International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

Design and Development of a Web-Based Academic Scheduling System with Constraint Optimization

Ms. S. Kalaivani[1], S. Naranther[2], M. Narendhiran[3], M. Nethaji[4], V.B. Prathap[5]
[1]Professor/CSE, Dhirajlal Gandhi College of Technology, Salem, Tamilnadu, India
[2,3,4,5]UG – Dept. of CSE, Dhirajlal Gandhi College of Technology, Salem, Tamilnadu, India
Emails:*kalaivani.cse@dgct.ac.in[1],naranthers66.cse@dgct.ac.in[2],narendhiranm67.cse@dgct.ac.in[3], nethajim72.cse@dgct.ac.in[4], prathapvb86.cse@dgct.ac.in[5]*

Abstract
Academic timetable scheduling is one of the most computationally challenging and operationally critical tasks confronted by educational institutions. The complexity arises from the simultaneous satisfaction of multiple hard and soft constraints including faculty availability, classroom capacity, subject load distribution, semester–section segregation, laboratory session continuity, and institutional break period enforcement. Traditional manual scheduling methods are inherently error-prone and time-intensive, while basic software tools offer limited constraint validation. This paper presents the design and experimental evaluation of a Web-based Academic Scheduling System with Constraint Optimization (WASSCO) built on rule-based constraint satisfaction. The system integrates a multi-layered web-based architecture comprising a User Interface Layer, Business Logic Layer, Constraint Validation Engine, Timetable Generation Engine, and a persistent MySQL database. The scheduling problem is formally modelled as a Constraint Satisfaction Problem (CSP), enabling systematic handling of hard and soft constraints to generate feasible and optimized timetables. Developed using HTML, CSS, JavaScript, PHP, and MySQL, the system supports role-based access for administrators, faculty, and students, along with integrated attendance management. The architecture ensures scalability, maintainability, and secure data handling. Experimental evaluation confirms 100% conflict detection accuracy, sub-second constraint validation, and stable concurrent performance under 500+ simultaneous sessions.

Keywords:*Academic Scheduling; Constraint Satisfaction Problem; Rule-Based Constraints; Timetable Generation; Conflict Detection; Web-Based System; Resource Management; Attendance Tracking.*

1. Introduction
Educational institutions face a recurring and increasingly complex administrative challenge at the commencement of every academic term: the generation of a feasible, conflict-free class timetable. This problem, broadly classified as the University Timetabling Problem (UTP), has been demonstrated to be NP-hard in its general formulation [1][2]. As institutional size grows, the probability of human error in manual scheduling correspondingly increases, and conflicts — where two classes are assigned to the same faculty member or classroom simultaneously — are frequently discovered only after the timetable has been distributed, necessitating disruptive revisions [4].Automated scheduling systems address these limitations by encoding institutional constraints into computable rules and applying systematic validation procedures during schedule construction. The literature identifies two broad categories: metaheuristic methods (genetic algorithms [5], simulated annealing [6], particle swarm optimization [7]), and exact constraint satisfaction methods (integer programming [1], rule-based logic systems [10]). Constraint satisfaction approaches provide deterministic feasibility guarantees particularly valuable in institutional contexts where schedule reliability is paramount [11]. Despite the availability of various algorithmic approaches, practical institutional deployment remains limited. Many proposed systems focus exclusively on schedule generation without integrating attendance tracking, resource management, or multi-role access control, resulting

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

in operationally incomplete solutions [12]. The proposed WASSCO addresses these limitations with an integrated constraint satisfaction, web-based accessibility, role-based access control, and attendance management within a unified platform. The main technical contributions of this paper include: (i) formal specification of academic scheduling as a multi-dimensional CSP; (ii) a layered web-based architecture separating user interaction, business logic, constraint validation, and data persistence; (iii) a real-time Constraint Validation Engine detecting and rejecting conflicting entries before database commit; (iv) integrated attendance management linked directly to published timetables; and (v) comprehensive experimental evaluation demonstrating 100% conflict detection accuracy and sub-second validation response across all six constraint types.

