A Constraint-Based Academic Timetable and Conflict Detection System for Dynamic Class Rescheduling in a Higher Education Context

Matimba Handabile; Kayaza Ngoma

Department of ICT, School of Business Studies, Kwame Nkrumah University, Kabwe, Zambia

DOI: https://doi.org/10.51584/IJRIAS.2026.11060113

Received: 11 June 2026; Accepted: 16 June 2026; Published: 27 June 2026

ABSTRACT

Academic timetabling is a complex institutional process that becomes particularly difficult when timetable changes are required after the initial schedule has been published. In many universities, post-publication rescheduling is still coordinated through fragmented manual channels such as class representatives, messaging groups, phone calls, and informal notices. These practices increase the likelihood of delays, venue uncertainty, communication failure, and hidden timetable conflicts. This study designed, implemented, and evaluated an Academic Timetable and Conflict Detection System for a higher education context, with emphasis on dynamic class rescheduling rather than initial timetable generation alone. The system uses a Constraint Satisfaction Problem (CSP) approach to generate conflict-free timetable allocations and a localised dynamic validation mechanism to support lecturer-initiated rescheduling. It checks lecturer, venue, and programme-level conflicts, supports grouped courses shared across programmes, updates timetable state after successful changes, and notifies affected students through in-application and email alerts. Evaluation combined scenario-based comparison using documented scheduling incidents, functional testing, performance observation, and limited usability assessment. Results showed that grouped course rescheduling which previously required at least two hours and forty-three minutes of manual coordination was completed in an average lecturer-visible time of 12.26 seconds. The rescheduling preview mechanism produced an average response time of 99.6 ms, and all eight functional test cases were successfully executed. The findings indicate that integrating CSP-based scheduling with dynamic conflict-aware rescheduling can improve timetable reliability, responsiveness, and communication efficiency in academic environments.

Keywords: academic timetabling; constraint satisfaction problem; dynamic rescheduling; conflict detection; higher education information systems; timetable notification

INTRODUCTION

University timetabling is a core administrative function in higher education because it coordinates lecturers, courses, venues, programmes, student groups, and teaching periods into a usable academic schedule. Within the literature, these problems are formally categorised into three main domains: High-School (HTT), University Course (CTT), and University Examination (ETT) timetabling; this study specifically addresses the CTT domain (Ceschia, Di Gaspero and Schaerf, 2022). Constructing a valid timetable is notoriously difficult because a valid timetable must satisfy multiple interacting constraints, including room availability, lecturer allocation, programme structure, student enrolment patterns and temporal restrictions. As institutions expand and programmes share teaching spaces and lecturers, the complexity of timetable management increases significantly (Chen et al., 2021; Ceschia et al., 2023), a challenge often intensified by a paradigm shift in education logistics where institutions must manage strictly limited budgets and share diminishing resources like classrooms more efficiently (Vrielink, et al., 2019).

Research on university course timetabling has produced a wide range of computational approaches, including heuristic optimisation, graph-based methods, mathematical programming, metaheuristics, and constraint-based models (Bashab et al., 2023; Premananda & Muklason, 2021). These approaches have contributed substantially

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

to the generation of feasible and optimised academic timetables. However, many practical timetable difficulties occur after an initial timetable has already been released. Lecturers may become unavailable, venues may change, shared courses may need to be moved, and institutional events may disrupt the original schedule. In such cases, the problem is not simply to generate a new timetable from scratch, but to adjust an existing timetable while preserving consistency across the affected academic environment.

Within the institutional context considered in this study, timetable changes were often coordinated through informal channels such as class representatives, WhatsApp messages, phone calls and physical notices. These channels may function in simple cases, but they are vulnerable to communication delays, inconsistent information, venue uncertainty and undetected timetable clashes. A lecturer may identify a preferred replacement slot, yet the change may still require manual confirmation from several student groups or representatives before being communicated more widely. Where shared courses are involved, separate groups may also hold different timetable representations for the same academic session.

