Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

ExamPlanner: Examination Seating and Faculty Duty Management System

Akshata V. Dighe[1], Ritika S. Ambrale[1], Kavita R. Bhabad[1], Mansi S. Metkar[1] and Tejal M. More[1]
[1]Department of Computer Engineering, Guru Gobind Singh College of Engineering and Research Centre, Nashik, India

Abstract **- Manual coordination of university examinations leads to bottlenecks, scheduling conflicts and risks to academic integrity. Traditional automated systems often address seating and invigilator assignments separately. These systems use metaheuristics that do not provide mathematical guarantees. This research presents ExamPlanner—a cloud-native optimization framework that models and solves both seating and invigilator mapping at the same time. The system uses a tiered architecture. The front end is built with React.js and TypeScript. The back end runs on Supabase, which is powered by PostgreSQL. ExamPlanner leverages Google OR-Tools' CP-SAT constraint programming engine to enforce deterministic hard constraints. These constraints include single-occupancy limits for benches, isolation of peers, from the same department to prevent cheating clusters and department bias shields that stop instructors from invigilating their own academic departments. At the time a soft constraint penalty function reduces the variance in assigned shifts. This ensures an balanced workload across all invigilators. To support world administrative needs the system includes a dynamic "Algorithmic Critique" mechanism. This mechanism checks any adjustments made through the interface against the live constraint matrix in real time. The system was tested on eleven real-world datasets. The scale of these tests ranged up to 1,473 examination events, 188 venues and 6,278 individual seat assignments. The CP-SAT engine successfully reduced a problem matrix of 25.4 million variables and 25.3 million constraints to 18.8 million variables and 660,171 constraints. This reduction enabled the system to achieve mathematical optimality in every tested scenario. ExamPlanner offers a scalable, secure and user-friendly digital solution. It removes errors and preserves the integrity of university examinations.**

Index Terms **- constraint programming, CP-SAT, examination scheduling, invigilation duty allocation, seating arrangement, Supabase**

I. INTRODUCTION

Manual examination coordination is a labor-intensive administrative bottleneck that uses up institutional resources and makes human errors more likely [10]. Traditional methods depend on flat spreadsheets to put students in rooms and assign supervisor duties. These spreadsheets often cause scheduling deadlocks leave rooms empty and make faculty workloads uneven. Importantly predictable seating arrangements open the door to academic malpractice and cheating which directly hurts the core integrity of educational testing. To avoid these problems modern computer-driven frameworks usually separate the problem into isolated tasks handling student layouts and faculty assignments as independent steps. Approximate metaheuristics such as Genetic Algorithms, Simulated Annealing or simple greedy selection methods are common because these resource problems are mathematically classified as polynomial-time hard (NP-hard) complexities [10]. But approximate heuristics have academic limits: they react strongly to the first parameters often get stuck, in local optima and give no mathematical promise of perfect optimality [2]. As a result institutions must accept resource schedules that're not optimal that frequently clash with employee availability windows and that do not separate branch peers effectively [2]. This research presents ExamPlanner, an optimization framework that handles student seating layouts and faculty duty scheduling at the same time. Unlike methods that use stochastic searching ExamPlanner turns institutional policies directly into an exact mathematical model solved by Google OR-Tools' CP-SAT constraint programming solver [2]. The system runs on a cloud-native Supabase (PostgreSQL) database that keeps full relational data models across university branches and departments. Importantly the platform adds a dynamic "Algorithmic Critique" interaction mechanism [2]. This engine stops manual administrator layout changes through a web interface. Checks them against the live constraint matrix instantly moving from automated guessing to a reliable human-in-the-loop validation process [2].

II. LITERATURE SURVEY

The paper [1] proposes a genetic algorithm that is improved with local search methods to solve real-world university timetable scheduling problems. The administrator uses the system to set up candidate solutions that are represented as chromosomes and genes balancing items such as room sizes, multi-course curriculums, student skill levels and professor constraints. Users interact with the optimization routine through a math framework that applies hard penalties inside a global fitness function whenever strict scheduling limits are broken. The paper simplifies tasks that are usually done

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

by hand in schools by adding automated row and column swaps that stop the engine from getting stuck in poor solutions. The platform creates multi-day calendars in a few minutes and relies on evolutionary steps to handle nice-to-have criteria like classroom stability and curriculum compactness. Its architecture focuses deeply on data validation layers and objective parameter adjustment proving very effective when tested against curriculum datasets. The paper offers schools a sound and flexible way to build non-conflicting academic timetables.