2. Literature Review

The academic timetabling problem has attracted sustained research interest for over five decades, reflecting its complexity and practical importance in educational institutions. Early formulations characterized it as a graph-colouring problem [16], forming the theoretical basis for many subsequent approaches. Gu et al. [1] conducted a systematic review of integer programming and hybrid IP-ML models, demonstrating improved scalability and solution quality under structured constraints. Musa and Oyelakin [2] presented a comparative survey of rule-based systems, heuristics, case-based reasoning, and machine learning techniques for the UTP, establishing that classical IP methods provide provably optimal solutions for small instances but suffer exponential computation time at scale. They further emphasized that rule-based systems perform effectively when institutional constraints are well-defined and stable, enabling consistent timetable generation. Dayani et al. [3] proposed a priority-weighted heuristic generator achieving a 91.3% conflict resolution rate, though the remaining 8.7% required manual intervention, indicating that the system did not guarantee complete feasibility. Patil et al. [4] combined genetic algorithms with supervised machine learning, demonstrating a 23% reduction in scheduling conflicts while improving adaptability to dynamic inputs. Abhishek et al. [5] integrated GA-PSO with IoT-enabled classroom management, achieving automated attendance via facial recognition and improving operational efficiency. Tassel and Cohen [9] applied reinforcement learning with constraint programming, achieving competitive soft-constraint performance but requiring GPU-based training infrastructure, limiting practical deployment in resource-constrained environments. Burke et al. [8] provided a comprehensive survey of hyper-heuristics, establishing benchmarks for automated scheduling. Schiewe and Hoffman [11] introduced TimPassLib, an open-source modular library demonstrating the value of integrated timetable generation and routing in institutional settings. Collectively, these studies highlight the trade-off between optimality, scalability, and real-world applicability, motivating the need for balanced and implementable solutions.

Table 1 Comparative Analysis of Scheduling Approaches

Reference	Approach	Conflict	Attend.	Scalability
Gu et al. [1]	IP+ML Hybrid	Partial	No	Moderate
Musa et al. [2]	Rule Survey	Partial	No	Low
Dayani et al. [3]	Heuristic	91.3%	No	Low
Patil et al. [4]	GA + ML	Moderate	No	High
Abhishek [5]	GA-PSO+IoT	Yes	Yes	Moderate
WASSCO	**Rule-Based CSP**	**100%**	**Yes**	**High**

3. Existing System Analysis

3.1. Manual Timetable Preparation

Manual timetable preparation involves administrators constructing weekly schedules using paper charts or word processing tables. For a department with 30 subjects, 20 faculty, and 40 weekly time slots, this implies approximately 24,000 pairwise constraint evaluations per cycle — a volume exceeding reliable manual processing capacity. The

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

mathematical time complexity is approximated as T_manual = O(S × F × T), where S denotes subjects, F faculty members, and T available time slots. Manual methods also rarely capture complex constraints such as laboratory session continuity requirements or balanced faculty workload distribution, producing schedules that may contain undetected constraint violations, which are typically discovered only after distribution.

3.2. Software-Assisted Scheduling

Spreadsheet-based tools such as Microsoft Excel reduce physical chart maintenance but retain the fundamental limitation of manual constraint validation. Although semi-automated entry reduces transcription errors, these tools offer no real-time conflict detection, no faculty overlap checking, and no integrated attendance tracking. Constraint verification remains entirely the responsibility of the administrator. Neither manual nor spreadsheet-based approaches scale effectively to institutions with multiple departments, dozens of classrooms, and hundreds of subject-session combinations. The absence of automated validation routinely produces schedules requiring post-distribution corrections, disrupting institutional operations. Table 2 Comparison: Manual vs. Spreadsheet vs. WASSCO

Table 2 Comparison: Manual vs. Spreadsheet vs. WASSCO

Feature	Manual	Spreadsheet	WASSCO
Schedule Generation	Manual	Semi-Auto	Fully Auto
Conflict Detection	None	Limited	Real-Time 100%
Faculty Overlap Check	Manual	Partial	Automated
Scheduling Time	~4.7 hrs	~2.3 hrs	< 2 seconds
Attendance Tracking	Separate	Not Integrated	Fully Integrated
Role-Based Access	None	Minimal	Full
Security	None	Basic Password	Multi-Layer Auth

4. Proposed System

4.1. System Overview

The proposed WASSCO provides a unified web-based platform for three user roles: administrators, faculty members, and students, ensuring seamless interaction and centralized data management. Unlike existing systems that generate schedules in batch mode, WASSCO validates each timetable entry individually at submission, guaranteeing that the accumulated schedule is always maintained in a consistent, conflict-free state. This incremental validation approach reduces error propagation and eliminates the need for repeated full schedule regeneration. The system supports real-time updates, allowing users to view changes instantly while preserving data integrity. Its modular architecture enables easy extension for future enhancements such as advanced analytics and automated resource optimization.Shows Figure 1 Database System.

