WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Design and Evaluation of an Integrated Faculty Management System with Multi-Format Timetable Parsing, Conflict Detection, and Period-Wise Attendance Monitoring

[AKILESHPRIYAN D][1], [ARAVIND S][2], [DANIEL ALEX A][3], [DANIEL WILLIAM BG][4], [DHARANITHARAN N][5], [Mrs.S.NANDHINI][6]

[COMPUTER SCIENCE AND ENGINEERING], [SRI SHAKTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY], [COIMBATORE], India
[aravindsri0314@gmail.com]

ABSTRACT—

Academic departments frequently manage teaching schedules, faculty assignments, classroom allocation, and student attendance through loosely coupled workflows. This paper describes the design of a web-based Smart Faculty Assistant Portal that connects timetable ingestion and validation with period-wise attendance monitoring. The prototype uses a PHP application layer and MySQL database, supports role-specific workflows, and accepts structured spreadsheet files and textual information extracted from digital PDF documents. Imported timetable entries are normalized against institutional reference records and checked for duplicate section slots, faculty double-booking, room clashes, and existing schedule assignments before confirmation. Faculty attendance is associated with a published timetable entry and supports present, absent, and on-duty statuses; HOD users can review records and summary reports. We specify a reproducible evaluation protocol covering parsing completeness, mapping accuracy, conflict-detection precision, data integrity, and administrative task completion time. Quantitative results and comparisons are reserved for controlled experiments, which are necessary before submission.

Index Terms—academic administration, attendance management, conflict detection, educational information systems, timetable parsing, web applications.

INTRODUCTION

Higher-education departments must coordinate courses, faculty members, classrooms, and student attendance across repeated teaching periods. Manual transfer of schedule data between spreadsheets, administrative records, and attendance registers can introduce duplicate assignments and inconsistent records. Digitization alone does not resolve these inconsistencies when scheduling and attendance modules remain disconnected.

The Smart Faculty Assistant Portal studied here is a role-based web application for academic administration. Its main design principle is to use the published timetable as the authoritative context for period-wise faculty attendance, while allowing departmental administrators to import and inspect schedules before publication. The implementation targets an ordinary institutional environment using PHP, MySQL, and an Apache-based local server.

The contribution examined in this paper is workflow integration rather than a new optimization algorithm or an artificial-intelligence model. Specifically, we document (1) multi-format timetable ingestion and reference-data normalization; (2) detection and presentation of conflicting schedule entries; and (3) linkage between published teaching slots and student-level attendance records. We then define an empirical evaluation that can test whether the integration reduces avoidable errors and administrative effort.

RELATED WORK

Course timetabling has been studied extensively as a constrained assignment problem. Bettinelli et al. [1] survey models, hard and soft constraints, and algorithmic approaches for curriculum-based timetabling. Hao and Benlic [2] develop bounds for benchmark curriculum-based timetabling instances. These studies emphasize generating or evaluating feasible schedules under explicit constraints, whereas the present prototype focuses on parsing schedules supplied by users, detecting operational conflicts, and linking accepted slots to downstream attendance.

Attendance-management literature explores alternative identification and recording technologies. Chen and Li [3] describe an RFID and facial-recognition attendance approach. Such work targets automated

identification; our system instead examines authenticated faculty entry, timetable-grounded class selection, and departmental monitoring. These approaches address related but different parts of the attendance workflow and should not be treated as directly comparable without common test conditions.

The research gap considered here is practical interoperability between human-prepared timetable files, validated departmental records, and period-specific attendance transactions. Demonstrating added value requires experimental comparison to a baseline process; the presence of several modules in a web portal is not by itself evidence of improvement.

SYSTEM DESIGN

A. *Roles and deployment*

The application exposes role-specific views for the head of department (HOD), faculty members, and students. The HOD workflow includes timetable management, monitoring, and attendance reports. Faculty members select an authorized date and period to load assigned classes and record attendance. Student-facing functions are maintained separately. The prototype is deployed with Apache, PHP, and MySQL; its functionality depends on the configured institutional reference tables.

B. *Reference data and core entities*

Timetable entries refer to a section, weekday, period, subject, faculty member, classroom, and publication status. Reference tables provide identifiers and labels for sections, periods, subjects, faculty, and classrooms. Attendance records associate a student and date with a timetable entry, enabling the selected class period to determine the valid student roster. The published state distinguishes active teaching entries from drafts.

FIG. 1. TO ADD: Three-layer architecture (HOD/Faculty/Student interfaces → PHP validation and workflow services → MySQL tables). Use a diagram derived from your actual source code.

TIMETABLE INGESTION AND VALIDATION

A. *Parsing and normalization*

The file-ingestion component handles comma-separated values (CSV) and Excel spreadsheets through a table-or-grid parser. Digital PDF text

is extracted using a PDF parsing library, then interpreted as structured day-period-subject-faculty-room lines. Image input is conditional on an available optical character recognition (OCR) executable. Consequently, support for scanned documents is environment-dependent and should be reported separately from digital-PDF support.

