WWW.IJRPR.COM

International Journal of Research Publication and Reviews

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

Automated Time Table Generator

***Yaswanthraj S*[*1], Abisheak S*[*2] ,Deepak S*[*3], Dharsan N*[*4], Hariprasath SK*[*5]**

*[1,2,3,4,5]Department Of Computer Science And Engineering (CyberSecurity), Sri Shakthi Institute Of Engineering And Technology, India.

I. ABSTRACT

The Automated Time Table Generator is a dynamic web-based project designed to simplify the manual burden of academic scheduling in schools and colleges. Most institutions struggle with creating class schedules due to constraints like teacher availability, room assignments, and subject distributions. This system is developed using JavaScript, React, and MySQL to automate timetable creation and provide easy interaction for different users. The key features include role-based login, real-time period highlighting, email-based period change requests, and PDF download of schedules. Each user has specific permissions: students can view timetables, faculty can request changes, and admins can manage everything. The current period is highlighted based on system time to enhance visibility. The scheduling algorithm prevents overlapping classes or rooms and follows a logical assignment pattern. With a clean UI and quick access, the system eliminates confusion and boosts planning efficiency. The paper discusses the challenges of manual scheduling, the proposed automated solution, and its technical implementation. It also outlines future improvements like AI-based optimization and mobile app integration.

II. INTRODUCTION

Timetable management is one of the core administrative activities in any educational institution. It involves the careful alignment of multiple variables such as subjects, teachers, classrooms, and time slots. In most traditional systems, this process is handled manually using spreadsheets or handwritten tables. While functional, these methods become increasingly difficult to manage as the size of the institution grows, especially when departments, electives, and resource limitations add complexity. Conflicts such as teacher overlaps, room double-booking, or incomplete schedules are common, and any modification often requires restructuring the entire timetable manually. The need for a dynamic, scalable, and automated timetable generator has become vital in modern academic institutions. With advancements in web technologies, there is an opportunity to replace outdated systems with interactive, efficient solutions. This project proposes the Automated Time Table Generator, a web-based application built using React, JavaScript, and MySQL. The system aims to simplify the creation, modification, and access of academic schedules while providing real-time visibility and reducing human error. What makes this system unique is its role-based login architecture, allowing users to interact with the timetable based on their designation. Administrators can create and manage the timetable, teachers can view their assigned periods and request changes, and students can instantly access their class schedules. It also includes features such as period change request via email, real-time class tracking, and timetable PDF export. What makes this system unique is its role-based login architecture, allowing users to interact with the timetable based on their designation. Administrators can create and manage the timetable, teachers can view their assigned periods and request changes, and students can instantly access their class schedules. It also includes features such as period change request via email, real-time class tracking, and timetable PDF export.Moreover, this platform enhances communication between stakeholders, reduces administrative load, and improves academic planning. The introduction of real-time features ensures that students and teachers are always aware of ongoing classes without the need for printed schedules. In summary, this project seeks to not only automate but also modernize how educational timetables are created and managed in institutions of all sizes.

III. LITERATURE REVIEW

Before the introduction of automated systems, timetable preparation was handled manually by academic coordinators. Typically, these timetables were drawn up using pen and paper or spreadsheet software like Microsoft Excel. While manageable for small institutions, these methods quickly become inefficient as the number of classes, faculty, and subjects increases. Manual methods often lack visibility and transparency, especially when changes need to be made. If a faculty member becomes unavailable or if classroom allocations must change, it results in a complete revision of the entire timetable—costing hours of administrative work. Moreover, manual systems do not account for resource constraints like overlapping teacher periods or double-booked classrooms. The lack of centralized access and live visibility means students and faculty are often unaware of last-minute changes, leading to confusion. Communication between stakeholders is usually informal, increasing the chance of errors. There is also no personalization—faculty cannot see just their schedules, and students don't receive custom views of their own class routines. In today's academic environment, where speed, clarity, and adaptability are crucial, manual timetable systems are clearly outdated and pose a serious bottleneck to institutional productivity.