This study addresses the operational phase of academic timetable management by designing and implementing an Academic Timetable and Conflict Detection System that combines CSP-based timetable generation with dynamic, conflict-aware rescheduling. The purpose is to provide a structured system through which lecturers can initiate class changes, view available alternatives, avoid invalid venue or schedule allocations and trigger immediate notifications to affected students. Unlike timetable systems that focus mainly on initial timetable construction, the system presented here emphasises controlled adaptation after timetable publication.

The study makes four main contributions. First, it demonstrates a practical design for lecturer-driven timetable rescheduling that validates changes before they are committed. Second, it introduces grouped course handling so that shared academic sessions across programmes are updated consistently. Third, it applies override-aware timetable state management so that rescheduled slots are released and reused correctly. Fourth, it integrates timetable communication through in-application and email notifications, reducing dependence on informal communication channels.

LITERATURE REVIEW

Academic Timetabling and Existing Systems

Academic timetabling has been widely recognised as a complex combinatorial optimisation problem because of the number of resources, constraints, and institutional rules that must be coordinated (Chen et al., 2021; Ceschia et al., 2023). Existing academic timetabling systems range from open-source platforms to enterprise scheduling systems. Open-source systems such as UniTime and FET provide accessible timetable generation and scheduling functionality, while commercial platforms such as Ad Astra/Infosilem and Scientia Syllabus+ support broader institutional scheduling administration and resource planning (Müller, 2024; Müller et al., 2025).

A key pattern in existing systems is that they are strong in timetable generation, resource allocation, and institutional scheduling support. However, comparatively less emphasis is placed on localised, lecturer-driven rescheduling workflows in which a targeted change must be evaluated quickly, applied safely, and communicated directly to affected students. This is important because the operational reality of academic timetable management involves frequent post-publication disruptions.

Table 1. Functional emphasis of selected academic timetabling systems.

System type	Examples	Typical strengths	Observed limitation for this study
Open-source/research-oriented systems	UniTime; FET	Cost-effective timetable generation, flexible deployment, support for constraint-aware scheduling.	Limited documented emphasis on lecturer-driven post-publication rescheduling and direct student notification.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

Commercial/enterprise systems	Ad Astra/Infosilem; Scientia Syllabus+	Institutional scheduling administration, large-scale planning, resource coordination.	Implementation complexity and enterprise workflow orientation may be less suited to focused local rescheduling problems.
Proposed institutional prototype	Academic Timetable and Conflict Detection System	Dynamic rescheduling, real-time conflict validation, grouped course handling, student notifications.	Prototype evaluated within a limited institutional dataset and requires scalability testing before production deployment.

Computational Approaches to Timetabling

Several computational approaches are used in timetabling (Chen et al., 2021; Ceschia et al., 2023). Heuristic methods are widely applied because they can construct feasible timetables efficiently without exhaustively searching the entire solution space. Graph colouring models represent timetable entities as vertices and conflicts as edges, allowing time slots to be assigned so that conflicting events do not overlap. Metaheuristic methods, including genetic algorithms, simulated annealing, tabu search, and hybrid approaches, are useful for exploring large solution spaces and producing near-optimal solutions (Bashab et al., 2023).

Constraint Satisfaction Problem (CSP) models provide a particularly relevant foundation for academic timetabling because they represent scheduling as a set of variables, domains, and constraints (Qefalija et al., 2024; Ďuriš, 2020). In a timetable setting, variables may correspond to class sessions, lecturers, rooms, or time slots; domains represent allowable scheduling options; and constraints define conditions that must be satisfied for a valid timetable. Hard constraints include preventing lecturer double-booking, avoiding venue clashes, and preventing overlapping classes for the same programme. Soft constraints may include preferences such as balanced distribution of teaching periods or lecturer time preferences.