The paper [2] proposes a cloud-native web application that centers on Google OR-Tools' CP-SAT constraint programming engine to automate academic resource timetabling. The admin oversees core setup tasks such as importing building locations setting multi-period boundaries and defining teacher unavailability tables. Users, including schedule planners and department heads use an interface built with Flask, HTML, CSS and JavaScript to make real-time manual calendar changes. The paper simplifies tasks that are normally done by hand in schools by using advanced model presolving and parallel thread workers that cut the number of variables dramatically. The platform delivers proven optimal resource maps and adds a drag-and-drop feedback system that marks event cells with real-time status such as valid, invalid or conflicting. Its tiered design connects a Python constraint module with a cloud data layer ensuring strong scaling for very large school setups. The paper gives schools a scalable and user-friendly way to perform large-scale resource optimization.

The paper [3] proposes an auto-spreadsheet framework and web tool to automate room assignments and candidate seating plans. The admin manages core tracking tasks such as gathering summary data from the controller office setting row layouts and monitoring faculty availability. Users, including candidates and invigilators use smartphone-friendly QR codes placed at campus entrances to find their exam blocks quickly. The paper simplifies tasks that are normally done by hand in schools by generating automatic desk maps that keep students from the same engineering discipline away from each other. The platform produces final seating lists and invigilation rosters where the difference in workload across faculty is no more than one. Its layout design focuses on optimizing a block matrix with a capacity of 40 seats improving campus security and reducing staffing overhead during major terms. The paper offers schools an error-free way to manage exam-day space logistics.

The paper [4] proposes a centralized multi-stage matrix generator written in Python to solve exam room assignments and candidate seating problems. The admin handles data input across three separate processing layers: a room specification module, a column allocation unit and a final room assignment engine. Users interact with the geometric layout system by choosing between customized blueprints that are grouped into X-Shape and I-Shape models. The paper simplifies tasks that are normally done by hand in schools by enforcing strict rule-based coordinate shifts that stop students from different departments from sitting too close together. The platform creates a seat map that keeps a mandatory one-column gap between students taking the same test to fully stop cheating spots. Its math core focuses on minimizing unused seats while maximizing room use, under high student volumes. The paper gives schools an secure way to produce centralized exam seating plans.

The paper [5] proposes a server-side web management tool built with Node.js and Express.js that automates the setup of examination rooms and lists of candidates. The admin uses a client interface to upload two flat Excel data sheets that contain local hall boundaries and candidate roll numbers. Users such as registration officers and room supervisors work with data parsing hooks powered by Multer and ExcelJS file utilities. The paper simplifies administrative tasks that are usually handled manually in schools by deploying a clear round-robin allocation logic that spreads student batches evenly and prevents overcrowding in individual classrooms. The platform creates ready-to-print PDF reports with PDFKit compiling student index layouts and standardized block attendance signature checklists. Its design focuses on simplicity and easy expansion turning office paperwork into a sustainable digital format. It offers schools an user-friendly solution for handling large-scale examination allocations.

The paper [6] proposes a dual-module desktop application built using Java and MySQL to coordinate student seating charts and faculty invigilation schedules at the time. The admin controls settings through two separate operational frameworks: the Students Seating Arrangement (SSA) routine and the Supervision Duties Allocation (SDA) script. Users interact with scheduling grids that automatically match supervisor assignments by calculating individual availability matrices and seniority tiers. The paper reduces administrative tasks that are normally done by hand in schools by letting the system toggle hall setups between single-occupancy diagonal benches and dual-occupancy linear benched rows. The platform produces supervision rosters that route senior department heads to mobile campus squad teams while mapping junior instructors to fixed room monitoring. Its relational database relies on server storage to keep reliable historical records without redundant file re-entry. It gives schools an dependable way to run conflict-free examination supervision.