3.2 Use of Automation

Automation in timetable generation started gaining popularity with the growth of educational ERP software and web applications. Some platforms introduced features like auto-assignment of classes and periods based on pre-entered constraints. These automated systems use algorithms to minimize conflicts and ensure optimal resource usage. Several academic papers have proposed different approaches, such as constraint satisfaction problems (CSP), greedy algorithms, or even genetic algorithms for timetable optimization. One such software is FET (Free Educational Timetabling),

which uses a constraint-based model to auto-generate schedules. However, such tools often lack user-friendly interfaces or require high technical skills to configure. Additionally, these platforms generally focus only on generating the timetable and neglect other essential features like period change requests, personalized dashboards, or PDF exports. Most tools do not offer dynamic updates or real-time visibility, and they often miss user engagement. Therefore, while automation is a step forward, many existing systems fail to deliver a complete solution. This is where our project fills the gap by combining automation with usability, personalization, and real-time communication features.

3.3 Gap in Existing Solutions

Despite advancements in automation, there is a significant gap in tools that combine smart scheduling with user accessibility and real-time management. Many ERP systems that include timetable modules are costly and bundled with unnecessary features that small or medium institutions do not require. They also lack flexibility in adapting to real-time changes, making them more suited for static planning rather than dynamic academic environments. Open-source platforms, while free, often demand technical expertise for installation and maintenance. Furthermore, most current systems lack proper role-based login, meaning that users see data not meant for them, compromising both usability and security. The inability to easily submit and track period change requests, absence of PDF download features, or lack of real-time period highlighting makes them less effective in practical academic scenarios. Our system bridges this gap by offering all these features in a lightweight, web-based tool built with React and MySQL. It is cost-effective, easy to use, and designed with the real-world needs of faculty, students, and administrators in mind. Thus, this project is not just an improvement, but a redefinition of how timetable management should work in modern institutions.

3.4 Importance of Role- Access Control (RBAC)Based

Role-Based Access Control (RBAC) ensures that users can only access features relevant to their role, improving both security and usability. In academic scheduling, RBAC restricts admin-level controls to authorized personnel, while faculty and students are granted only the access they require—such as viewing timetables or requesting period changes. Many existing systems overlook this, resulting in security risks and potential data corruption. RBAC simplifies user interaction, avoids misuse, and supports institutional discipline. It also makes the system easier to navigate by showing only necessary features. In our project, RBAC is implemented at the core, with dashboards and actions customized for each user type. This promotes a clean interface and protects the system from unauthorized edits.

IV. METHODOLOGY

4.1 Requirement Analysis

Before beginning development, a detailed requirement analysis was carried out to understand the expectations of all users involved—administrators, faculty members, and students. This phase involved surveys, interviews, and discussions with academic coordinators and teaching staff from various departments. Key challenges were identified, such as timetable clashes, lack of visibility of current class periods, time-consuming manual edits, and poor communication during schedule changes. The requirements were categorized into functional and non-functional aspects. Functionally, the system had to support timetable creation, role-based logins, period change requests, and real-time updates. Non-functional requirements included usability, performance, and compatibility across devices. Each role (admin, faculty, student) had specific needs—for example, admins needed editing privileges, faculty required request handling features, and students demanded personalized and real-time views. Security, especially in login mechanisms, was a priority to protect sensitive academic data. Once the requirements were finalized, system workflows were designed. These workflows mapped out how a user would interact with the system—from logging in, viewing the timetable, requesting changes, to downloading PDFs. This analysis laid the foundation for a scalable and user-centric design, ensuring the solution would be practical and effective for daily academic operations.

4.2 Database Design

The core of any timetable management system lies in its ability to store and retrieve scheduling data efficiently, which makes database design a crucial step. We used MySQL, a robust and widely used relational database, due to its reliability, speed, and support for structured data. The system's database includes several interconnected tables—Users, Subjects, FacultyAssignments, Classrooms, Timetable, and Requests. Each user in the system is assigned a role (admin, faculty, or student), and this role determines access rights. The Timetable table holds the schedule for all classes, linking to both teacher and subject information. Foreign keys are used to enforce relational integrity—for example, ensuring that a subject cannot be assigned to a timetable entry unless it exists in the Subjects table. Normalization techniques were applied to avoid redundancy and ensure clean data management. Indexing was added on frequently queried fields such as teacher_id and day to enhance performance. The database is structured to support multiple departments, batch years, and customizable period slots. This allows the system to scale as more data is added. The design was tested for data consistency, performance under load, and compatibility with the frontend React components, ensuring a seamless user experience.

