

ScheduleEase – Smart Institutional Timetable Management System

Dr. Janani V¹, Kowsalya Devi M², Dineshkumar M² & Gobburu Prudhvi Raj²

¹Associate Professor and Head, Department of Information Science and Engineering
Sri Sairam College of Engineering, Bangalore

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

DOI: doi.org/10.34293/iejcsa.v4i4.168

Abstract - Class timetable scheduling is a critical administrative task in higher education, directly affecting academic flow, faculty workload distribution and infrastructure utilization. Spreadsheets and basic scheduling tools struggle with the growing complexity of modern curricula, especially the multidisciplinary pathways introduced by NEP 2020. This paper presents ScheduleEase, an intelligent web-based timetable management system that automates schedule creation for undergraduate and postgraduate programs. A rule-based scheduling engine assigns subjects, faculty and classrooms to weekly day/period slots while checking classroom availability, batch size, subject distribution, weekly teaching hours, faculty expertise and departmental constraints, and supports per-department, per-semester and per-section timetables. The MERN-stack platform provides JWT-based role-based access for admins and faculty, an interactive data-entry dashboard, conflict-free timetable generation, faculty-specific timetable views, a lecture-notes file-sharing module, attendance tracking, and a smart absence-cover workflow that suggests available substitute faculty. The design reduces manual effort, minimizes scheduling errors and improves resource utilization, giving institutions a scalable and future-ready scheduling solution.

Index Terms: Timetable scheduling, smart classroom, conflict-free scheduling, faculty absence cover, MERN stack, NEP 2020, academic resource allocation, conflict detection, higher education management.

INTRODUCTION

Higher education institutions manage increasingly complex academic structures: multiple departments, diverse course offerings, electives, laboratory sessions and large faculty pools. Every semester, an academic coordinator must convert this structure into a weekly timetable that assigns each subject to a faculty member, a classroom and a time slot without letting any two of these commitments collide. Timetables prepared manually in spreadsheets or physical registers are time-consuming, error-prone and unable to cope with the dynamic nature of modern academic environments [1].

The National Education Policy (NEP) 2020 [8] introduced multidisciplinary learning pathways and flexible course options, which further complicate scheduling. Students opting for open electives or interdisciplinary subjects raise the probability of clashes, classrooms stay under-utilized because availability is not tracked in real time, and faculty workload becomes uneven [2]. A timetable that was manageable under a fixed curriculum may

therefore become unworkable when students are free to combine subjects across departments.

A smart classroom and timetable scheduler addresses these challenges by combining digital classroom management with intelligent timetable automation. Room, laboratory and shared-infrastructure allocation is driven by availability and capacity, while classes are arranged around faculty availability, student batches, room constraints, subject requirements and fixed sessions [3]. Recent surveys of smart learning environments [6], [7] confirm that institutional digitization of this kind is becoming a baseline expectation rather than an optional upgrade.

Academic timetabling is also a well-studied computational problem. It is known to be computationally hard in its general form [9], and a large body of work has examined heuristic, constraint-based and optimization approaches to it [10]. For a working institution, however, an algorithm alone is not enough: the timetable must be entered, reviewed, published, queried by faculty, and repaired when a faculty member is absent. ScheduleEase therefore treats scheduling as one component of a complete web-based administrative platform.

This paper presents the design, methodology and implementation of ScheduleEase. Its first contribution is a constraint-aware, rule-based scheduling engine that assigns subjects, faculty and classrooms to weekly day/period slots, scoped per department, semester and section. Its second contribution is a conflict-detection mechanism covering five conflict types, namely faculty, classroom, student batch, subject hour and special slot, with backtracking and alternative suggestions. The third contribution is a MERN-stack implementation with JWT-based role-based access control for administrators and faculty, and the fourth is a set of supporting modules for lecture-note sharing, attendance tracking and a smart absence-cover workflow that proposes only conflict-free substitute faculty.

The remainder of the paper is organized as follows. The literature review surveys the theoretical and applied work on which the design rests, the research gap and problem statement then define what ScheduleEase sets out to solve, and the proposed system and its architecture are described next. The methodology and workflow section formulates the scheduling constraints and explains the algorithm, the implementation and evaluation section reports the realized platform together with its limitations and future directions, and the conclusion closes the paper.

LITERATURE REVIEW

The literature relevant to ScheduleEase can be organized into three groups defined by their primary concern: the theoretical foundations of timetabling as a computational problem, the application of smart-classroom technology and artificial intelligence to scheduling and resource allocation, and the design philosophy that separates a deployable administrative tool from a purely algorithmic contribution. Within each group, the

discussion prioritizes what the studies collectively reveal over a sequential description of each one.

A. Timetabling as a Computational Problem

University course timetabling asks for an assignment of events to time slots and rooms that satisfies a set of hard constraints, such as no resource being used twice at the same time, while respecting as many soft preferences as possible. Even, Itai and Shamir showed that the timetable problem is NP-complete in general [9], which explains why no inexpensive method guarantees an optimal timetable for every institution. Surveys of the field describe a range of approaches, including constraint-based methods and local search [10]. The practical lesson of this work is that a scheduling tool must be judged by whether it reliably finds feasible timetables and explains infeasible ones, since optimality cannot be promised in the general case.

B. Smart Classrooms and AI-Based Scheduling

More recent work has revisited the problem with integer programming and machine learning [4], and with artificial intelligence for classroom scheduling and resource optimization [5]. Studies of smart classrooms examine how sensors and AI are reshaping educational paradigms [1] and how smart classrooms influence the learning experience of students [2], [3], while surveys of smart learning environments in the post-pandemic era place institutional digitization firmly on the higher-education agenda [6], [7]. Read together, these studies establish a strong motivation for digital classroom management and show that the field is moving from fixed, hand-built schedules toward data-driven allocation of rooms, faculty and time.