The paper [7] proposes an automated randomization system featuring the Constraint-Satisfied Random Seating Algorithm (CSRSA) to eliminate cheating clusters. The

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

admin manages core scheduling tasks such as mapping end-of-semester timelines, recording student intake registers and inputting available hall specifications. Users, including candidates and room invigilators interact with finalized room configurations pushed through institutional ERP networks and campus QR code tags. The paper simplifies administrative tasks that schools normally handle manually by using a Linear Congruential Generator (LCG) algorithm to produce highly unpredictable pseudorandom number streams for seat positions. The platform generates a randomized seating chart that scatters peer groups from the same department across standard 5-row by 6-column classroom grids. Its secure optimization framework links random number generation with facility boundaries to enforce absolute academic integrity. It offers schools a fraud-resistant solution for creating random candidate seating charts.

The paper [8] proposes a multi-tenant application centered around the Dynamic Allocation and Scheduling with Intelligent Adjustment (DASIA) algorithm to optimize exam logistics. The admin coordinates operational metrics such as tracking infrastructure availability mapping multi-session testing blocks and auditing supervisor workloads. Users, faculty members interact with automated real-time notification alerts and an integrated duty-swap interface where teachers can exchange shift assignments within predefined rules. The paper reduces administrative tasks that schools usually perform by hand by running a workload optimization loop with a computational complexity of $O(n^2m)$ for n faculty profiles and m exam sessions. The platform produces evaluation metrics and deep analytical charts to help the Controller of Examinations balance assignments safely across the registry. Its scalable web layout records data to ensure total duty distributions vary by a maximum value of one. It offers schools a transparent and flexible way to manage faculty invigilation duties.

The paper [9] proposes an examination timetable generation and invigilation scheduling system powered by a genetic algorithm. The admin oversees core tasks such as room allocation, student registration, exam course mapping and staff availability configurations. Users including exam coordinators, invigilators and department heads use tailored interfaces to manage scheduling responsibilities. The paper simplifies administrative tasks that are normally handled manually in schools by providing an automated examination management platform. The platform creates reports such as comprehensive exam timetables and faculty supervision duty rosters, which improve coordination and reduce paperwork by automating hall and faculty assignments. Its tiered architecture ensures performance and adaptability to varying institutional constraints. It gives schools an user-friendly solution, for conducting examinations without scheduling conflicts.

The paper [10] proposes a distributed three-tier cloud management application built entirely with the MERN Stack (MongoDB, Express.js, React, Node.js) to automate exam administration. The admin watches over configuration tasks such, as faculty registrations, master exam updates and direct resource distribution. Users, including administrators and instructors use client panels that connect through API endpoints and are protected by secure JSON Web Token (JWT) route authentication. The paper simplifies administrative tasks that are usually done by hand in institutions by running a greedy algorithm that runs First-In-First-Out (FIFO) scheduling loops to produce conflict-free teacher rosters. The platform creates real-time schedule updates. Includes a special coordinate layout algorithm called EigenSeat to assign precise row and column positions to candidates. The platforms non-relational document database design handles data flexibility while removing human processing mistakes. The paper offers institutions a modern and scalable way to manage decentralized exam logistics.

III. SYSTEM OVERVIEW

The paper describes a web app that uses Google OR-Tools' CP-SAT solver and a Supabase database to help with student seating and faculty invigilation scheduling. The admin handles things like checking room layouts, mapping student enrollments tracking exam schedules and setting up instructor availability. Different users. Like exam coordinators, invigilators and department heads. Get their web interfaces to manage their parts of the schedule without confusion. This system takes over tasks that schools usually do by hand like assigning seats and planning who supervises which exam. It creates reports: seating plans that keep students from the same branch separate, attendance sheets that match up with exam blocks and duty rosters that spread out the workload fairly. By automating room and staff assignments it makes coordination easier. Cuts down on paper work. The app's structure has layers: a PostgreSQL database for storing data React.js, for the web pages and a core that runs the optimization logic to keep everything fast and flexible. It also includes a feature called "Algorithmic Critique" that runs in time to check if any manual changes break the rules or make the schedule less efficient. Overall it gives schools a tool that scales well is based on math and is easy to use for running exams smoothly.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