4.3 Scheduling Algorithm

The scheduling algorithm is the brain of the timetable generator. It was custom-developed using JavaScript to ensure compatibility with our frontend framework. This algorithm is responsible for assigning subjects to available time slots in a way that avoids conflicts such as double-booking teachers or allocating multiple classes to the same room at once. The logic begins by fetching available classes, teachers, and rooms. For each subject, it checks how many periods per week are needed and loops through available time slots to find the best fit. During this process, it verifies that the selected teacher is not already assigned during that period and that the room is not in use. It also ensures that each subject is distributed fairly across the week and not overloaded on one day. If no slot is found that satisfies all conditions, the subject is flagged for manual assignment by the admin. The algorithm includes fallback logic and prioritizes core subjects in morning sessions and electives in the afternoon where possible. Future versions

may include constraint satisfaction or AI-based optimization, but the current logic is fast, effective, and performs well in institutions with medium complexity. Multiple test runs validated its accuracy and efficiency.

4.4 Real-Time Period Highlighting

To provide a more interactive and intuitive experience, the system includes a real-time period highlighting feature that shows users which class is currently ongoing. This feature uses JavaScript's Date() function to capture the current time from the user's device. The system compares this time with the predefined period slots stored in the database. If the current time falls within a certain period's start and end time, the corresponding cell in the timetable grid is automatically highlighted using a distinct color. The page uses the setInterval() function to refresh this check every 60 seconds, ensuring that users always see the most current class session. This feature is especially useful for students trying to keep up with their day or for faculty managing their classes. It eliminates the need to cross-reference printed schedules or remember period times. Additionally, this real-time highlight works across all dashboards (admin, faculty, student) to maintain consistency. On mobile devices, the same functionality is maintained using responsive design.

V. IMPLEMENTATION

5.1 Frontend (React)

The frontend of the Automated Time Table Generator was developed using the React.js library, known for its component-based architecture, speed, and flexibility. Each part of the user interface is modular and reusable—for example, the login form, timetable grid, and dashboard layout are built as separate components. This modularity helps in maintaining and scaling the system efficiently. React Router was used for client-side routing to manage different pages for admin, faculty, and student dashboards. State management was handled using React's built-in useState and useEffect hooks to track user inputs, current sessions, and system time for the real-time period highlighting feature. The timetable is displayed in a grid layout, dynamically populated based on data fetched via APIs from the backend. CSS and Bootstrap were used to ensure the UI is responsive and mobile-friendly, adapting seamlessly to different screen sizes. Conditional rendering is implemented so that users only see features relevant to their roles. For example, faculty see request buttons, students get view-only timetables, and admins get full controls. User experience (UX) testing showed high satisfaction due to fast loading times, intuitive layout, and clearly marked actions. This frontend setup ensures an engaging and functional user interface that works across all roles.

5.2 Backend (Node.js & MySQL)

The backend logic is developed using Node.js with the Express.js framework. This server handles all core business logic, such as authenticating users, processing timetable data, handling period change requests, and sending emails. The system uses RESTful APIs that communicate with the React frontend and MySQL database. Each API endpoint is protected with JWT (JSON Web Token) authentication, ensuring that only authorized users can access or modify data. For instance, when a user logs in, their credentials are verified, and a token is issued. This token is used in every API call to validate user identity and permissions. All read and write operations—from fetching the timetable to saving a new request—are routed through the server. The backend also contains logic for conflict detection during timetable generation, verifying that no teacher or room is double-booked. Sequelize ORM (Object-Relational Mapping) was used to interact with the MySQL database, simplifying complex queries and enabling fast development. Error handling was implemented to catch and respond to invalid requests, such as scheduling clashes or unauthorized access attempts. All major actions are logged for audit purposes. The backend architecture is scalable, and additional modules (like attendance tracking or AI optimization) can be added in the future.