What the same studies do not provide is a deployable administrative tool that a college can adopt directly. The solver-oriented papers [4], [5] concentrate on the quality of the assignment, and the smart-classroom papers [1]–[3] concentrate on the learning environment, so the surrounding workflow of data entry, review, publication, faculty access and leave handling is left to the institution. This observation shapes the position of ScheduleEase: it is conceived as a complete platform in which the scheduling engine is one module among several.

C. Rule-Based Design as a Deliberate Choice

ScheduleEase deliberately follows a rule-based, constraint-aware design. Every placement decision can be explained in terms of explicit rules (the room was free, the faculty member had no other class, the weekly hours were not exceeded), which matters for administrators who must justify a timetable to departments and faculty. The approach also keeps the engine lightweight enough to run inside a Node.js service, so that the timetable can be regenerated on demand whenever the underlying data changes. ScheduleEase differentiates itself by integrating timetable generation, role-based data management,

personal faculty views, lecture-note sharing, attendance and absence cover in a single platform built on widely used open-source technologies.

Taken together, the three groups of work point in the same direction. Theory tells us that a general timetabling problem cannot be solved optimally by inexpensive means [9], [10], applied studies show growing interest in AI-assisted scheduling and smart learning environments [1]–[7], and neither group delivers the complete administrative workflow that an academic office needs each semester. ScheduleEase is positioned in this space: it accepts that optimality cannot be guaranteed, concentrates on feasibility, transparency and repairability, and supplies the surrounding platform in which a feasible timetable becomes a published, queryable and maintainable institutional asset.

RESEARCH GAP

Three research gaps emerge from the reviewed literature. Each is stated below together with the reason it matters for a system intended for real institutional use.

Gap 1: Algorithms without a deployable administrative platform. The surveyed approaches to timetabling [4], [5], [10] describe how an assignment can be computed, but they rarely describe the complete cycle in which a timetable is entered, reviewed, published, queried by faculty and repaired. Justification: a college does not adopt a solver, it adopts a working tool, so a scheduling method has to be embedded in a platform with secure role-based access, data management and per-faculty views before it can replace spreadsheets.

Gap 2: Explainable placement and preventive conflict detection. A spreadsheet stores only the final result, so it can neither explain why a particular slot was chosen nor warn an administrator before a clash is introduced, and general-purpose optimizers give little insight into why a particular placement was made. Justification: administrators must defend a timetable to departments and faculty, so each placement should follow from explicit rules that are checked before the slot is confirmed, with the reason for any failure reported back.

Gap 3: Handling faculty absence without creating new conflicts. Existing manual practice offers no automated rescheduling when faculty take leave [4], and an ad hoc substitute can easily introduce a secondary clash. Justification: absences are a routine part of an academic term, so a practical system must identify substitutes who are genuinely free in the affected period and track the effect of the absence on syllabus completion.

ScheduleEase is designed to close these three gaps by combining rule-based generation, pre-confirmation conflict checking and a conflict-free absence-cover workflow within one platform.

PROBLEM STATEMENT

Manual timetable creation frequently results in class clashes when students choose open electives or interdisciplinary subjects, double-booked classrooms caused by the lack of

real-time room availability tracking, and uneven faculty workload in which some faculty are overloaded while others are under-utilized. It also produces poorly spaced timetables with back-to-back classes or needless gaps, and last-minute changes that are hard to update and communicate to the people affected.

Most colleges rely on Excel sheets or simple templates. Although easy to use, these tools cannot handle multiple simultaneous constraints such as room capacity, faculty preferences and subject-hour requirements. Key gaps include no automated generation, no real-time classroom status, no multi-department or multi-shift support, and no automated rescheduling when faculty take leave [4]. Because a spreadsheet stores only the final result, it also cannot explain why a particular slot was chosen or warn an administrator before a clash is introduced.

Formally, given a set of faculty, subjects, classrooms, student batches, weekly teaching hours and fixed events, the objective is to produce, for each department, semester and section, a weekly timetable in which no faculty member, classroom or student batch is scheduled twice in the same period, every room fits the batch and type of class placed in it, every subject receives its required weekly hours, and fixed events remain clear of regular classes. In addition, when a faculty member reports an absence, the system must identify substitutes who are free in the affected slot so that the cover itself does not create a new conflict.

The system that addresses this problem must satisfy a set of functional requirements. It must provide role-based secure login for administrators and faculty with protected routes, an input module for faculty data, subjects, classroom capacities, laboratory availability, batch strength, teaching hours and fixed event slots, and automatic timetable generation free of class clashes, faculty conflicts and room-assignment errors. It must detect conflicts and offer alternative scheduling suggestions when constraints cannot be met, present a grid view in which faculty see a personalized schedule and administrators can view any faculty timetable, reflect faculty leave, events or room maintenance in real time, and support lecture-note sharing of up to ten files per note, attendance sessions with summary reports and an absence-cover workflow.

The non-functional requirements are equally important. The platform must stay fast on large multi-department datasets, remain highly available during peak academic planning periods, and scale with departmental growth and NEP 2020 curriculum changes. Authentication, access control and data protection are mandatory, and detailed logs of timetable changes and approvals must be kept for traceability [5]. These requirements shape the architecture and security design presented in the later sections.