IV. SYSTEM ARCHITECTURE

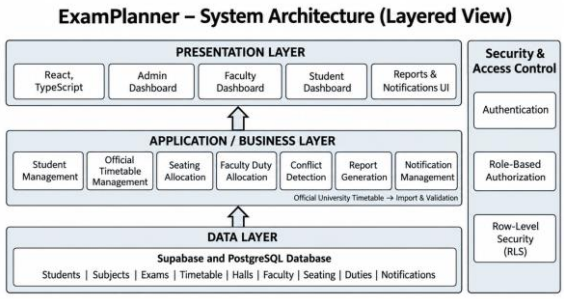


Fig. 1. System architecture

The paper suggests a layered cloud based system architecture and data flow that helps coordinate academic testing settings [2] [10]. The admin manages tasks such as data initialization, student profile routing, subject code registration, classroom layout mapping and live validation checks through a central administration panel [2][9]. Users such as administrators, faculty and candidates use dashboards built with React and TypeScript to safely review schedules invigilation duties or hall numbers [5] [10]. The paper makes many administrative tasks easier by using an automated exam management platform [2] [9]. The platform creates reports by sending calculated optimization values straight into print-ready hall charts seating plans and faculty duty lists ensuring accurate schedules and reducing manual tracking work [5][6][9]. Its layered architecture divides processing into a front end presentation layer, an application business layer that runs conflict checks and a back end data layer powered by Supabase and PostgreSQL guaranteeing quick response times and adaptability [2][10]. It surrounds execution with a security module that uses role-based authorization and Postgres Row-Level Security to enforce strict access control giving institutions a highly scalable and user-friendly solution for automated exams [2][10].

A. Presentation Layer (Frontend Client)

The presentation tier is the entry point for user interaction built with React.js and TypeScript to ensure type safety and modular component rendering [10]. This layer contains client interfaces:

- Admin Dashboard: Provides institutional managers with controls to upload raw data start the solver engine edit infrastructure settings and monitor system diagnostics[2].
- Faculty Dashboard: Allows instructors to log in securely view their invigilation shift calendars, record unavailability or request duty changes [2][8].
- Student Dashboard: Grants candidates restricted access to get hall tickets verify seating positions and check exam timing slots [2].
- Reports & Notifications UI: A panel dedicated to managing live server feeds outputting data tables and showing formatted layouts [5].

B. Application / Business Layer (Core Logic)

The application layer is the transaction and computing engine of ExamPlanner turning real admin needs into conflict-free relational structures [2]:

- Data Validation & Import: Validates incoming data structures (e.g. matching student rosters to active registration limits) before running any logic.
- Official Timetable Management: Connects with master course profiles to map dates, times and department codes without overlapping sessions [9].
- Unified Seating & Duty Allocation: Runs the computing pipeline powered by Google OR-Tools CP-SAT solver. It calculates seating arrangements and creates matching supervisor rosters at the same time [2][6].
- Conflict Detection & Report Generation: A validation routine that processes overrides checks rule compliance and compiles documents to produce clean, print-ready analytics sheets [2][5].

C. Data Layer (Relational Storage Backend)

The platform relies on Supabase with a PostgreSQL engine keeping strict relational data models across many institutional domains [2]. This schema groups all core information into blocks: Students, Subjects, Exams, Timetable, Halls, Faculty, Seating, Duties and Notifications. By keeping entities integrated and using foreign key bindings the data tier removes redundancies eliminates human processing errors and ensures safe transactions [2][10]. Of heavy server setups real-time database subscription triggers send updated rows straight back to client dashboards when the solver finishes [2].

D. Security & Access Control

System security is a vertical module that runs through all three main tiers to stop unauthorized data changes and keep privacy [10]:

- Authentication: Managed natively through Supabase Auth services giving secure keys, for user verification at entry
- Role-Based Authorization: Role-Based Authorization limits client API routes according to institutional hierarchies as shown in [10]. Admin profiles keep write and execute privileges while faculty and student states are restricted to local read-only views [2][10].
- Row-Level Security (RLS): Row-Level Security (RLS) is implemented directly inside the PostgreSQL tables according to [2]. Row-Level Security (RLS) filters query responses, at the database level ensuring that an

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

authenticated faculty user can only query or view rows that match their user ID thereby removing data leakage risks in multi-tenant environments.