5.3 Email & PDF Integration

Two essential features—email notifications and PDF download—were integrated into the system to improve communication and usability. For email functionality, NodeMailer, a powerful mailing library for Node.js, was used. When a faculty member submits a request for a period change, the backend generates a formatted email containing the details of the request: the original period, proposed changes, the reason for the request, and the teacher's details. This email is automatically sent to the administrator's registered email address. If the admin approves, they can make the changes in the dashboard manually, keeping the workflow efficient and semi-automated. The email system ensures better communication, especially in urgent situations when verbal communication is not feasible. For timetable downloads, the frontend integrates jsPDF, a JavaScript library that allows HTML elements to be captured and saved as a PDF file. When a user clicks "Download Timetable," the visible grid layout is exported into a neatly formatted PDF, which can be saved or printed. This feature is particularly helpful for offline access or sharing via other platforms. These two integrations significantly boost the system's practicality and enhance real-world usability by combining automation with convenience.

VI. TESTING AND RESULTS

6.1 Unit Testing

Unit testing was a crucial step to ensure that each individual module in the system worked correctly. We broke the entire application into manageable components such as the login system, timetable generator, database queries, email service, and PDF export. Each function and method was tested separately using dummy data. For example, we tested the login module by inputting both correct and incorrect credentials to verify role-based access. The scheduling algorithm was tested with various subjects, time slots, and teacher availabilities to confirm that it avoided double-booking. We also checked that email triggers fired only when requests were submitted. The PDF export feature was tested to ensure accurate layout and content across different browsers. For every unit, edge cases like missing fields, invalid inputs, and server errors were included to test how well the system handles

exceptions. These tests helped identify bugs early, such as the system allowing a class to be assigned during a teacher's existing session. Fixes were implemented, and re-testing was done. Automated testing tools like Postman and unit test scripts helped streamline the process. By testing modules independently, we confirmed their correctness and reliability before integrating them into the complete system.

6.2 Integration Testing

Once individual modules were successfully unit-tested, the next step was integration testing, which ensured that all modules worked together seamlessly. For example, the login system had to work correctly with the role-based dashboard display. We verified that a faculty user logging in would see their assigned timetable and have access to the change request form, while students only had a view-only timetable. The integration between the backend database and frontend rendering was carefully tested. When the admin added or modified a class, the updates reflected instantly in the user dashboards. We also tested the API responses using tools like Postman and ensured that invalid or unauthorized requests were correctly blocked. Another critical integration point was the email system—when a faculty request was submitted, we checked that the backend successfully captured the form data and generated a well-formatted email. The PDF download was also tested in various screen sizes and across multiple browsers. Through these integration tests, we identified a few issues such as incorrect table joins and front-end misalignment, which were promptly fixed. The overall result was a smooth, responsive system with well-coordinated modules, ensuring that all components communicated and functioned as intended.

6.3 User Feedback

After successful internal testing, the system was deployed in a controlled environment for real-world testing and feedback. A group of 10 faculty members, 5 administrators, and 25 students were asked to use the platform over a week. Each user tested features specific to their role—admins generated and edited schedules, teachers submitted period change requests, and students viewed and downloaded their timetables. The feedback collected through surveys and one-on-one interviews was largely positive. Most users found the interface intuitive and the system highly responsive. Faculty members appreciated the email-based request system, noting it reduced the need for in-person follow-ups or WhatsApp-based coordination. Students particularly liked the real-time period highlight feature, which helped them stay aware of their current and next classes throughout the day. The ability to export the schedule as a PDF was praised for its offline accessibility. Admin users found the timetable generation process fast and accurate, significantly reducing their workload. Some minor suggestions included adding a subject-wise color scheme to the timetable grid and integrating notifications for request approvals. A few mobile users reported UI glitches and screen layout issues, which were resolved in the final version with better responsive design. Teachers also recommended adding a summary view of weekly class hours to track teaching load easily. Most students requested a mobile app version for easier access on smartphones. These suggestions are being considered for future releases. Overall, the feedback confirmed that the system was user-friendly, efficient, and a major improvement over traditional scheduling methods, with high acceptance across all user roles.