The parser maps weekday abbreviations to canonical day names, interprets period identifiers such as 2 and P2, and resolves subject, faculty, classroom, and section labels to MySQL identifiers. Each normalized row retains warnings when a label cannot be matched and errors when required schedule information is invalid. The validation interface should permit human correction before any irreversible import.

B. *Conflict detection*

The prototype checks duplicate section–day–period entries within an imported batch, as well as repeated faculty–day–period and classroom–day–period assignments. Database comparisons identify previously occupied section slots and cross-section faculty or room collisions. Existing entries can be presented under replacement or skip policies, while blocking conflicts are distinguished from nonblocking warnings.

Let a timetable row be $x = (s, d, p, f, r)$, where s is section, d is day, p is period, f is faculty, and r is room. A potential double-booking occurs when two different section rows share the same (d, p, f) or (d, p, r). This is a validation rule, not an automatic schedule optimizer. Experiments should test false positives as well as missed collisions, including repeated or malformed rows.

C. *Import boundary*

A preview step separates file parsing from database modification. For the evaluation, every import test should begin from an identical database snapshot so that collision counts are reproducible. The selected replacement policy, parser version, file type, and input size must be recorded for each run.

PERIOD-WISE ATTENDANCE WORKFLOW

The faculty view checks the authenticated faculty context and the selected date and period before loading a published timetable entry. The student list is derived from the section associated with that entry. Each student can be assigned PRESENT, ABSENT, or OD (on duty). The interface displays counts for these states and allows the faculty member to confirm the selected class before saving.

The attendance implementation uses database transactions to insert or update individual student records for a specified timetable and date. This design

aims to keep the records for one attendance-saving operation consistent if a database error occurs. Its transactional guarantees, concurrency behavior, duplicate prevention, and authorization boundaries still require dedicated tests; observing a working screen is not equivalent to validating these properties.

HOD users can review attendance by section, date, and period and examine aggregate weekly, monthly, or semester reports. The study should verify that HOD views are read-only where intended and that faculty users cannot modify attendance for classes assigned to another faculty member.

EXPERIMENTAL METHODOLOGY

A. *Research questions*

RQ1: How accurately does the parser extract and map timetable entries across CSV, Excel, and digital-PDF formats? RQ2: How reliably does validation identify timetable conflicts and malformed entries? RQ3: Does an integrated timetable-to-attendance workflow reduce task completion time and recording errors relative to a manual or spreadsheet-based baseline? RQ4: Are role and data-integrity constraints respected under normal and negative test cases?

B. *Test dataset and baselines*

Create a controlled dataset containing representative departments, sections, subjects, faculty, rooms, periods, and students. Prepare the same timetable scenarios in CSV, Excel, and digital PDF, including at least one clean batch, unknown labels, absent required fields, duplicate slots, faculty conflicts, room conflicts, and existing-record replacements. Document the number of rows, files, and intentional faults. Use a ground-truth answer key independently reviewed by at least two team members.

Compare import-supported data entry to manual entry of the same timetable, measuring elapsed time and number of correction steps. To study attendance, compare equivalent class-recording tasks with a paper or spreadsheet baseline. Counterbalance task order if participants perform both workflows. Record device specifications, PHP/MySQL versions, participant count, task instructions, and any institutional review or consent requirements.

C. *Evaluation metrics*

Parsing completeness = correctly extracted entries / ground-truth entries. Entity-mapping accuracy = correctly mapped fields / all ground-truth fields. Conflict-detection precision = true-positive conflict alerts / all conflict alerts; recall = true-positive alerts / all ground-truth conflicts. Attendance-record accuracy = correctly persisted student statuses / expected statuses. Report mean and median task time, variability across trials, and relevant confidence intervals.

D. *Functional and security checks*

Validate access restrictions for HOD, faculty, and student sessions; reject future-date attendance when disallowed; test empty classes, duplicate submissions, invalid status values, concurrent writes, and failed database transactions. Use anonymized or synthetic student data in evaluation files and screenshots. Document the limits of local XAMPP testing versus a deployed multi-user environment.

RESULTS AND DISCUSSION

Table I. Timetable Parser, Entity Mapping, and Conflict Detection Benchmark Results

The sample CSV and Excel datasets each contained 35 records and reported 13 faculty conflicts and 2 classroom conflicts, with processing times of 0.171489 seconds and 0.118339 seconds, respectively. The full-section CSV successfully parsed 54 records in 0.124186 seconds without conflicts. The digital PDF parser extracted 14 records in 0.052596 seconds. The five-section master CSV processed 270 records in 0.556682 seconds, with 28 unresolved faculty mappings and 20 unresolved classroom mappings. The invalid CSV test blocked all 3 rows, while the synthetic collision test detected duplicate section slots, faculty double-bookings, and classroom collisions in 0.010162 seconds. The real-time conflict detection API completed four checks in 0.006575 seconds. Finally, the database integrity audit examined 270 timetable records in 0.004471 seconds and reported zero faculty overlaps, classroom overlaps, duplicate section-period slots, and faculty-subject qualification mismatches.