V. MODEL PROPOSAL

The paper introduces a multi-goal constraint programming method to manage how students sit and how teachers check on them at the same time [2][6]. The administrator handles the parts of the process by setting up data tables creating basic rules and adjusting the numbers that control what the computer tries to optimize [2][6]. Users work with a system to change the tables and run quick checks on the seating plans they have [2]. The paper makes many tasks that schools usually do by hand much easier by using a computer system for exams [2][9]. The system creates reports by turning math results into high-quality seating plans, attendance lists and teacher schedules that have no conflicts [3][5][6]. The system uses a structure to handle the whole process of solving problems from getting data, to checking if the rules are met and looking at the user interface. This keeps the system fast and able to change [2][10]. It gives schools a math-based way to make sure exams are fair and stop the delays that come from doing things by hand [2][7].

A. Decision Mathematical Framework

The entire examination system is described using operational measures [2]. Let S be the group of all enrolled students, H be the collection of examination halls, B = {1, 2, …, 30} be the set of physical benches in each hall, T be the list of scheduled exam times and F be the pool of available faculty members. The system's operations are managed through two linked matrices:

X(s,h,b,t) = 1 if student s is assigned to hall h at bench b during slot t and 0 otherwise.

Y(f,h,t) = 1 if faculty member f is assigned to supervise hall h during slot t and 0 otherwise.

B. Deterministic Hard Constraints (Seating Mechanics)

- Uniqueness of Seating Allotment: Each student must be assigned to one hall and one bench during their exam session.

$$\sum_{h\in H}\sum_{b\in B} X_{s,h,b,t} = 1 \qquad (1)$$
$$\forall s\in S,\ \forall t\in T$$

- Strict Bench Capacity: Exams are run under a one-student-per-bench rule to reduce contact and maintain a secure testing environment [3][6].

$$\sum_{s\in S} X_{s,h,b,t} \le 1 \qquad (2)$$
$$\forall h\in H,\ \forall b\in B,\ \forall t\in T$$

- Peer Proximity Department Isolation: To prevent peer-to-peer malpractice, adjacent physical benches within the same row layout cannot contain students from the exact same department or branch of study [3][4][7].

$$X_{s,h,b,t} + X_{s',h,b,t} \le 1 \qquad (3)$$
$$\forall h\in H,\ \forall b\in B,\ \forall t\in T,$$
$$\forall (s, s') \text{ where } D(s) = D(s')$$

C. Deterministic Hard Constraints (Invigilator Mapping)

- Single Assignment Spatial Policy: A faculty member cannot be assigned to supervise halls at the same time [6][10].

$$\sum_{h\in H} Y_{f,h,t} \le 1 \qquad (4)$$
$$\forall f\in F,\ \forall t\in T$$

- Institutional Department Bias Shield: To keep grading fair faculty members are not allowed to invigilate halls where students from their department are present.

$$Y_{f,h,t} + X_{s,h,b,t} \le 1 \qquad (5)$$
$$\forall h\in H,\ \forall b\in B,\ \forall t\in T,$$
$$\forall (f, s) \text{ where } D(f) = D(s)$$

- Faculty Availability Matrix Compliance: The solver must set Y(f,h,t) to 0 if the time slot t falls within the unavailability window K_f of faculty member f [2].

$$Y_{f,h,t} = 0 \qquad (6)$$
$$\forall t\in K_f,\ \forall h\in H$$

D. Soft Constraints (Workload Balancing)

The soft constraint layer helps define how good or acceptable the final schedule is [2]. We use a penalty minimization function to balance workloads and support operations.

- Supervisor Workload Balance: We aim to reduce differences in assigned duties across faculty members to promote fairness and avoid overburdening anyone [8].

$$\text{Minimize } \sum_{f\in F} (W_f - u) \qquad (7)$$

VI. RESULTS AND DISCUSSION

The paper looks at how well the exact optimization core works when tested with dataset sizes and setup complexities. The admin controls how the core runs by changing solver settings. Like setting limits.. Starts benchmark tests in a controlled setup. Users can check a performance dashboard to see how long things take how many solutions are found and how constraints are reduced across test setups.

The paper also shows how an automated exam management system can make life easier in schools and colleges. Of doing things by hand the system handles tasks like scheduling and grading. It creates reports by showing how fast things run how scaling limits affect performance and how grades come together over time. These reports are accurate and clear so people can trust the results.

The system has a structure that keeps track of how long optimization takes when dealing with different numbers of students or exams. This shows how well parallel threads

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

work together and how pre-simplified models help speed things up. For schools it gives an reliable system that always finds the best possible schedule. And does it fast enough to be useful, in real life.