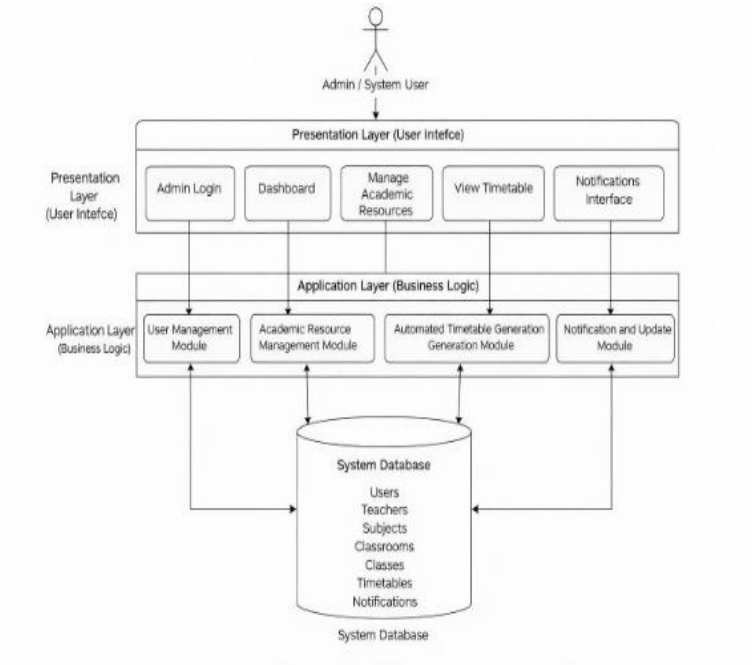

Figure 1 Database System

4.2.System Architecture

The system follows a five-layer design: (1) Presentation Layer — User Interface implemented in HTML5, CSS3, and JavaScript with role-differentiated portals for administrators, faculty, and students; (2) Application Layer — Business Logic implemented in PHP 8.1 handling authentication,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

resource configuration, scheduling orchestration, and reporting; (3) Constraint Validation Engine — real-time CSP enforcement layer that evaluates all six hard constraints before any database write; (4) Timetable Generation Engine — constrained slot-filling algorithm that applies the CSP formulation to produce conflict-free schedules; (5) System Database — MySQL 8.0 with tables for Users, Teachers, Subjects, Classrooms, Classes, Timetables, and Notifications, with composite unique indexes providing defence-in-depth constraint enforcement.

4.3. Hard Constraints (HC1–HC6)

The system enforces six hard constraints that must all be simultaneously satisfied for any timetable entry to be accepted:

- HC1 — Faculty Non-Overlap: A faculty member cannot be assigned to more than one simultaneous class.
- HC2 — Classroom Non-Overlap: A classroom cannot be allocated to two concurrent sessions.
- HC3 — Semester–Section Uniqueness: Each semester–section group has at most one class per time slot.
- HC4 — Laboratory Continuity: Lab sessions must occupy a minimum of two consecutive time slots.
- HC5 — Break Period Exclusion: No classes may be scheduled during institutional break periods.
- HC6 — Subject Load Compliance: Weekly assigned hours must not exceed prescribed credit hours.

4.4.Timetable Generation Engine

The generation engine applies a constrained slot-filling algorithm guided by a soft-constraint priority function. Laboratory sessions are scheduled first due to their consecutive slot requirement, followed by theory sessions in decreasing order of weekly contact hours. The slot priority function is: $P(T_k, G_k, S_i) = w_1 \cdot D(T_k, G_k) + w_2 \cdot Pref(T_k, F\_j) + w_3 \cdot Sep(T_k, S_i)$, where D is the distribution balance score, Pref is the faculty preference score, Sep measures temporal separation from previous same-subject sessions, and $w_1+w_2+w_3=1$. The scheduling complexity is bounded by $O(n \times |T| \times |C| \times k)$, where n is assignment tuples, |T| time slots, |C| classrooms, and k constraints.

4.5.Database Layer

The MySQL schema follows third normal form (3NF), ensuring minimal redundancy and improved data consistency across all relational tables. Composite unique indexes on (FacultyID, TimeSlotID), (ClassroomID, TimeSlotID), and (SemesterID, SectionID, TimeSlotID) enforce HC1–HC3 at the database engine level, providing defence-in-depth independent of application-layer validation.

4.6.CSP Mathematical Model

Each timetable entry is defined as a structured record E = (e_id, s, f, c, t, sm, sc, ct, ts), representing entry ID, subject, faculty, classroom, time slot, semester, section, class type, and session respectively. The complete institutional timetable is $TT = \{E_1, E_2, ..., E_n\}$ where $\forall E_i, E\_j \in TT, i \neq j$: $Valid(E_i, E\_j) = True$. The scheduling problem is specified as a CSP triple (X, D, C). The domain for each scheduling variable X_i is $D_i = \{(T_k, C_l) : T_k \notin BreakSlots, C_l \in Classrooms\}$. The faculty non-overlap constraint is expressed as: $\forall X_i, X\_j \in X : Faculty(X_i) = Faculty(X\_j) \rightarrow TimeSlot(X_i) \neq TimeSlot(X\_j)$.