A comparative evaluation was conducted to assess the performance of SmartFacultyPortal against the traditional manual timetable preparation method using 270 timetable slots across five academic sections. The automated portal processed the dataset in 0.231701 seconds, achieving a throughput of approximately 1,165 slots per second. In comparison, the manual preparation time was estimated at 270 minutes based on an assumed average of 60 seconds per slot. Based on these assumptions, the portal achieved an estimated 99.999% reduction in processing time. The automated system also performed faculty, classroom, and duplicate-slot conflict checks. The database integrity audit reported zero conflicts across the 270 published timetable records. However, the manual preparation time and error counts were model-based estimates rather than measurements from an actual human scheduler. Therefore, the

results demonstrate the portal's computational efficiency and database consistency, while a controlled manual trial would be required to establish a definitive empirical comparison.

LIMITATIONS AND THREATS TO VALIDITY

The parser expects recognizable spreadsheet headers or structured PDF text and is sensitive to arbitrary graphical timetable layouts. OCR-based image processing requires separate software configuration and testing. The current approach validates imported schedules rather than computing globally optimal schedules. Small datasets, artificial room allocations, or a single faculty account would limit external validity. Interface validation must not be treated as proof of access-control security.

Repeated experiments should use fixed database snapshots, independently verified ground truth, and representative workloads. Human-subject evaluations should follow the institution's ethics and consent requirements. Source files or supplementary materials must not expose real student records, database credentials, session cookies, or password-test artifacts.

CONCLUSION

This paper specifies and describes an integrated academic-administration prototype that connects multi-format timetable ingestion, conflict checks, published teaching schedules, and period-wise attendance recording. The design centers on traceability from an imported schedule entry to the faculty member, section, and student attendance record. The next research step is rigorous measurement of extraction and mapping accuracy, conflict-detection quality, task efficiency, and authorization/data-integrity behavior. Final claims and conclusions must be revised after these experiments are completed.

ACKNOWLEDGMENT / AI DISCLOSURE

We express our sincere gratitude to our project supervisor and the faculty members of the Department of Computer Science and Engineering for their valuable guidance, support, and encouragement throughout the development of SmartFacultyPortal. We also thank our institution for providing the necessary facilities and resources to complete this project. Finally, we thank our team members and everyone who supported us during the successful completion of this project.

Artificial intelligence tools were used to assist with coding, debugging, technical explanations, troubleshooting, and improving project documentation. The project team reviewed and implemented relevant suggestions and conducted testing within the project environment. AI tools were used as supportive resources and did not replace the team's responsibility for project development, result interpretation, and final documentation. All reported measurements and results should be understood according to the tests actually performed and verified by the team.

REFERENCES

A. Research Papers and Journals

The papers below can be found by searching their titles onGoogle Scholar.

- Schaerf, A. (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), 87–127. https://iolab.uniud.it/publication/scha-99/
- Abubakar, S., Dauda, M. K., Abdullahi Musa, M., & Muhammad,
- M. (2026). Design and implementation of a web-based lecture timetable scheduling system. Journal of Informatics and Web Engineering, 1(5), 252. https://shdl.mmu.edu.my/16288/
- Kadav, S., Patil, A., et al. (2022). Time table scheduling system. MIT University's Abhivruddhi Journal, 2(1), 14–16.
- https://abhivruddhi.mituniversity.ac.in/wp-content/uploads/2024/01/Vol-2-Issue-1-3.pdf
- Shamsul, N. U., & Affandi, A. (2026). Development and evaluation of a cross-subject attendance pattern system. Journal of Digital System Development, 4(1), 90–107. https://e-journal.uum.edu.my/index.php/jdsd/article/view/jdsd2026.4.1.7
- Yanuar, Y., & Waluyo, A. (2023). Designing a web-based student attendance information system using extreme programming. Jurnal E-Komtek, 7(2), 291–301. https://jurnal.politeknik-kebumen.ac.id/E-
- KOMTEK/article/download/1464/737
- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2), 38–
- 47.
- Smith, R. (2007). An overview of the Tesseract OCR engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition *(ICDAR)*, IEEE Computer Society, pp. 629–633. https://ieeexplore.ieee.org/document/4376991
- Shafait, F., & Smith, R. (2010). Table detection in heterogeneous documents. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems *(DAS '10)*, ACM, pp. 65–72. https://doi.org/10.1145/1815330.1815339

B. Technical Documentation and Standards

- The PHP Group. PHP Manual: password_hash(), password_verify() and mysqli prepared statements. https://www.php.net/manual
- PhpSpreadsheet Documentation. https://phpspreadsheet.readthedocs.io

- Oracle Corporation. MySQL Reference Manual. https://dev.mysql.com/doc
- Composer: Dependency Manager for PHP. https://getcomposer.org/doc
- Tesseract OCR. https://github.com/tesseract-ocr/tesseract
- Poppler PDF Rendering Library. https://poppler.freedesktop.org
- Apache Friends. XAMPP Documentation. https://www.apachefriends.org
- OWASP Foundation. Password Storage Cheat Sheet and Session Management Cheat Sheet.https://cheatsheetseries.owasp.org
- Chacon, S., & Straub, B. Pro Git. https://git-scm.com/book
- GitHub Docs. https://docs.github.com