The formulation rests on a small number of explicit assumptions. Each timetable is generated for one scope, that is one department, semester and section, so that instances stay small and independent of one another; the working week is divided into a fixed number of days and periods; a laboratory session occupies consecutive periods and may be placed

only in a laboratory; and the data entered by the administrator, including faculty availability, subject hours and room capacities, is taken as accurate. Under these assumptions the engine can state precisely when a timetable is feasible and, when it is not, which constraint prevented completion.

PROPOSED SYSTEM

Guided by the findings of the review, the proposed system is ScheduleEase, a web-based timetable management platform for engineering colleges and universities. An administrator enters faculty profiles, subjects, classroom and laboratory information, student records and weekly teaching hours; the scheduling engine then generates a conflict-free weekly timetable for the chosen department, semester and section, which is displayed as a grid and exposed to faculty through a personalized My Timetable view. Faculty can additionally share lecture notes, mark attendance and report absences, and the administrator receives suggestions for substitute faculty who are free in the affected period.

Each design choice follows from the reviewed literature. A rule-based, constraint-aware engine was chosen because every placement can be explained by explicit rules and because it is light enough to run inside a Node.js service, which addresses the explainability gap identified above. The MERN stack was chosen because it is built on widely used open-source technologies and suits the flexible, document-shaped scheduling data. JWT-based role-based access was chosen to keep sensitive actions, such as generating timetables and managing users, restricted to authorized accounts, and the absence-cover module was added to address the leave-handling gap by proposing only conflict-free substitutes.

Timetable quality depends on how well the engine handles a set of optimization parameters. Faculty availability is respected through free and busy slots and leave periods are avoided. Classroom capacity and type are enforced so that the batch size fits the room and lab subjects go only to laboratories. Subject-hour requirements fix the theory and lab hours per week, a maximum number of classes per day prevents student and faculty overload, and faculty workload is balanced so that teaching hours are distributed equally. Special sessions such as guest lectures, NCC/NSS slots and department meetings are kept clear of regular classes, and for absence cover only faculty who are free in the slot are suggested.

The platform is organized into eight functional modules. Authentication and user management provides role-based login with JWT tokens, hashed passwords and e-mail password reset. Faculty management holds faculty details, subjects handled, availability and weekly workload, and classroom management records capacity, type and availability and prevents room clashes. Subject management creates subjects, assigns faculty, sets weekly hours and tracks syllabus completion, including bulk updates, while the timetable scheduling module is the admin-triggered engine for per-department, per-semester, per-section timetables with personalized faculty views. The notes module lets faculty upload

lecture notes, up to ten files per note, for students and staff to browse and download; the attendance module marks sessions, shows summaries and generates per-student reports; and the absence-cover module suggests substitutes, tracks cover status and reports the effect on syllabus completion.

A typical usage scenario illustrates how these modules work together. At the start of a semester the administrator enters or updates the master data and triggers generation for each department, semester and section in turn, reviews the resulting grids and publishes them. Faculty then open their own schedule, upload lecture notes after each topic and mark attendance for every class session. When a faculty member must be absent, the absence is reported through the platform, the system lists the substitutes who are free in that period, the administrator selects one, and the timetable and the syllabus-completion figures are updated without any manual search through spreadsheets.

SYSTEM ARCHITECTURE

ScheduleEase follows the three-tier model of Fig. 1, with external services for file storage and e-mail. The presentation layer is a React single-page application with sidebar navigation covering the Dashboard, Faculty, Students, Classrooms, Subjects, Timetable, My Timetable, Notes and Attendance pages. The application layer comprises Express routes and controllers for authentication, users, classrooms, subjects, timetables, notes, attendance and absence handling, guarded by JWT middleware and role checks, and it is the decision-making core of the system. The database layer holds Mongoose-modeled MongoDB collections hosted on MongoDB Atlas.

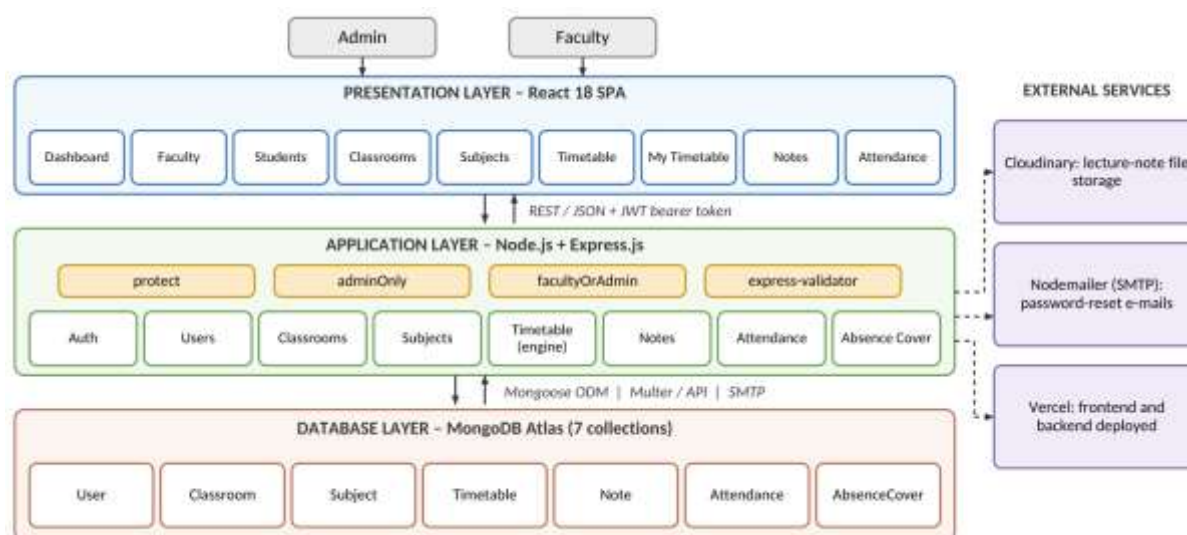


Figure 1 Three-tier architecture of ScheduleEase with external services

The user interaction layer provides a web-based dashboard that allows administrators to enter data, generate timetables and monitor the institution, and allows faculty to view their schedule, share notes and mark attendance. Global authentication