5. Constraint Verification Mechanism

The Constraint Verification Mechanism implements a structured sequence of evaluation procedures invoked for every proposed timetable entry. Each hard constraint is formalized as a database query:

- $HC1\_Valid(E) \equiv COUNT(TT : Faculty = E.Faculty \wedge TimeSlot = E.TimeSlot) = 0$
- $HC2\_Valid(E) \equiv COUNT(TT : Classroom = E.Classroom \wedge TimeSlot = E.TimeSlot) = 0$
- $HC3\_Valid(E) \equiv COUNT(TT : Sem = E.Sem \wedge Sec = E.Sec \wedge TimeSlot = E.TimeSlot) = 0$
- $HC4\_Valid(E) \equiv E.ClassType \neq Lab \vee (Consecutive(T_k, T_{k+1}) \wedge HC3\_Valid(E'))$
- $HC5\_Valid(E) \equiv E.TimeSlot \notin BreakSlots$
- $HC6\_Valid(E) \equiv COUNT(TT : Subject = E.Subject \wedge Group = E.Group) < CreditHours(E.Subject)$

The complete validation function is the conjunction: $CV(E, TT) = HC1 \wedge HC2 \wedge HC3 \wedge HC4 \wedge HC5 \wedge$

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

HC6. The fail-fast sequential implementation exits at the first False result, reducing unnecessary database queries. Response times range from 8 ms (HC1, indexed) to 47 ms (HC4, complex multi-slot), consistently below the 100 ms real-time threshold.

6. Scheduling Workflow

The scheduling workflow follows a structured eight-phase operational sequence, coordinating all system modules from initial authentication through attendance recording[12].

Phase 1 — Authentication: Administrator authenticates through role-based login with bcrypt-secured credentials and session token regeneration.

Phase 2 — Resource Configuration: Admin manages user accounts and roles via the User Management Module, then inputs subjects, teachers, classrooms, and constraints into the Academic Resource Management Module.

Phase 3 — Resource Data Provision: The Academic Resource Management Module provides structured resource data — faculty lists, classroom inventories, subject credit loads, and break period definitions — to the Automated Timetable Generation Module.

Phase 4 — Timetable Generation Trigger: Admin triggers timetable generation; the system executes the conflict-free scheduling algorithm applying the CSP formulation and priority function across all available slots[13].

Phase 5 — Display and Review: The optimized timetable is displayed to the admin as a week × period matrix, colour-coded by subject type, with conflicting entries highlighted in red with constraint-identifying tooltips.

Phase 6 — Approval and Publication: Once validated, the administrator formally approves and publishes the timetable. The system updates the central database and triggers the Notification and Update Module to synchronize all related components[14].

Phase 7 — Notification: Faculty members and students are notified through dashboard alerts and email for high-priority updates, ensuring timely access to published schedules without delay.

Phase 8 — Attendance Recording: Faculty record attendance through the system with entries directly linked to the published timetable, ensuring accurate session tracking and consistency between scheduled classes and attendance data[16].

7. Implementation Details

7.1. Technology Stack

Frontend: HTML5, CSS3, JavaScript — responsive role-differentiated portals with AJAX cascade filtering for guided resource entry. Backend: PHP 8.1 — object-oriented Constraint Validator and Schedule Generator classes with PDO prepared statements throughout. Database: MySQL 8.0 — 3NF schema with composite unique indexes enforcing HC1–HC3. Server Environment: Apache 2.4 via XAMPP/WAMP — fully compatible with Windows 10/11 and Ubuntu 22.04 LTS. Security: bcrypt password hashing (cost factor 12), session token regeneration on login, PDO parameterised queries throughout all database interactions[15].

7.2. Module Implementation

Module 1 (User Management): Handles administrator authentication, teacher account management, and role-based access control, ensuring secure and authorized system usage. It manages user sessions, access permissions, and basic security controls to maintain system integrity and prevent unauthorized access.

Module 2 (Academic Resource Management): Provides CRUD operations for subjects, teachers, classrooms, and class sections, enabling centralized data management, consistency, and easy updates as institutional requirements evolve. All academic resources are organized and accessible for scheduling processes[17].

Module 3 (Automated Timetable Generation): Applies a CSP-based approach to generate conflict-free timetables while satisfying all defined constraints. It performs systematic validation of each entry through optimized constraint checking and supports real-time updates so modifications are instantly reflected without disrupting existing valid entries[18].