Traditional CSP approaches are effective for generating valid timetables, but post-publication rescheduling introduces a dynamic problem. Recomputing the entire timetable for every small change may be inefficient and disruptive. Dynamic Constraint Satisfaction Problem (DCSP) thinking is therefore useful because it supports localised adaptation when variables, constraints, or assignments change over time (Qefalija et al., 2024). In this study, CSP is used for initial constraint-based timetable generation, while localised dynamic validation is used for rescheduling an affected class without reconstructing the entire timetable. Formally, a Constraint Satisfaction Problem (CSP) consists of a set of variables, a domain of values for each variable, and a set of constraints that the assignment must satisfy simultaneously (Russell & Norvig, 2021)

Recent 2026 implementations of Web-Based Lecture Timetable Scheduling Systems (WLTSS) have demonstrated that using a React and Node.js stack can significantly improve scheduling transparency and student-lecturer communication in these environments (Abubakar, et al., 2026). Furthermore, faculty-level scheduling has been optimised using the Timefold Solver to handle complex, multi-objective constraints while ensuring room capacity and teacher availability are maintained (Diallo & Tudose, 2024)

METHODOLOGY

Research Design and Context

The study adopted an applied Design Science Research (DSR) approach, which seeks to extend organisational capabilities by creating and evaluating innovative technological artefacts (Hevner et al., 2004). The research followed the six-step Design Science Research Methodology (DSRM) problem identification, objective definition, design and development, demonstration, evaluation, and communication to provide a rigorous framework for building a functional software solution for academic scheduling (Peffers et al., 2007). The central objective was not only to analyse an academic scheduling problem, but to design, implement, and evaluate a functional software artefact capable of improving timetable coordination. The project was conducted in the

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

context of the School of Business at Kwame Nkrumah University, with the initial implementation scoped to selected programmes rather than full enterprise deployment.

Requirements were derived from direct observation of existing timetable coordination practices and from documented scheduling communication records, including WhatsApp exchanges related to class rescheduling, venue confirmation, and shared course coordination. These incidents were used to identify recurring operational problems and were later used as comparative scenarios for evaluation.

3.2 Development approach and technology stack

The system was developed using a sequential Waterfall-oriented process (Royce, 1970), which arranges development into sequential phases of requirements, design, implementation, testing and prototype evaluation. . This approach was appropriate because the major functional requirements were stable: timetable generation, conflict detection, lecturer-driven rescheduling, grouped course coordination, and automated notification of affected students.

The system was implemented as a three-tier web application. The presentation layer provides browser-based interfaces for administrators, lecturers, and students. The application layer contains timetable generation, conflict detection, rescheduling, grouped course handling, publication control, and notification logic. The data layer uses a relational database to store users, programmes, courses, venues, timetable entries, course groups, enrolments, reschedules, and notifications.

Table 2. Technology Stack Used in the Prototype.

Component	Technology	Purpose in the system
Backend	PHP 8.2 and Laravel 12	MVC-based application logic, role-based functionality, email integration, and timetable processing.
Frontend	Laravel Blade, Tailwind CSS, JavaScript, Fetch API	Dynamic role-specific interfaces, asynchronous preview generation, and interactive rescheduling feedback.
Database	MySQL	Relational storage for academic entities and scheduling relationships.
Build/development environment	Vite, XAMPP, Visual Studio Code	Local development, asset compilation, prototype testing, and demonstration.

Scheduling Model and Constraint Validation

For initial timetable generation, class sessions are treated as CSP variables. Possible combinations of time slots and venues form the assignment domain. The system uses backtracking search with Minimum Remaining Values (MRV) heuristic ordering to assign sessions while satisfying hard constraints, reflecting common CSP search and pruning principles described in timetabling literature (Ďuriš, 2020; Qefalija et al., 2024). The constraints include lecturer availability, venue occupancy, programme-level timetable consistency, and grouped course coordination.

For timetable rescheduling, the system uses localised constraint validation. Instead of regenerating the full timetable when a lecturer moves one class, the system evaluates the affected class session against the current active timetable state. This reduces unnecessary recomputation and supports interactive decision-making. The conflict detection mechanism is override-aware: once a class is rescheduled, the new allocation becomes the active assignment and the original slot is released for future availability checks. This prevents the system from treating historical timetable entries as permanently occupied.

Evaluation Design

Evaluation combined four complementary methods. Scenario-based comparative evaluation compared documented manual scheduling incidents against equivalent workflows supported by the prototype. Performance

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

analysis measured timetable generation, rescheduling preview generation, full rescheduling completion, and notification behaviour. Functional testing used predefined test cases to verify timetable generation, conflict detection, valid and invalid rescheduling, conflict visualisation, and rescheduled slot reuse. Limited usability observation involved five student participants interacting with the student interface, with an additional supervisor demonstration of the lecturer-side workflow.