state is held in a React Context, navigation is handled by React Router, and Axios attaches the access token to every request. The application layer consists of RESTful services that handle user requests, run the scheduling engine and coordinate the other components, and request validation is performed with express-validator before a controller acts on the data. External services are used where a specialized service is the better choice: Cloudinary stores lecture-note files, Nodemailer sends password-reset e-mails, and Vercel hosts both the frontend and the backend.

Fig. 2 shows the seven MongoDB collections and their relationships. A User document stores the role (admin or faculty), the hashed password, availability and weekly workload. A Subject references its faculty member and records theory and lab hours, syllabus completion, department and semester. A Classroom stores capacity, type (lab or theory), availability and fixed-slot flags. A Timetable document represents the day/period slots of one department, semester and section and refers to the subjects, faculty and rooms placed in it. Note documents hold a title, the subject and up to ten Cloudinary file URLs; Attendance documents hold a class session, per-student marks and the data for summary reports; and AbsenceCover documents link a class and slot to a substitute faculty member and a cover status.

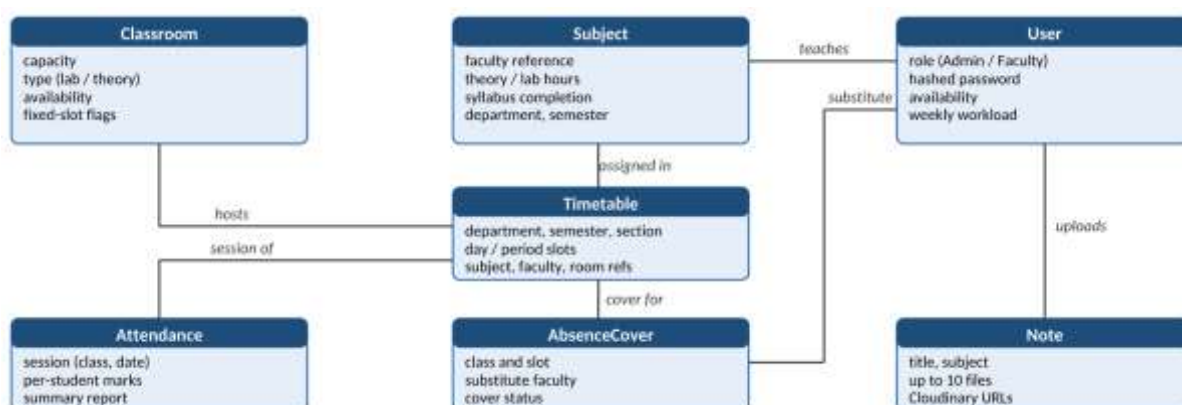


Figure 2 Data model: MongoDB collections and their relationships

Table 1 summarizes the technology stack. The NoSQL document structure suits flexible scheduling data and fast retrieval. Passwords are hashed with bcryptjs, incoming requests are sanitized by express-validator, and cross-origin access is restricted through CORS and environment-based configuration.

Table 1 Technology Stack of ScheduleEase

Layer	Technologies
Front end	React.js v18, React Router DOM, Axios, Context API (global auth state), custom dark-themed CSS
Back end	Node.js, Express.js REST API, JWT, bcryptjs, Multer, express-validator, dotenv, CORS
Database	MongoDB with Mongoose ODM, hosted on MongoDB Atlas
Services	Cloudinary (note storage), Nodemailer (e-mails), Nodemon (development), Vercel (deployment)

Security and access control follow the flow of Fig. 3. A successful login issues a signed JSON Web Token [11] after the bcryptjs password check; the React client attaches the token to each Axios request, the protect middleware verifies it, and the adminOnly or facultyOrAdmin middleware then decides whether the role may reach the controller. A request with an invalid token or an insufficient role is rejected with a 401 or 403 status before any controller code runs. Passwords are stored only as bcryptjs hashes, based on the adaptive hashing scheme of [12]. Table II summarizes the resulting permissions: only administrators enter master data, generate timetables, view any faculty timetable and receive substitute suggestions, both roles can view their own timetable and mark attendance, and only faculty report absences.

**Figure 3 JWT authentication and role-based authorization flow****Table 2 Role-Based Permissions**

Capability	Admin	Faculty
Enter faculty, student, classroom and subject data	Yes	No
Generate timetables	Yes	No
View any faculty timetable	Yes	No
View own timetable (My Timetable)	Yes	Yes
Mark attendance sessions	Yes	Yes
Report an absence	No	Yes
Receive substitute suggestions	Yes	No

METHODOLOGY AND WORKFLOW

A. Overall Methodology

ScheduleEase follows a modular, data-driven methodology that supports the administrator from data entry to timetable publication and the faculty from schedule

viewing to absence reporting. The administrator first enters faculty profiles, subjects, classroom and laboratory information, student records and weekly teaching hours. The scheduling engine then loads the data for the chosen scope and places sessions one at a time, verifying every constraint before a slot is confirmed. The resulting timetable is stored, displayed as a grid and exposed to faculty through a personalized view. Changes in faculty, subject or classroom data can be followed by regeneration of the affected timetable.