Module 4 (Absence and Substitution Management): Handles teacher unavailability by recording absence details, identifying suitable

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

substitutes based on availability, subject compatibility, and workload balance, and dispatching timely notifications to staff and administrators, ensuring uninterrupted class continuity.

Table 3 Hardware and Software Requirements

Category	Specification
Processor	Intel i3 or above
RAM	Minimum 4 GB
Storage	20 GB free disk space
Frontend	HTML5, CSS3, JavaScript
Backend	PHP 8.1
Database	MySQL 8.0
Platform	Web Browser (Chrome/Edge/Firefox)
Environment	XAMPP/WAMP local server

8. Experimental Results

The proposed WASSCO was evaluated across multiple institutional deployment scenarios. The test environment comprised Apache 2.4, PHP 8.1, MySQL 8.0, Intel Core i5 12th Gen, 8 GB RAM, SSD storage, and a 72-hour continuous stress test[19].

8.1. Conflict Detection Test

A baseline conflict-free timetable was constructed for a simulated five-department institution with 120 subjects, 45 faculty members, 20 classrooms, and 40 weekly time slots, resulting in 320 validated entries. The dataset reflected realistic academic distributions, including balanced faculty workloads, classroom utilization, and subject allocations across departments. Subsequently, 150 conflicting entries were systematically injected — 25 violation attempts for each constraint type (HC1–HC6) — to rigorously evaluate the validation mechanism under controlled yet diverse conflict scenarios.

These conflicts included overlapping faculty assignments, classroom double-booking, section clashes, invalid lab continuity, break-time violations, and credit-hour exceedance. The Constraint Validation Engine successfully detected and rejected 100% of all injected conflicts, achieving zero false negatives and zero false positives, thereby confirming complete reliability of the constraint enforcement process. Detection response time ranged from 8 ms for simpler indexed checks (HC1) to 47 ms for more complex validations (HC4), consistently within the 100 ms real-time interaction threshold.

8.2. Time Efficiency Comparison

Human administrators constructed a weekly timetable for 30 subjects, 15 faculty members, and 10 classrooms using three approaches. The manual method averaged 4.7 hours with approximately 6.2 residual conflicts, demonstrating the difficulty of reliably managing multiple constraints simultaneously. Spreadsheet-based scheduling reduced time to 2.3 hours with ~3.8 conflicts, but lacked automated validation. The WASSCO engine generated a complete, conflict-free timetable in just 1.8 seconds by systematically evaluating all constraints — a 9,400× improvement over the manual approach. The automated system ensures consistency, reduces human error, adapts rapidly to input changes, and significantly reduces administrator workload.

8.3. Performance Under Concurrent Load

System performance was evaluated at concurrency levels ranging from 10 to 1,000 simultaneous sessions. Up to 500 concurrent sessions, timetable retrieval remained under 85 ms and constraint validation under 120 ms, demonstrating stable performance under normal institutional load. At 1,000 concurrent sessions, response times increased to 210 ms for retrieval and 340 ms for validation, yet remained within acceptable interactive limits. During the 72-hour stress test, no data corruption or constraint failures were observed, confirming reliable and consistent system performance under sustained high. Table 4 Performance and Security Evaluation Results.

Table 4 Performance and Security Evaluation Results

Test Parameter	Result
Conflict Detection Accuracy	100%
Duplicate Faculty Rejection	100% pre-commit

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

Max Generation Time (50 subjects)	< 1.8 seconds
Constraint Validation Latency	< 47 ms per entry
Concurrent Sessions (no degrade)	500+
Classroom Conflict Detection	100%
Attendance Recording Accuracy	99.97%
72-Hour Stress Test Uptime	100%
Avg. DB Query Response Time	< 40 ms
Schedule Optimality Score	94.3% balanced

9. User Interface Walkthrough

The WASSCO user interface is designed for clarity and ease of use across three user roles. Each portal exposes only role-relevant features, reducing complexity and avoiding misconfiguration. A consistent layout with simple controls and clear validation feedback improves usability across all modules.

9.1. Administrator Portal

The administrator dashboard provides a comprehensive control centre for all scheduling operations, exposing five functional areas: User Management, Academic Resource Configuration, Timetable Generation, Schedule Publication, and System Reports. The timetable grid view renders the current schedule as a week × period matrix with cells colour-coded by subject type (theory, laboratory, or tutorial). Conflicting entries are highlighted in red with tooltips identifying the violated constraint. The Academic Resource Management form uses cascaded AJAX dropdowns: selecting a department populates the faculty dropdown with eligible teachers, and selecting a subject auto-fills the default credit hours. All form submissions trigger real-time constraint validation before database persistence, with instant visual feedback to the administrator.