VII. ANALYSIS

7.1 Efficiency

The Automated Time Table Generator significantly improves the efficiency of academic schedule management. Traditionally, timetable creation takes multiple hours to several days depending on the size of the institution. Manual methods involve a lot of back-and-forth verification, changes, and approvals. With our system, once subjects and faculty are entered into the database, a complete timetable can be generated within minutes. The algorithm ensures no conflicts, and visual validation is straightforward through the grid layout. In addition, features like role-based login and automated email requests reduce the time faculty and students spend communicating schedule changes. Even actions like generating PDFs—which would take manual formatting effort otherwise—are completed instantly. Testing showed that administrators could save up to 80% of their time previously spent on scheduling. Faculty no longer need to wait days for swap approvals, and students have immediate access to changes. Thus, the system not only reduces administrative workload but also improves decision-making speed. This high level of operational efficiency makes it an ideal candidate for deployment in both small and large educational institutions.

7.2 Limitations

While the current version of the system addresses many issues, it does have a few limitations. First, it relies on manual data entry of subjects, faculty names, and class details during setup. If any data is incorrectly entered, the timetable generation logic may produce errors or conflicts. Additionally, while the algorithm avoids scheduling overlaps, it does not yet optimize for faculty preferences, such as avoiding early morning slots or back-to-back classes unless manually configured. Another limitation is the approval workflow for period change requests—it still requires manual admin intervention to finalize the change, which delays real-time updates. The system currently does not consider classroom or lab capacity when assigning rooms. Also, although mobile-responsive, a dedicated mobile app could further improve accessibility. Lastly, while the PDF download is available, the layout may not always print neatly on smaller paper formats. These are areas identified for enhancement in future versions. Despite these issues, the system performs well under most academic conditions and delivers robust functionality for standard use cases.

7.3 User Adoption and Satisfaction

One of the most crucial indicators of a system's success is how well it is adopted by its users and the level of satisfaction it provides across different roles. After deploying the Automated Time Table Generator in a testing environment, we monitored how consistently users returned to the platform, how smoothly they adapted to its features, and whether they found it more efficient than traditional methods. Within a week, faculty members who initially relied on WhatsApp groups or handwritten schedules began using the system to check their periods and submit requests. Students quickly

adapted to the real-time period highlighting and PDF downloads, showing strong preference for digital timetables over printed ones. Admins, who usually spent 4–5 hours on manual scheduling, reported that the same task was now completed in under 30 minutes. Surveys showed over 90% satisfaction among users, especially regarding ease of use and time saved. The interface was described as "simple but powerful" by faculty, and students called the timetable "clear and convenient." Overall, the system demonstrated strong user engagement, minimal learning curve, and high acceptance, which indicates its practical usability in real-world academic settings. To further measure user satisfaction, we implemented feedback forms directly within the platform that users could fill after performing key actions such as timetable viewing, request submission, or PDF download. These forms revealed valuable insights—for example, faculty members emphasized how the centralized dashboard saved them from repeatedly ~~checking emails or contacting coordinators for schedule updates. Some faculty even mentioned that the visual clarity of the timetable helped them~~ better plan their day and reduce last-minute confusion. Students expressed appreciation for the color-coded highlight of current periods and the accuracy of the schedule, especially when classes were rescheduled. Many users noted that the system's performance was stable, with no crashes or delays during heavy usage. Adoption metrics also showed that over 80% of users logged in more than three times in the first five days, indicating high engagement. This positive adoption trend suggests that even users with limited technical knowledge were able to operate the platform without assistance. These findings reinforce the conclusion that the Automated Time Table Generator not only functions well but also delivers a superior user experience, making it ready for full deployment in academic institution.

VIII. CONCLUSION

The Automated Time Table Generator successfully transforms the traditional, time-consuming process of timetable creation into a fast, accurate, and user-friendly experience. By implementing automation, real-time features, and user-specific access, the project offers a powerful alternative to manual scheduling practices. The use of modern technologies such as React, Node.js, and MySQL ensures that the system is scalable, secure, and responsive.
The project supports administrators with scheduling tools, enables faculty to request changes conveniently, and gives students real-time access to their current and upcoming classes. Period highlighting adds an interactive element, while email notifications and PDF downloads enhance communication and usability. The testing results, both technical and from user feedback, confirm that the application meets its intended goals and performs reliably in realistic scenarios. It is particularly effective in reducing effort, improving accuracy, and increasing satisfaction among all users. In conclusion, the project lays a strong foundation for smarter academic planning and has the potential to evolve into a complete educational scheduling suite with future upgrades.