B. Mathematical Formulation of the Scheduling Constraints

The scheduling problem solved for one scope can be stated formally. Let U be the set of sessions to be placed, D the working days, P the periods in a day, R the classrooms, F the faculty and B the student batches. For a session u , let $f(u)$ be its faculty member, $b(u)$ its batch, $s(u)$ its subject and $l(u)$ its type (theory or lab); a lab session occupies consecutive periods. The binary variable $y_{u,d,p,r}$ equals 1 if session u occupies classroom r on day d in period p , and 0 otherwise. Let $a_{f,d,p}$ be 1 when faculty member f is available in that period, cap_r the capacity of room r , n_b the strength of batch b , and Z the set of fixed special slots. The hard constraints are:

$$\sum_u y_{u,d,p,r} \leq 1 \quad \text{for every } d, p, r \quad (1)$$

$$\sum_{u: f(u)=f} \sum_r y_{u,d,p,r} \leq a_{f,d,p} \quad \text{for every } f, d, p \quad (2)$$

$$\sum_{u: b(u)=b} \sum_r y_{u,d,p,r} \leq 1 \quad \text{for every } b, d, p \quad (3)$$

$$y_{u,d,p,r} = 0 \quad \text{if } cap_r < n_{b(u)}, \text{ or } l(u) = \text{lab and room } r \text{ is not a laboratory} \quad (4)$$

$$y_{u,d,p,r} = 0 \quad \text{for every } (d, p) \text{ in } Z \quad (5)$$

$$\sum_{u: b(u)=b} \sum_{p,r} y_{u,d,p,r} \leq M \quad \text{for every } b, d \quad (6)$$

$$\sum_{u: s(u)=s} h(u) = H_s \quad \text{for every subject } s \quad (7)$$

Equations (1)–(3) forbid classroom, faculty and student-batch clashes; (4) enforces room capacity and room type; (5) keeps fixed events clear of regular classes; (6) limits the number of periods per batch per day to M ; and (7) states that the sessions created for a subject add up to its weekly hours H_s , where $h(u)$ is the duration of session u . Beyond feasibility, the engine prefers an even distribution of teaching load. With $W_{f,d}$ the periods taught by faculty member f on day d and μ_f the mean of these values over the week, the soft objective is given in (8). Infrastructure use can be tracked through the room utilization ratio $\rho_r = (\text{periods occupied in room } r) / (|D| \times |P|)$.

$$\text{Minimize } \sum_f \sum_d (W_{f,d} - \mu_f)^2 \quad (8)$$

ScheduleEase does not solve (1)–(8) with a general-purpose solver; it uses the rule-based search of the next subsection, in which these constraints are the rules checked before each slot is confirmed.

C. Proposed Scheduling Algorithm

The scheduling engine is admin-triggered and constraint-aware. Subjects, faculty and classrooms are assigned to weekly day/period slots, and room, faculty and subject-hour requirements are checked before a slot is confirmed. Generation can be re-run on demand whenever faculty, subject or classroom data change. Fig. 4 shows the flowchart of the core loop, which proceeds as follows. For the selected department, semester and section, the engine first loads faculty, classrooms, subjects, weekly hours and fixed slots, and then selects the next (subject, faculty) session that still needs a slot. It proposes a candidate day and period together with a classroom that fits the batch size and the type of the class, so that laboratory classes go only to laboratories, and verifies the proposal against the five conflict types of Table III.

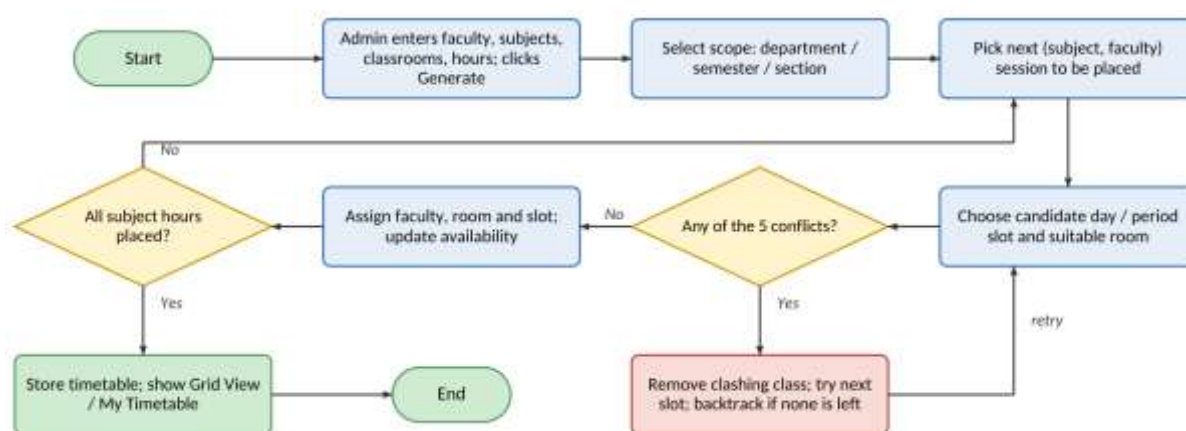


Figure 4 Flowchart of the constraint-aware timetable generation process. If no slot remains after backtracking, the conflict is reported, alternatives are suggested and generation is repeated after the data is corrected.