SYSTEM DESIGN AND IMPLEMENTATION

User Roles and Functional Scope

The system supports three primary user roles: administrator, lecturer, and student. Administrators manage programmes, courses, venues, lecturers, timetable records, timetable generation, and timetable publication. Lecturers view personalised teaching schedules, claim or manage assigned courses where applicable, and initiate class rescheduling subject to conflict validation. Students view personalised timetables and receive notifications when their schedules are affected by successful timetable changes.

Table 3. Core Functional Scope of the Proposed System.

Functional area	Description
Timetable generation	Generates academic timetables based on defined scheduling constraints.
Personalised timetable access	Displays role-specific timetable views for lecturers and students.
Lecturer rescheduling	Allows lecturers to initiate class changes through an interactive interface.
Conflict detection	Checks lecturer, venue, and programme-level conflicts before committing changes.
Grouped course handling	Treats shared courses across programmes as linked scheduling units.
Override-aware state management	Releases the original slot after successful rescheduling and treats the new slot as active.
Automated communication	Sends in-application and email notifications to affected students after successful rescheduling.

Dynamic Rescheduling Workflow

The dynamic rescheduling workflow begins when a lecturer selects a scheduled class from the timetable interface. The system retrieves the relevant class, lecturer, programme, venue, and grouped course context. It then generates a preview grid that evaluates available scheduling alternatives against active constraints. The preview grid visually distinguishes feasible, warning, and high-conflict options. A recommended conflict-free option is highlighted where available.

After a lecturer selects a preferred time slot, the system restricts venue selection to valid venues for that scheduling context. The system then validates the proposed change against lecturer availability, venue occupancy, programme overlaps, and grouped course constraints. If the change is invalid, it is blocked and feedback is provided. If the change is valid, the system updates the timetable, synchronises grouped course entries, releases the old slot, commits the new active assignment, and dispatches notifications to affected students.

This workflow is designed to prevent two common problems in manual rescheduling: hidden conflicts and delayed communication. It also supports consistency across shared courses, which is difficult to manage through informal manual coordination.

RESULTS

Scenario-Based Comparative Evaluation

The first evaluation scenario concerned a grouped Hardware course delivered to multiple student groups under different course codes. In the manual process, the lecturer’s request to move the class required availability

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

confirmation through student representatives. Timestamped records showed that coordination began at 16:51 and final confirmation was only received at 19:34. This represented a minimum delay of approximately two hours and forty-three minutes, excluding further lecturer confirmation and wider student communication. The proposed system handled the equivalent situation by treating the shared academic session as a grouped scheduling unit and applying the change atomically across all affected schedules.

The second scenario concerned delayed timetable communication for the Entrepreneurship course. In one documented incident, a student enquiry regarding class timing was made at 19:54, but confirmation was only received at 05:57 the following morning; venue confirmation was communicated at 07:12, after the scheduled 07:00 start time. In a separate case, venue confirmation was delayed until 09:09, with campus clarification at 09:11. These incidents illustrate the risk of relying on informal message relay for authoritative timetable information. The system addressed this by providing direct timetable visibility and automated notifications.

The third scenario concerned venue allocation uncertainty. In the manual process, students sometimes had to identify available rooms reactively. One documented case required approximately forty-four minutes for venue confirmation, while another required approximately twenty-one minutes. In the system-supported process, venue availability is validated before a rescheduling action is committed, reducing the risk of venue clashes and class disruption.

Table 4. Summary of Scenario-Based Comparative Findings.

Scenario	Manual workflow issue	System-supported workflow	Observed improvement
Grouped course rescheduling	Manual confirmation through representatives; minimum delay of 2 hours 43 minutes.	Linked grouped schedules updated as one unit.	Reduced coordination time and intermediary dependence.
Delayed timetable communication	Confirmations sometimes arrived after the scheduled start time.	Students receive updated timetable and email/in-app notices.	Improved transparency and reduced uncertainty.
Venue allocation uncertainty	Venue confirmation took 21 to 44 minutes in documented cases.	System presents only valid venue options.	Prevents venue conflicts before disruption.