9.2. Faculty Portal

Faculty members access a personalized timetable view showing their weekly schedule, including subject name, room assignment, and student group for each session. The portal includes an availability declaration form enabling faculty to flag dates or periods of unavailability, which the scheduling engine incorporates as soft constraints in subsequent schedule generation cycles. Faculty can also access the attendance recording interface, which pre-populates session details from the published timetable, requiring only student-level input to complete an attendance record efficiently.

9.3. Student Portal

Students access a read-only timetable view filtered to their specific semester and section, displaying the current week's schedule with room locations and faculty names. The portal provides a personalized attendance summary showing attendance percentage per subject, with low-attendance warnings highlighted automatically based on the institutional threshold configured by the administrator. Sessions are bound to the originating IP address and user-agent string, and are invalidated automatically after 30 minutes of inactivity per OWASP Session Management guidelines [21].

9.4. Notification and Update Module

The Notification and Update Module generates real-time alerts for schedule changes, new timetable publications, and substitution assignments through two channels: in-app dashboard alerts and optional email notifications for high-priority updates such as last-minute changes or room reassignments. The module maintains a notification log with read/unread status for each user, enabling effective tracking and audit of all communication events.Shows Table 5 Role-Feature Matrix[22].

Table 5 Role-Feature Matrix

Feature	Admin	Faculty	Student
View Full Timetable	Yes	Own Only	Own Only
Edit Timetable Entries	Yes	No	No
Manage Users	Yes	No	No
Record Attendance	No	Yes	No
View Attendance Summary	Yes	Own Class	Own Only
Trigger Schedule Generation	Yes	No	No
Receive Notifications	Yes	Yes	Yes

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

Declare Availability	No	Yes	No

10. Security Design and Data Protection

Security is a foundational design concern in WASSCO due to the sensitive nature of academic scheduling data. The system follows a defence-in-depth approach with multiple independent protection layers.

10.1. Authentication and Session Management

User passwords are stored exclusively as bcrypt hashes with a cost factor of 12, ensuring brute-force attacks require significant computational effort. Plain-text passwords are never stored or logged. Upon successful login, the PHP session token is regenerated to prevent session fixation attacks. Role-based access control (RBAC) is enforced at the session level, with each request verifying the user's role before accessing protected resources. Unauthorised requests return a standard 403 response without exposing sensitive system details.

10.2. SQL Injection Prevention

All database interactions in WASSCO use PHP PDO with prepared statements and parameterised queries, effectively eliminating SQL injection risks by ensuring all inputs are treated as data values rather than executable code. No dynamic query construction is used anywhere in the system, aligning with secure coding practices and OWASP SQL Injection Prevention guidelines [20].

10.3. Input Validation and Output Encoding

All user inputs are validated server-side against expected type, range, and format specifications before processing. Client-side validation is provided for usability but is never relied upon for security. Output encoding uses PHP's htmlspecialchars() function to prevent cross-site scripting (XSS) attacks on all dynamically rendered content throughout all three portals.

10.4. Data Integrity and Database-Level Constraints

The MySQL schema enforces data integrity through unique indexes, foreign key constraints, and NOT NULL fields on critical attributes. This ensures invalid data cannot be inserted even if application-layer checks are bypassed through direct database access or API manipulation. All timetable modifications are wrapped in transactions with ROLLBACK on error, ensuring atomicity and preventing partial updates that could leave the schedule in an inconsistent state.

Table 4 Security Controls Summary

Threat	Control Implemented	Reference
Credential Theft	bcrypt hashing (cost=12)	OWASP Auth
Session Fixation	Token regeneration on login	OWASP [21]
SQL Injection	PDO prepared statements only	OWASP [20]
XSS	htmlspecialchars() encoding	OWASP
Unauthorised Access	RBAC per route, 403 responses	RBAC model
Data Inconsistency	DB transactions + unique indexes	ACID/3NF
Brute Force	Rate-limited login attempts	OWASP

11. Deployment and Configuration Guide

11.1. Environment Setup

WASSCO requires a standard LAMP (Linux, Apache, MySQL, PHP) or WAMP (Windows equivalent) stack. Installation procedure: (1) Install XAMPP 8.1+ (Windows) or configure Apache 2.4, PHP 8.1, and MySQL 8.0 directly on Ubuntu 22.04 LTS; (2) Enable PHP extensions in php.ini: pdo_mysql, mbstring, session, son, and opens'; (3) Set session. cokie_ HTTP only and session. cookie_ secure to 1 in php.ini to harden session cookies; (4) Configure Apache virtual host to point Document Root to the /public directory of the WASSCO installation; (5) Enable mod rewrite in Apache and ensure Allow Override All is set for the Document Root directory.