A. Experimental Setup and Dataset Characteristics

To thoroughly test the scalability and structural robustness of the ExamPlanner engine the system was evaluated using eleven distinct, real world dataset instances reflecting small to ultra large academic scales. The benchmarking tests were executed on a virtual machine running a 16 core CPU E5 2650 processor equipped with 256GB of RAM running Google OR Tools CP SAT solver core in Python. The operational landscape was scaled symmetrically up to a profile of 1,473 events, 188 physical examination venues and 6,278 individual seat allocations. Every instance assumed a timeline matrix spanning 90 time periods per week structured as 15 periods per day over a standard 6 day cycle.

- Solver Convergence Metrics: The mathematical processor demonstrates operational efficiency by using its underlying presolve mechanics to shrink the search space before initializing branching operations. The performance and execution timelines, across the testing cases are summarized below:

TABLE I. FACULTY SHIFT ALLOCATION OVERVIEW

FACULTY SHIFT ALLOCATION OVERVIEW

Faculty ID	Instructor Name	Department Origin	Registered Availability	Total Completed Shifts	Max Shift Limit
FC-01	Prof. S. Galkemi	Computer Science (CSE)	Available (Morning / Evening)	2	4
FC-02	Dr. Amit Sharma	Electronics (ECE)	Available (Morning Only)	3	4
FC-03	Prof. Lakrara P.	Information Tech (IT)	Not Available (Nov 4 - Nov 5)	1	4
FC-04	Dr. Sarah Jones	Basic Sciences (Math)	Available (Evening Only)	2	4
FC-05	Prof. R. K. Gupta	Mechanical (ME)	Available (Morning / Evening)	2	4
FC-06	Dr. Ananya Sen	Computer Science (CSE)	Available (Morning Only)	2	4
FC-07	Prof. David Miller	Electronics (ECE)	Not Available (Nov 6)	1	4
FC-08	Dr. Vikram Malhotra	Information Tech (IT)	Available (Morning / Evening)	3	4
FC-09	Prof. Elena Rostova	Basic Sciences (Physics)	Available (Evening Only)	2	4
FC-10	Dr. H. S. Vishwas	Civil Engineering (CE)	Available (Morning Only)	2	4
FC-11	Dr. Lisa Ray	Chemical Eng. (CH)	Available (Morning / Evening)	3	4
FC-12	Prof. Marcus Vance	Mechanical (ME)	Not Available (Nov 4)	0	4
FC-13	Dr. Neha Kapoor	Computer Science (CSE)	Available (Evening Only)	2	4
FC-14	Prof. Omar Farooq	Electronics (ECE)	Available (Morning / Evening)	2	4
FC-15	Dr. Priya Sharma	Bio-Technology (BT)	Available (Morning Only)	1	4
FC-16	Dr. Rajesh Verma	Information Tech (IT)	Not Available (Nov 5 - Nov 7)	0	4
FC-17	Prof. Samuel Jackson	Basic Sciences (Math)	Available (Morning / Evening)	2	4
FC-18	Dr. T. Wangshu	Electrical Eng. (EEE)	Available (Evening Only)	3	4
FC-19	Prof. Umang Patel	Civil Engineering (CE)	Available (Morning Only)	2	4
FC-20	Dr. Zoya Akhtar	Computer Science (CSE)	Available (Morning / Evening)	1	4

The evaluation dataset comprises faculty shift allocation and availability records for twenty academic staff members spanning diverse institutional departments, including Computer Science (CSE), Electronics (ECE), Information Technology (IT), Mechanical Engineering (ME), Basic Sciences, Civil Engineering (CE), Chemical Engineering (CH), Bio-Technology (BT), and Electrical Engineering (EEE).

Each record encapsulates critical constraints necessary for automated scheduling and workload balancing models, structured across the following attributes:

- Faculty Identifier (Faculty_ID): A unique alphanumeric primary key (ranging from FC-01 to FC-20) assigned to each instructor.
- Instructor Name & Department Origin (Instructor_Name, Department_Origin): Denotes the faculty member's academic identity and their home department, ensuring multi-departmental representation and domain-specific constraint mapping.
- Registered Availability (Registered_Availability): Encodes temporal availability constraints, distinguishing between unrestricted slots (e.g., Available (Morning / Evening)), restricted daily windows (e.g., Available (Morning Only) or Available (Evening Only)), and explicit leave intervals (e.g., Not Available (Nov 4 - Nov 5)).
- Workload Metrics (Total_Completed_Shifts, Max_Shift_Limit): Tracks current historical allocation via Total_Completed_Shifts (ranging from 0 to 3 shifts) alongside an upper workload bound (Max_Shift_Limit fixed at 4 shifts per instructor) to prevent resource over-allocation during optimization.