Performance Analysis

The prototype was evaluated using a dataset comprising nine programmes in the School of Business, representing approximately 390 scheduling variables across more than 180 courses. The CSP-based timetable generation process produced complete conflict-free timetables in an average range of four to seven minutes. This duration is acceptable for periodic administrative timetable generation but indicates that optimisation would be necessary for larger deployments.

The dynamic rescheduling workflow produced stronger real-time responsiveness. Preview generation was measured across five test runs with recorded times of 99 ms, 108 ms, 105 ms, 94 ms, and 92 ms. This produced an average preview response time of 99.6 ms, indicating near-instant feedback during interactive rescheduling. Full lecturer-visible rescheduling completion was measured across five test runs with observed times of 14.24 seconds, 10.81 seconds, 10.81 seconds, 14.15 seconds, and 11.31 seconds, producing an average completion time of 12.26 seconds. This includes constraint validation, grouped schedule coordination, database update processing, and notification triggering.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

Table 5. Performance Observations from Prototype Evaluation.

Metric	Observed result	Interpretation
Initial timetable generation	4 to 7 minutes for nine programmes, about 390 variables, and more than 180 courses.	Acceptable for administrative generation, but scalability optimisation is needed for larger deployments.
Rescheduling preview generation	Average of 99.6 ms across five runs.	Supports near-instant interactive decision-making.
Full rescheduling completion	Average lecturer-visible completion time of 12.26 seconds across five runs.	Substantially faster than manual coordination workflows that required minutes or hours.
Notification dispatch	Notifications triggered immediately after successful rescheduling.	Reduces dependency on manual communication channels.

Functional Testing

Functional testing confirmed that the system behaved correctly across all evaluated core operations. The test cases covered timetable generation, lecturer conflict detection, venue conflict detection, programme-level conflict detection, valid rescheduling, invalid rescheduling, conflict visualisation, and reuse of previously occupied slots after rescheduling. All test cases were successfully executed, confirming that the system enforced defined constraints and maintained timetable consistency during both generation and modification.

Table 6. Functional Verification Summary.

Test case	Function evaluated	Observed outcome
TC01	Timetable generation	Passed: generated valid timetable under defined constraints.
TC02	Lecturer conflict detection	Passed: prevented double-booking of lecturer.
TC03	Venue conflict detection	Passed: prevented simultaneous venue allocation.
TC04	Programme conflict detection	Passed: prevented overlapping classes for the same programme/year.
TC05	Rescheduling with valid slot	Passed: allowed valid timetable adjustment.
TC06	Rescheduling with conflict	Passed: blocked invalid change.
TC07	Conflict visualisation	Passed: displayed feasibility indicators in preview grid.
TC08	Rescheduled slot reuse	Passed: released old slot and allowed appropriate reuse.

Usability Observations

A limited usability evaluation involved five student participants interacting with the prototype under guided observational conditions. Participants performed representative tasks such as viewing personalised timetables, identifying schedule updates, and interpreting notification indicators. Observations indicated that participants were generally able to navigate the student interface without significant difficulty. The timetable display and update indicators were understandable with minimal explanation. The lecturer-side workflow was also demonstrated to the project supervisor, and the rescheduling process was judged functionally coherent, particularly in relation to conflict visualisation and guided timetable modification.

The usability evaluation was exploratory rather than formal. The small participant sample, supervised testing conditions, and local deployment setting limit the generalisability of the findings. Nevertheless, the observations suggest that the core workflows are understandable and provide a useful foundation for broader usability testing. Future formal assessments will employ heuristic evaluation to test the interface against canonical usability principles, such as minimising user memory load and ensuring simple, natural dialogues (Nielsen & Molich, 1990).

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

DISCUSSION

Addressing the Research Problem

The findings indicate that the developed system addresses the central operational problem identified in the study: the lack of a structured, conflict-aware, and timely mechanism for managing timetable changes after publication. Manual workflows created delays because decisions depended on message relay through representatives, repeated confirmation, and venue verification. The system reduced these dependencies by placing timetable validation and communication within a centralised digital workflow. This outcome is consistent with the Design Science perspective that evaluates research through the usefulness and effectiveness of the artefact in solving practical organisational problems (Hevner, et al., 2004). By addressing these operational gaps, the system functions as a purposeful artifact that balances research rigor with problem relevance to solve a practical organizational challenge (Hevner, et al., 2004).