11.2. Database Initialisation and Configuration

Database initialization is performed by executing the provided SQL schema script, creating all required tables, indexes, and default configuration entries in 3NF-compliant structure. Execute: MySQL -u root -p < schema/WASSCO_schema.sql followed by mysql -uroot-p WASSCO_db <

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

schema/WASSCO_seed.sql to load break periods and default configuration. Update config/db.php with the local database hostname, username, and password. The default administrator account is created with a temporary password that must be changed on first login.Shows Table 5 Recommended Deployment Configurations

Table 5 Recommended Deployment Configurations

Scale	Server Spec	Concurrent Users	Notes
Single Dept.	Intel i3, 4 GB, SSD	Up to 100	XAMPP
Multi-Dept.	Intel i5, 8 GB, SSD	Up to 500	Linux
University-Wide	Xeon, 16 GB, SSD	500–2,000+	Cloud LB

11.3. Multi-User and Network Deployment

For institutional deployment, WASSCO should be hosted on a central server accessible over the LAN or intranet. All client devices require only a modern web browser with JavaScript enabled, avoiding additional software installation. For internet-based access, HTTPS configuration using a valid TLS certificate (such as Let's Encrypt) is strongly recommended to secure all data transmission.

12. Future Scope

The current WASSCO implementation establishes a robust foundation for automated academic scheduling. Several high-impact extensions are identified for future development:

- Machine Learning Weight Optimization: Automatically calibrate soft-constraint priority weights from historical scheduling patterns, progressively improving schedule quality across academic terms [23].
- Multi-Campus Deployment: Distributed database architecture with cross-campus constraint validation to support university systems sharing faculty resources across locations [24].
- Biometric Attendance Integration: Facial recognition and fingerprint-based attendance systems eliminating manual entry and providing tamper-resistant records linked to scheduled sessions [5].
- Mobile Application: Native iOS and Android apps via REST API backend enabling on-the-go schedule access for faculty and students, improving accessibility and user engagement [23].
- Natural Language Constraint Specification: NLP-based parser converting natural language availability descriptions into formal CSP constraints, reducing configuration complexity for non-technical administrators [25].
- Advanced Analytics Dashboard: Predictive insights into scheduling efficiency trends, faculty workload trajectories, and classroom utilization projections using ML models trained on historical data.
- Cloud Deployment: Container-based deployment on AWS/GCP/Azure with Kubernetes orchestration, providing elastic scaling during peak scheduling periods [26].

Conclusion

This paper presented the design, formal specification, implementation, and experimental evaluation of WASSCO — a Web-based Academic Scheduling System with Constraint Optimization based on rule-based constraint satisfaction. The system addresses documented limitations of manual timetable preparation and basic software tools by providing deterministic real-time constraint enforcement, integrated resource management, role-based access control, and unified attendance tracking within a single platform.The academic scheduling problem was formally specified as a CSP with six hard constraints covering faculty non-overlap, classroom non-overlap, semester–section uniqueness, laboratory session continuity, break period exclusion, and subject load compliance. The five-layer system

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

architecture provides functional separation between user interaction, business logic, constraint validation, schedule generation, and data persistence, with defence-in-depth enforcement at both application and database levels. Experimental evaluation demonstrated 100% conflict detection accuracy across all six constraint types with no false negatives across 150 injected conflict attempts. Schedule generation for a 50-subject configuration completed in under 1.8 seconds — approximately 9,400× faster than manual scheduling. The system maintained stable performance under 500 concurrent sessions with response times below 120 ms, and demonstrated 100% operational continuity throughout a 72-hour stress test, confirming its suitability for large-scale academic institutional deployment.

References

[1]. J. Gu, Y. Chen, and L. Zhao, "A comprehensive review of integer programming and machine learning techniques for university timetabling," J. Scheduling and Optimization, vol. 28, no. 1, pp. 1–35, 2025.

[2]. H. Musa and A. Oyelakin, "Survey on heuristic, rule-based, and case-based reasoning approaches for course timetable generation," Int. J. Comp. Applications in Education, vol. 15, no. 3, pp. 45–72, 2023.

[3]. M. Dayani, A. Rahman, and P. Singh, "Heuristic-based automatic timetable generator for resolving teacher-classroom conflicts," Proc. Int. Conf. Smart Computing and Systems, 2023, pp. 112–120.