IX. FUTURE SCOPE

This system is designed with extensibility in mind, and several enhancements are already planned for future versions. One major improvement would be integrating artificial intelligence (AI) or machine learning to optimize timetable generation based on historical data, faculty preferences, and institutional rules. This would allow more intelligent scheduling and better use of time and resources. Push notifications can be introduced to alert users about schedule changes, period swaps, or announcements. Another upgrade could involve auto-approval logic for change requests based on certain criteria, reducing admin dependency. Additionally, features such as analytics dashboards can provide admins with insights into faculty workload, classroom utilization, and free slots. A cloud-based deployment could also be considered to allow remote access without local installation.
All of these enhancements will turn the current system into a fully-fledged academic operations platform, benefiting institutions aiming for digital transformation in education.

X. REFERENCES

[1].(2024). Automated Timetable Generation for Academic Institutions. AIP Conference Proceedings.
[2].(2024). OptiSchedule Algorithm for Automatic Time Table Generator. IEEE Xplore.Unknown .
[3].Author. (2024). University Schedule Generator. AIP Conference Proceedings. (2024).
[4].Comparative Analysis of Manual and Automatic Timetabling Approaches in Higher Education. IJCMSMC. [5].(2024). Optimizing the Scheduling of Teaching Activities in a ResearchGate.Faculty.
[6] Shraddha Ambhore, Pooja Walke, Rohit Ghundgrudkar, Akshay Alone, Anushree Khedkar, Automatic Timetable Generator , IJRESM | Volume-3, Issue-3, March-2020 | ISSN (Online): 2581-5792 | Published by: www.ijresm.org.
[7] Shraddha Thakare, Tejal Nikam, Prof. Mamta Patil, Automated Timetable Generation using Genetic Algorithm, International Journal of Engineering Research & Technology, IJERT | Vol. 9 Issue 07, July-2020 | ISSN: 2278-0181 Published by: www.ijert.org.
[8] Rutuja Kavade, Sohail Qureshi, Nikita Veer, Vaishnavi Ugale, Prof. Priyanka Agrawal, Smart Time Table System Using AI and ML,2023 IJCRT | Volume 11, Issue 5 May 2023 | ISSN: 2320-2882 | Published by: www.ijcrt.org.
[9] Mrs. G. Maneesha, T. Deepika, S. BhanuSri ,N. Ravi Kumar, P. Siva Nagamani, Automatic Time Table Generation Using Genetic Algorithm, JETIR Journal – Journal of Emerging Technologies and Innovative Research, July 2021 | ISSN-2349-5162 | Published by: www.jetir.org.
[10] Henry Techie-Menson, Paul Nyagorme, Design and Implementation of a Web-Based Timetable System for Higher Education Institutions, March 4, 2021 | International Journal of Educational Research and Information Science. Vol. 7, No. 1, 2021, pp. 1-13 | Published by: www.openscienceonline.com/journal/eirs.
[11] Automatic Timetable Generator (2020), Shraddha Ambhore, Pooja Walke, Rohit Ghundgrudkar, Akshay Alone, Anushree Khedkar. International Journal of Research in Engineering, Science and Management.
[12] Automatic Timetable Generator (2023), Prof. Manisha. P. Navale, Aniket Marne, Omkar Pasalkar, Atharva Pawar, Shubham Yadav. International Research Journal of Modernization in Engineering Technology and Science.
[13] Shraddha Ambhore, Pooja Walke, Rohit Ghundgrudkar, Akshay Alone, Anushree Khedkar, Automatic Timetable Generator , IJRESM | Volume-3, Issue-3, March2020 | ISSN (Online): 2581-5792 | Published by: www.ijresm.org.

[14] Shraddha Thakare, Tejal Nikam, Prof. Mamta Patil, Automated Timetable Generation using Genetic Algorithm, International Journal of Engineering Research & Technology, IJERT | Vol. 9 Issue 07, July-2020 | ISSN: 2278-0181 Published by: www.ijert.org.

[15] Rutuja Kavade, Sohail Qureshi, Nikita Veer, Vaishnavi Ugale, Prof. Priyanka Agrawal, Smart Time Table System Using AI and ML,2023 IJCRT | Volume 11, Issue 5 May 2023 | ISSN: 2320-2882 | Published by: www.ijcrt.org