The most significant improvement was observed in grouped course rescheduling. Shared academic sessions are particularly difficult to manage manually because a change affects multiple student groups and may be represented differently across timetables. By modelling grouped courses as linked scheduling units, the system preserved consistency and prevented partial updates. This is a meaningful contribution because it addresses a practical rescheduling issue that is not fully solved by initial timetable generation alone.

Suitability of CSP and Dynamic Validation

The results support the suitability of CSP principles for academic timetable management. Academic timetabling is constraint-driven, and valid schedules must satisfy multiple hard constraints simultaneously. CSP provides a formal way to represent and enforce these requirements. However, the study also shows that practical timetable management requires more than initial CSP-based generation. Post-publication disruptions require localised validation, state-aware updates, and immediate feedback.

The system's override-aware mechanism is particularly important. When a class is moved, the original slot must no longer be considered occupied by that class; otherwise, future availability checks may incorrectly block valid scheduling options. By updating the active timetable state after successful rescheduling, the system supports dynamic adaptation while maintaining constraint integrity. This demonstrates how CSP concepts can be extended into an operational timetable management workflow.

Communication as Part of Timetable Integrity

The evaluation also shows that timetable integrity is not only a computational issue. A conflict-free timetable is insufficient if stakeholders do not receive updated information promptly. Automated in-application and email notifications are therefore not peripheral features; they are part of the timetable management solution. By ensuring that students receive schedule updates after successful rescheduling, the system improves transparency and reduces the risks associated with informal message relay.

LIMITATIONS

The study has several limitations. First, the prototype was evaluated using a limited institutional dataset comprising nine programmes from the School of Business rather than the full university scheduling environment. Second, usability testing involved only five student participants under supervised conditions, and the lecturer workflow was demonstrated rather than tested through a large sample of active lecturers. Third, the system was evaluated in a local development environment rather than a live production setting, so concurrent multi-user access, network variability, server performance, and long-term operational stability were not assessed. Fourth, while scenario-based evaluation used documented scheduling incidents, some aspects of the manual baseline relied on operational experience rather than a formal institutional time-and-motion study. Finally, CSP-based scheduling may face scalability challenges as the number of variables, constraints, and grouped course relationships increases.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VI June 2026

CONCLUSION AND RECOMMENDATIONS

This study presented the design, implementation, and evaluation of an Academic Timetable and Conflict Detection System that supports dynamic class rescheduling in a higher education environment. The system combines CSP-based timetable generation with localised conflict-aware rescheduling, grouped course coordination, override-aware state management, and automated student notifications.

The results demonstrate that the system improves timetable responsiveness and reliability when compared with manual scheduling workflows. Scenario-based evaluation showed that timetable changes which previously required prolonged coordination could be completed within seconds using the system. Performance results showed near-instant preview generation and practical full rescheduling completion times. Functional tests confirmed correct behaviour across timetable generation, conflict detection, rescheduling validation, and dynamic slot reuse. Limited usability observations indicated that the system's core workflows were understandable to users.

The principal contribution of the study is the demonstration that academic timetable systems should move beyond static schedule construction toward dynamic timetable adaptation. In real institutional contexts, the ability to safely modify a timetable after publication is as important as generating the timetable itself.

For practical deployment, the system should be extended to include SMS alerts in addition to email and in-application notifications, particularly where students rely heavily on mobile communication. To further improve the underlying engine's performance on large-scale institutional datasets, future iterations may incorporate advanced local search metaheuristics or simulated annealing techniques (Selimi, et al., 2022). It should also be integrated with existing institutional student information systems to reduce manual data entry and improve data accuracy. Before full deployment, broader testing should involve active lecturers, administrators, and a larger student population. The system should also be extended to cover all programmes within the School of Business before wider institutional rollout.