If no conflict exists, the faculty member, room and slot are assigned and availability is updated; otherwise the clashing class is removed and the next slot is tried. If no slot is left for a session, the engine returns to the previous session and tries an alternative placement. When all subject hours are placed the timetable is stored and shown in the grid view; if the instance is infeasible the conflict is reported with alternative suggestions and generation can be repeated after the data is corrected.

D. Conflict Detection Mechanism

Detection runs continuously during slot assignment and covers the five conflict types in Table III. On a conflict, the engine removes the conflicting class, tries the next available slot, applies backtracking if the conflict persists, and regenerates the timetable if necessary.

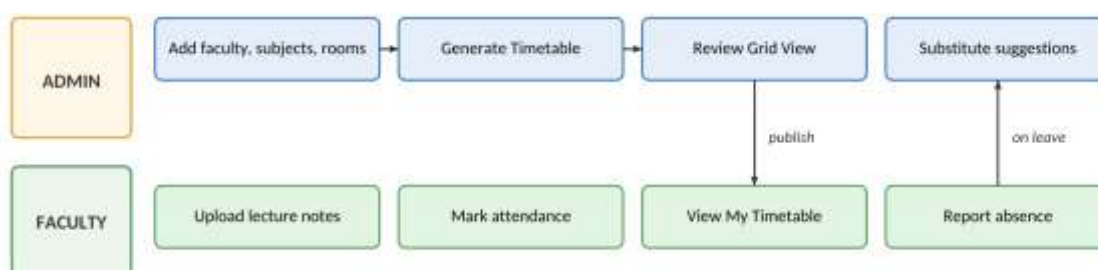
Table 3 Conflict Types Checked by the Engine

Conflict	Condition detected
Faculty	Faculty assigned to two classes at once, or during unavailability
Classroom	Two batches in one room, or lab class placed in a non-lab room
Student batch	A batch has two subjects in the same time slot
Subject hour	Hours exceed weekly requirement, or practical hours split incorrectly
Special slot	Fixed events (NCC/NSS, meetings) overlap regular classes

The five conflict types map directly onto the formal constraints. Classroom conflicts correspond to (1) and to the room-type rule in (4), faculty conflicts to (2), student-batch conflicts to (3), special-slot conflicts to (5), and subject-hour conflicts to (6) and (7). Because each rule is checked before a slot is confirmed, an administrator who asks why a session was placed where it is, or why a scope could not be completed, receives an answer in terms of the rule that allowed or blocked the placement.

E. Timetable Generation Interface

The administrator first adds faculty, students, classrooms and subjects and assigns faculty to subjects with weekly hours. On the Timetable page, Generate Timetable calls the auto-scheduling API, which returns a conflict-free weekly grid for review; faculty open their own schedule under My Timetable. Fig. 5 summarizes the end-to-end workflow of both roles, and Fig. 6 shows the layout of a generated weekly grid, in which lab sessions occupy consecutive periods and fixed slots are kept clear of regular classes.

**Figure 5 End-to-end usage workflow for the Admin and Faculty roles**

Day	P1	P2	P3	P4	P5	P6
Mon	S1	S2	S3	Lab A		S5
Tue	S4	S1	S5	S2	S3	S4
Wed	S2	S3	Lab B		S1	S4
Thu	S5	S4	S1	S3	S2	S5
Fri	S3	S1	S2	S4	NCC / NSS	

Figure 6 Illustrative weekly Timetable Grid View (S1–S5 denote subjects; white = theory, green = lab in consecutive periods, orange = fixed slot; layout only, not measured data)

F. Smart Absence-Cover Workflow

When a faculty member reports an absence, the system queries substitutes who are free during the affected slot, records the cover status and reports the effect on syllabus

completion (Fig. 7). The administrator chooses a substitute from the suggestions and the cover status becomes Assigned; if nobody is free, the status stays Uncovered, the administrator is notified and the class is rescheduled manually. For a class c held by faculty member $f(c)$ in period (d, p) , the candidate substitutes are given by (9).

$$\text{Sub}(c) = \{ g \text{ in } F, g \neq f(c) : a_{g,d,p} = 1 \text{ and } \sum_{u: f(u)=g} \sum_r y_{u,d,p,r} = 0 \} \quad (9)$$

Equation (9) guarantees that every suggested substitute is both available and free of other classes in the affected period, so an absence never produces a secondary conflict. Candidates whose subject matches the missed class are the natural first choice.

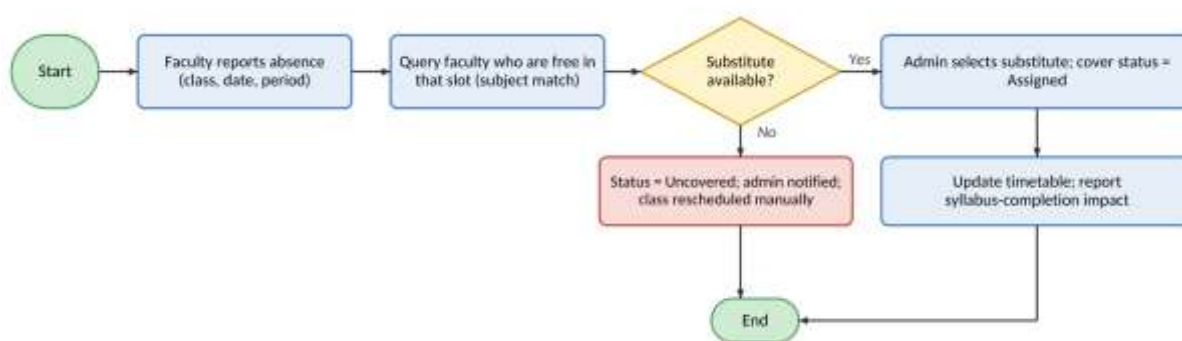


Figure 7 Flowchart of the smart absence-cover module

G. Data Flow, Regeneration and Traceability

Every stored timetable is derived from the master data held in the database, so the data flow is deliberately simple. When an administrator edits faculty availability, subject hours, classroom capacity or a fixed event, the affected scope is regenerated and the personal faculty views, which are read from the stored timetable, reflect the change at once. Leave periods, events and room-maintenance flags are held as availability fields, so that the next generation avoids them automatically. Detailed logs of timetable changes and approvals are kept for traceability [5], which allows an institution to review who changed a schedule and when.