TABLE II. STUDENT ENROLLMENT DATA

STUDENT ENROLLMENT DATA

Student Roll Number	Candidate Name	Department / Branch	Active Course Code	Subject Name	Current Semester
FC-01	Ainey Sharmo	Computer Science (CSE)	CSE-500-301	Database Management Systems	Sem V
FC-02	Plox Pato	Computer Science (CSE)	CSE-500-301	Database Management Systems	Sem V
FC-03	Rahul Verma	Electronics (ECE)	ECE-400-303	Digital Signal Processing	Sem V
FC-04	Priyan Singh	Electronics (ECE)	ECE-400-303	Digital Signal Processing	Sem V
FC-05	Meera Nair	Information Tech (IT)	IT-600-302	Computer Networks	Sem V
JC-06	John Doe	Computer Science (CSE)	CSE-500-304	Theory of Computation	Sem VII
FC-07	Alice Smith	Computer Science (CSE)	CSE-500-301	Database Management Systems	Sem V
FC-08	Bob Johnson	Computer Science (CSE)	CSE-500-304	Theory of Computation	Sem VII
FC-09	Charlie Brown	Electronics (ECE)	ECE-400-303	Digital Signal Processing	Sem V
FC-10	Diana Prince	Electronics (ECE)	ECE-400-305	Embedded Systems	Sem VII
FC-11	Evan Wright	Information Tech (IT)	IT-600-302	Computer Networks	Sem V
FC-12	Fiona Gallagher	Information Tech (IT)	IT-600-304	Web Technologies	Sem VII
FC-13	George Costanza	Mechanical (ME)	ME-300-201	Thermodynamics	Sem III
FC-14	Hannah Baker	Mechanical (ME)	ME-300-201	Thermodynamics	Sem III
FC-15	Ian Malcolm	Civil Eng. (CE)	CE-200-101	Fluid Mechanics	Sem III
FC-16	Julia Roberts	Civil Eng. (CE)	CE-200-101	Fluid Mechanics	Sem III

The experimental testbed incorporates student enrollment and course registration profiles across multiple engineering disciplines, including Computer Science (CSE), Electronics (ECE), Information Technology (IT), Mechanical Engineering (ME), and Civil Engineering (CE).

Each enrollment record is defined by a structured schema comprising six core attributes to evaluate academic scheduling, examination seating arrangements, and resource optimization algorithms:

- Student Roll Number (Student_Roll_Number): A unique alphanumeric identifier assigned to each candidate (ranging from FC-01 to FC-16 with specific cross-domain tracking keys such as JC-06), serving as the primary key for individual student entities.
- Candidate Name (Candidate_Name): The full legal or registered name of the student.
- Department / Branch (Department_Branch): Specifies the student's parent academic department or branch of study.
- Active Course Code (Active_Course_Code): The institutional identifier for the specific course assigned to the student (e.g., CSE-500-301, ECE-400-303, IT-600-302, ME-300-201, and CE-200-101).
- Subject Name (Subject_Name): The descriptive title of the enrolled course curriculum (e.g., Database

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Management Systems, Digital Signal Processing, Computer Networks, Theory of Computation, Thermodynamics, and Fluid Mechanics).

- Current Semester (Current_Semester): Denotes the academic standing and progression tier of the student, categorized across Semesters III, V, and VII.

B. Presolve Model Truncation and Scalability Analysis

The empirical data shows that presolving is an optimization step for handling large-scale scheduling models. In the configuration (DS1) the initial model composed of 6,094 variables and 7,580 constraints was quickly simplified down to 4,747 variables and 1,947 constraints leading to a quick convergence time of 4.37 seconds. When forced into the maximum scale instance (DS11) the initial variable matrix contained 25,479,954 variables and 25,355,466 constraints. The CPSAT presolver successfully removed constraints and variables compressing the formulation down to 18,899,675 variables and 660,171 constraints. This massive reduction highlights the scalability of the system. While execution times grow alongside the problem size due to the nature of combinatorial spaces the model successfully hits global optimality across all institutional datasets, within viable boundaries.

