steps would allow the intended reduction in scheduling conflicts and manual effort to be assessed without overstating the evidence.

REFERENCES

- [1] H. Algethami and W. Laesanklang, "A Mathematical Model for Course Timetabling Problem With Faculty-Course Assignment Constraints," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3103495.
- [2] H. Alghamdi, T. M. Alsubait, H. Alhakami, and A. Baz, "A Review of Optimization Algorithms for University Timetable Scheduling," *Engineering, Technology & Applied Science Research*, 2020, doi: 10.48084/etasr.3832.
- [3] N. L. Abdul Aziz, A. A. Ariffin, and S. Sulaiman, "An Integer Linear Programming Approach to a University Course Timetabling Problem," *Mathematics Letters*, 2022. 
- [4] M. Chen, S. Sze, S. Goh, N. R. Sabar, and G. Kendall, "A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3100613.
- [5] G. Colajanni, "An Integer Programming Formulation for University Course Timetabling," 2019.
- [6] S. Mohite, A. Shah, and A. B. Kasture, "Mobile Based Notice Board & College Management System using Firebase," 2018.
- [7] M. Ridwan and A. R. Yusuf, "Sistem Informasi Akademik Berbasis Framework Bootstrap untuk Pengelolaan Data Akademik dan Administrasi," *SULIWA: Jurnal Multidisiplin Teknik, Sains, Pendidikan dan Teknologi*, 2025, doi: 10.62671/suliwa.v2i2.66.
- [8] E. Rappos, E. Thiémarc, S. Robert, and J.-F. Hêche, "A Mixed-Integer Programming Approach for Solving University Course Timetabling Problems," *Journal of Scheduling*, 2022, doi: 10.1007/s10951-021-00715-5.
- [9] P. S. Saud, A. Chand, N. K. Chaurasia, and L. P. Bastola, "Pathshala: Student Information Management using Cutting-Edge Technology," *Journal of Soft Computing Paradigm*, 2023, doi: 10.36548/jscp.2023.3.003.

- [10] J. Suárez-Rodríguez, J. Pina, L. Malagón-Alvarado, V. Blanco, M. Correa, D. La Rosa, J. Valderrama, and C. A. Vega-Mejía, "An Optimization Model for University Course Timetabling: A Colombian Case Study," ICPR-Americas, 2020.