H. Search Behavior and Scope

Because the underlying problem is NP-complete in general [9], the placement search can in the worst case examine a very large number of partial timetables, and the backtracking step is what makes the engine complete for the instances it can reach. Two design decisions keep the search manageable in practice. First, generation is scoped per department, semester and section, so each run involves only the sessions, faculty and rooms of one scope rather than the whole institution. Second, the five conflict checks are applied before a slot is confirmed, which prunes infeasible placements immediately instead of discovering them after the timetable is complete. The present paper does not report timing measurements for these runs; measuring them on institutional data is part of the evaluation plan below.

IMPLEMENTATION AND EVALUATION

A. Implementation

ScheduleEase is implemented as a software-only web application on the MERN stack. The React 18 frontend uses React Router DOM for navigation, the Context API for global authentication state and Axios for requests, with custom dark-themed CSS. The backend is a Node.js and Express.js REST API that uses JWT for authentication, bcryptjs for password hashing, Multer for file uploads, express-validator for input validation, dotenv for environment configuration and CORS for restricted cross-origin access. Data is stored in MongoDB through the Mongoose ODM and hosted on MongoDB Atlas, lecture-note files are stored on Cloudinary, password-reset e-mails are sent through Nodemailer, and both the frontend and the backend are deployed on Vercel.

The supporting modules follow the same pattern of explicit rules and role checks. Faculty upload lecture notes through Multer to Cloudinary, up to ten files per note, and the stored URLs are kept in the Note collection so that students and staff can browse and download them. Attendance sessions record per-student marks against a class session and produce summary reports and per-student reports, while subject management tracks syllabus completion, including bulk updates, which the absence-cover module uses to report the syllabus impact of a missed class. Password-reset links are delivered through Nodemailer, and every protected route is wrapped by the protect middleware and, where needed, by the adminOnly or facultyOrAdmin check.

B. Evaluation Against Manual Scheduling

Table IV contrasts ScheduleEase with the spreadsheet-based practice described in the problem statement. The comparison is qualitative: it describes the behavior of the implemented modules and does not report measured performance figures. Generation is automated and admin-triggered instead of manual slot-by-slot entry, conflicts are checked before each slot is confirmed instead of being found after the fact, and room tracking covers capacity, type and availability per room. Faculty leave is handled by suggesting substitutes from free faculty, the scope extends to every department, semester and section, and changes are absorbed by regeneration with a personal view for each faculty member.

Table 4 ScheduleEase Versus Manual Scheduling

Aspect	Manual / spreadsheet	ScheduleEase
Generation	Manual entry, slot by slot	Automated, admin-triggered
Conflicts	Found after the fact	Checked before each slot is confirmed
Room tracking	No real-time availability	Capacity, type and availability per room
Faculty leave	Manual rework	Substitutes suggested from free faculty
Scope	Single sheet, no multi-department support	Per department, semester and section
Changes	Hard to update and communicate	Regenerate on demand; personal faculty view

ScheduleEase is designed to deliver conflict-free timetables for all departments, semesters and programs with no clashes for students or faculty, a balanced faculty workload, better infrastructure use through the prevention of double-booking and the exposure of idle spaces, and real-time updates and rescheduling in response to leave, events or other changes. A central dashboard gives live visibility into faculty, students, classrooms, subjects and active timetables, and the generate-review-publish cycle reduces administrative effort per semester, speeds up timetable release and improves inter-department coordination. Role-based access control, JWT authentication and bcrypt-hashed passwords keep sensitive actions, such as generating timetables, managing users and deleting classrooms, restricted to authorized admin and faculty accounts.

ScheduleEase targets engineering colleges, universities and other institutions that run several departments, programs and sections from a shared pool of classrooms and faculty. For the academic office, it replaces repeated manual timetable preparation each semester, and it makes late changes such as a new elective or an unavailable room easy to absorb by regenerating the affected scope. A quantitative evaluation on real institutional data, covering generation time, the proportion of successfully completed scopes and room utilization, is the natural next step and is listed among the future directions.

Beyond the qualitative comparison, the design offers several properties that can be verified by inspection of the implemented modules. Every generated timetable satisfies the hard constraints by construction, because a slot is confirmed only after all five conflict checks have passed. The same availability data that drives generation also drives substitute suggestions, so the two functions cannot disagree about who is free. Access to every protected route is decided centrally by middleware rather than by individual controllers, which keeps the security behavior uniform across modules and simple to audit.

C. Evaluation Plan for the Proposed System

Following the reporting practice suggested by the surveyed literature, the system will be assessed on real institutional data using three quantitative measures: the time taken to generate a timetable for a scope, the proportion of scopes that are completed without an infeasibility report, and the room utilization ratio defined with the mathematical formulation. The workload-balance objective in (8) will be reported as the variance of teaching periods per faculty member across the week. In addition, the access-control behavior of Fig. 3 will be tested by issuing requests with valid, invalid and insufficient-role tokens, and the absence-cover module will be tested by simulating absences in fully loaded and lightly loaded timetables to confirm that every suggested substitute is genuinely free in the affected period.