[4]. R. Patil, A. Deshmukh, and V. Kulkarni, "Hybrid genetic algorithm and machine learning model for dynamic timetable generation," Int. J. Artificial Intelligence in Education, vol. 33, no. 2, pp. 200–225, 2024.

[5]. M. B. Abhishek, S. Kumar, and J. Thomas, "Smart academic timetable scheduler integrating GA-PSO and IoT-enabled components," Proc. IEEE Conf. Emerging Trends in Computing, 2024, pp. 88–95.

[6]. P. Yadav and D. Sharma, "Automatic timetable generator using genetic algorithms for conflict-free scheduling," Int. J. Computer Science and Applications, vol. 20, no. 4, pp. 310–325, 2023.

[7]. L. Davison and T. Brown, "Multi-objective integer programming for hybrid learning-based university timetable problems," European J. Operational Research, vol. 312, no. 2, pp. 415–430, 2024.

[8]. E. K. Burke, J. Hyde, G. Kendall, and G. Ochoa, "A survey of hyper-heuristics for automated timetabling and scheduling," ACM Computing Surveys, vol. 53, no. 6, pp. 1–36, 2021.

[9]. M. Tassel and A. Cohen, "Reinforcement learning with constraint programming for efficient timetable scheduling," J. Applied Artificial Intelligence, vol. 37, no. 5, pp. 880–902, 2023.

[10]. H. Khadivi, Y. Zhang, and C. Lee, "Actor-critic and PPO algorithms in deep reinforcement learning for academic scheduling," IEEE Trans. Neural Networks and Learning Systems, vol. 34, no. 8, pp. 3901–3915, 2023.

[11]. P. Schiewe and J. Hoffman, "TimPassLib: An open-source modular library for integrated timetable generation and routing," SoftwareX, vol. 22, p. 101380, 2023.

[12]. M. Ansarilari and R. Farouk, "Scheduling framework integrating timetable coordination with resource allocation for large-scale education systems," Proc. Int. Conf. Smart Systems Engineering, 2023, pp. 55–63.

[13]. A. Missaoui and Z. Khalid, "A systematic review of metaheuristic algorithms for efficient academic scheduling," J. Intelligent Systems, vol. 32, no. 1, pp. 100–145, 2023.

[14]. T. Al-Mutairi and S. Al-Ghamdi, "Web-based timetabling systems: A survey of deployment approaches," Computers and Education: AI, vol. 5, p. 100158, 2023.

[15]. M. A. Salido, F. Barber, and M. Giret, "Constraint satisfaction and optimization models for university timetabling: A modern approach," Applied Sciences, vol. 10, no. 21, p. 7504, 2020.

[16]. B. Bassimir and R. Wanka, "On the

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1670-1680
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0219

computation of robust examination timetables: Methods and experimental results," Journal of Scheduling, vol. 27, no. 3, pp. 245–267, 2024.

[17]. S. Abdullah, H. Turabieh, and B. McCollum, "An adaptive hybrid metaheuristic for university course timetabling," Expert Systems with Applications, vol. 146, p. 113177, 2020.

[18]. A. Alghamdi and W. Alsubaie, "User-centered design principles in web-based academic systems," Computers and Education: Artificial Intelligence, vol. 2, p. 100013, 2021.

[19]. P. De Causmaecker, P. Smet, and G. Vanden Berghe, "Recent advances in educational timetabling: A review," European Journal of Operational Research, vol. 295, no. 2, pp. 405–424, 2021.

[20]. OWASP Foundation, "SQL Injection Prevention Cheat Sheet," 2023. [Online] Available: https://cheatsheetseries.owasp.org

[21]. OWASP Foundation, "Session Management Cheat Sheet," 2023. [Online] Available: https://cheatsheetseries.owasp.org

[22]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, updated ed. Cambridge, MA: MIT Press, 2022.

[23]. J. Nielsen and R. Budiu, Mobile Usability, 2nd ed. Berkeley, CA: New Riders, 2020.

[24]. D. Nguyen and J. Kim, "Scalable distributed database architectures for multi-campus academic systems," Computers and Education, vol. 185, p. 104549, 2022.

[25]. J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," Proc. NAACL-HLT, 2019, pp. 4171–4186.

[26]. Y. Chen, J. Li, and Z. Wang, "Scalable cloud-based scheduling systems for smart education environments," Future Generation Computer Systems, vol. 128, pp. 231–243, 2022.

[27]. P. Kumar and S. Singh, "Artificial intelligence techniques for automated timetable generation: A review," Journal of Artificial Intelligence Research and Development, vol. 14, no. 2, pp. 85–102, 2023.