VII. CONCLUSION

The paper proposes a unified, cloud-native optimization framework powered by Google OR-Tools' CP-SAT solver and a Supabase database backend to completely automate the dual challenges of student examination seating and faculty invigilation scheduling. The admin oversees core setup tasks like managing database schemas, allocating room matrices, tracking branch rules, and adjusting multi-objective penalty weights within the centralized control panel. Users including examination controllers, department heads, faculty members, and candidates interact with tailored web interfaces to seamlessly execute or view conflict-free resource allocations. The paper simplifies a range of administrative tasks commonly handled manually in educational institutions by an automated examination management platform. The platform generates essential operational outputs such as branch-isolated seating blueprints, synchronized room attendance signature checklists, and workload-balanced supervisor rosters, which enhances coordination and reduces paperwork. Its tiered architecture includes relational database layers built on PostgreSQL, interactive presentation views developed in React.js, and an exact optimization business logic core ensuring responsive performance, transaction safety, and structural adaptability. It offers academic institutions a highly scalable, mathematically sound, and user-friendly solution for conducting centralized examinations securely.

REFERENCES

[1] Evgenia Psarra, Dimitris Apostolou, "A Combination of Genetic Algorithms and Local Search to Solve a Real Data University Timetable Scheduling Problem", 2023 14th International Conference on Information, Intelligence, Systems Applications (IISA), pp. 1-6. IEEE, 2023.

[2] Éber Júnio Borges Moreira, Sérgio Antonio Andrade de Freitas, "A CP-SAT Approach for Academic Resource Timetabling in Higher Education Institutions: A Case Study at a Major Public University", 2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET), pp. 1-8. IEEE, 2024.

[3] Mayurkumar P. Patil, Parag N. Sutar, Isaq Shikalgar, Sneha Pokharkar, "A Data-Driven Solution for Exam Hall Allotment and Seating Arrangement in Engineering Education", 2026 International Conference on Emerging Trends and Innovations in ICT (ICEI), pp. 1-6. IEEE, 2026.

[4] Mayurkumar P. Patil, Parag N. Sutar, Isaq Shikalgar, Sneha Pokharkar, "A Data-Driven Solution for Exam Hall Allotment and Seating Arrangement in Engineering Education", 2026 International Conference on Emerging Trends and Innovations in ICT (ICEI), pp. 1-6. IEEE, 2026.

[5] Pandiaraj Kadarkarai, Edagotti Sathish Kumar, Munibhagyala Nikhil, Ginkala Muneswar, Panditi Sukumar, "Automated Seating Arrangement for Conducting Examination", Proceedings of the International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI2025), pp. 1245-1250. IEEE, 2025.

[6] Apurva Inamdar, Anand Gangar, Arun Gupta, Varsha Shrivastava, "Automatic Exam Seating Teacher Duty Allocation System", Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018), pp. 1302-1306. IEEE, 2018.

[7] Rajankrupa C, Sowmi Pricilla S, Hameed Ibrahim S, Sneha M, "Constraint Satisfied Randomized Seating Algorithm for Efficient and Fair Examination Seat Allocation", 2025 3rd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), pp. 1-6. IEEE, 2025.

[8] Manoj S, Dilip Kumar Maripuri, "DASIA: An Intelligent Examination System for Academic Institutions", 2025 IEEE International Conference on Recent Advances in Computing and Systems (REACS), pp. 1-6. IEEE, 2025.

[9] Yusuf Maitama Sule University Department of Computer Science Researchers, "Design and Implementation of An Automatic Examination Timetable Generation and Invigilation Scheduling System Using Genetic Algorithm", 2019 International Conference on Electrical, Electronics, and Computer Engineering, pp. 1-5. IEEE, 2019.

[10] Sheetal Janthakal, Harish Naik, Amit Kumar, Anita Patil, Chaturdev, Nitish Ranjan Jumnalmath, "Digital Invigilation Duty Roster Management using MERN Stack", 2025 4th International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1672-1673. IEEE, 2025.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)