D. Challenges and Limitations

Although ScheduleEase automates the main steps of timetable preparation, certain limitations remain. The engine is rule-based and uses backtracking, so it finds a feasible

timetable when one is reachable but does not guarantee an optimal one, and very tightly constrained scopes may be reported as infeasible until the input data is adjusted. The quality of the result depends on the accuracy of the entered data, including faculty availability, subject hours and room capacities. The absence-cover workflow suggests substitutes and leaves the final choice to the administrator, and lecture-note storage and password-reset e-mails depend on external services whose availability and limits follow those of the providers. The current evaluation is functional and qualitative and has not yet been benchmarked on large multi-department institutional datasets, and advanced capabilities such as predictive scheduling, real-time room monitoring and learning-management integration are not part of the present implementation.

E. Future Directions

Several enhancements can extend ScheduleEase, grouped in Fig. 8 into three themes. Under intelligence, machine learning on attendance patterns, leave trends and busy hours can support AI-driven predictive scheduling, assign substitutes automatically by subject expertise, free slots and departmental priority, and answer timetable and room queries through voice or chatbot support. Under access and experience, a mobile application can provide instant timetable access, push notifications and emergency updates, and advanced dashboards can add timetable heat maps, utilization graphs and workload charts. Under integration and scale, LMS integration can sync course plans, materials and deadlines with Moodle or Google Classroom, IoT or QR-based monitoring can verify room usage and reassign unused rooms, and cloud auto-scaling on AWS, Azure or GCP with backup can handle multi-branch peak load.

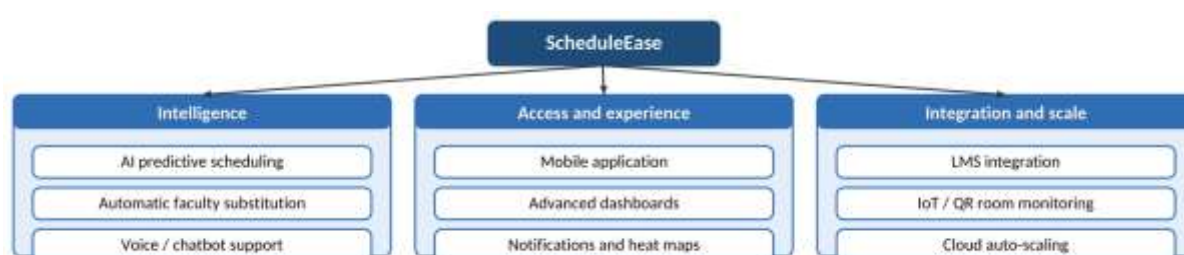


Figure 8 Roadmap of planned enhancements grouped by theme

CONCLUSION

This paper presented ScheduleEase, a smart institutional timetable management system that addresses efficient, conflict-free academic planning under the complexity introduced by NEP 2020, electives and dynamic faculty availability. The system captures key parameters, namely classrooms, faculty workload and availability, subject distribution, class strength, teaching hours and fixed slots, and produces timetables that minimize faculty burden, eliminate clashes and maximize classroom and laboratory utilization.

Through rule-based conflict avoidance, JWT-secured role-based access and the absence-cover workflow, the platform improves accuracy and reliability, and its React-and-

Node design is easily deployable across departments, semesters and sections. Combining automation with role-based data management yields a robust scheduling system for modern institutions. With the planned enhancements, namely predictive scheduling, automatic substitution, mobile access and LMS integration, the platform can evolve into a comprehensive academic management framework.

REFERENCES

- [1] X. Zhang, Y. Ding, X. Huang, et al., "Smart Classrooms: How Sensors and AI Are Shaping Educational Paradigms," *Sensors (MDPI)*, 2024.
- [2] D. Zhou, "The Influence of Smart Classrooms on Students' Learning Experience in the Digital Age," *Advances in Educational Technology and Psychology*, 2024.
- [3] I. Nasar, et al., "Artificial Intelligence in Smart Classrooms: An Investigative Learning Process," *Asian Journal of Applied Education*, 2023.
- [4] X. Gu, et al., "From Integer Programming to Machine Learning: Solving University Timetabling Problems," *Computation (MDPI)*, 2025.
- [5] O. Cayli and A. Yilmaz, "Smart Classroom Scheduling and Resource Optimization Using AI," *European Journal of Research and Development*, 2025.
- [6] D. G. Sampson, D. Ifenthaler, and P. Isaias (eds.), *Smart Learning Environments in the Post-Pandemic Era*, 2024.
- [7] M. D. Smith, *The Abundant University: Remaking Higher Education for a Digital World*, MIT Press, 2023.
- [8] Ministry of Education, Government of India, *National Education Policy 2020*, 2020.
- [9] S. Even, A. Itai, and A. Shamir, "On the Complexity of Timetable and Multicommodity Flow Problems," *SIAM J. Comput.*, vol. 5, no. 4, pp. 691–703, 1976.
- [10] A. Schaerf, "A Survey of Automated Timetabling," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 87–127, 1999.
- [11] M. Jones, J. Bradley, and N. Sakimura, *JSON Web Token (JWT)*, RFC 7519, Internet Engineering Task Force, 2015.
- [12] N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," in *Proc. USENIX Annual Technical Conf.*, 1999, pp. 81–91.