Future research may focus on scalability optimisation, hybrid scheduling approaches, behavioural learning from repeated rescheduling patterns, soft preference modelling, venue utilisation analytics, conflict trend reporting, and timetable stability analysis. These extensions could evolve the prototype into a more adaptive and intelligent academic scheduling platform.

Ethical Considerations

The prototype was developed and evaluated in a local environment using controlled records rather than live institutional production data. Documented communication records used during requirements analysis and scenario evaluation were operational scheduling communications available to the researcher through involvement in timetable coordination. Any future institutional deployment should include appropriate ethical and data protection safeguards, including secure authentication, role-based access control, transparent data handling, and collection of only scheduling data necessary for system operation.

Data Availability Statement

The prototype evaluation data described in this manuscript were generated from controlled prototype testing and documented operational scheduling scenarios. Access to raw communication records is restricted because they may contain personal or institutional communication details.

Funding Statement

No external funding was reported for this study.

Conflict of Interest Statement

The authors should declare whether any competing interests exist before submission. For the present draft, no competing interests have been specified.

Author Contributions

Author contribution details should be confirmed before submission. A suggested structure is: Matimba Handabile: conceptualisation, software development, data collection, testing, and original project report preparation. Kayaza Ngoma: supervision, methodological guidance, manuscript review, and academic editing.

REFERENCES

1. Bashab, A. et al. (2023). Optimisation techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues. Computers, Materials & Continua, 74(3), 6461-6484.
2. Ceschia, S., Di Gaspero, L., & Schaerf, A. (2023). Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research, 308(1), 1-18.
3. Chen, M. C. et al. (2021). A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access, 9, 106515-106529.
4. Davison, M., Kheiri, A., & Zografos, K. (2025). Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling, 28, 195-215.
5. Ďuriš, V. (2020). Algorithmic verification of constraint satisfaction method on timetable problem. Mathematics and Statistics, 8(6), 728-739.
6. Müller, T. (2024). Student scheduling at Purdue University. Purdue University.
7. Müller, T., Rudová, H., & Müllerová, Z. (2025). Real-world university course timetabling at the International Timetabling Competition 2019. Journal of Scheduling, 28, 247-267.
8. Premananda, G. A., & Muklason, A. (2021). Complex university timetabling using iterative forward search algorithm and Great Deluge algorithm. Khazanah Informatika, 7(1), 39-46.
9. Qefalija, E., Snopce, H., & Dermaku, A. (2024). Literature review on constraint satisfaction problems solving. International SymposiuPAPERm on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkiye.
10. Abubakar, S., Dauda, M. K., Musa, M. A. & Muhammad, B. M., 2026. Design and Implementation of a Web-Based Lecture Timetable Scheduling System. Journal of Informatics and Web Engineering, 5(1), pp. 252-266.
11. Ceschia, S., Di Gaspero, L. & Schaerf, A., 2022. Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research, pp. 23-54.
12. Diallo, F. P. & Tudose, C., 2024. Optimizing the Scheduling of Teaching Activities in a Faculty. Applied Sciences, Volume 14, pp. 1-27.
13. Hevner, A. R., March, S. T. & Park, J., 2004. Design Science in Information Systems Research. MIS Quarterly, 28(1), pp. 75-105.
14. Nielsen, J. & Molich, R., 1990. Heuristic Evaluation of User Interfaces. s.l., Conference on Human Factors in Computing.
15. Peffers, K., Tuunanen, T. & Rothenberger, M. A., 2007. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), pp. 45-77.
16. Russell, S. & Norvig, P., 2021. Artificial Intelligence A Modern Approach. 4th ed. s.l.:Pearson .
17. Selimi, S., Arbneshi, L., Sylejmani, K. & Musliu, N., 2022. Iterated Local Search for the examination timetabling problem with constructive-based initial solution. s.l., International Conference on the Practice and Theory of Automated Timetabling.
18. Vrielink, R. A. O., Jansen, E. A., Hans, E. W. & Hillegersberg, J. v., 2019. Practices in timetabling in higher education institutions: a systematic review. Ann Oper Res, pp. 145-160.
19. Royce, W.W., 1970. Managing the development of large software systems. In Proceedings of IEEE WESCON (Vol. 26, pp. 1-9).