विश्वं शास्त्रे प्रतिष्ठितम्।

MINI PROJECT REPORT

On

CLASSMATE TIMETABLE GENERATOR

Submitted in partial fulfilment for the award of degree

Of

Master of Computer Applications

By

ASWIN CHACKO JEEMON

(MLM24MCA-2020)

Under the Guidance of

Ms. DIVYA S.B

(HOD & Associate Professor, Dept. of Computer Applications)

DEPARTMENT OF COMPUTER APPLICATIONS

MANGALAM COLLEGE OF ENGINEERING, ETTUMANOOR

(Affiliated to APJ Abdul Kalam Technological University)

OCTOBER 2025

VISION

To become a centre of excellence in computer applications,competent in the global ecosystem with technical knowledge,innovation with a sense of social commitment.

MISSION

- To serve with state of the art education,foster advanced research and cultivate innovation in the field of computer applications.
- To prepare learners with knowledge skills and critical thinking to excel in the technological landscape and contribute positively to society.

Program Educational Objectives

- PEO I :Graduates will possess a solid foundation and in-depth understanding of computer applications and will be equipped to analyze real-world problems, design and create innovative solutions, and effectively manage and maintain these solutions in their professional careers.
- PEO II: Graduates will acquire technological advancements through continued education, lifelong learning and research, thereby making meaningful contributions to the field of computing.
- PEO III: Graduates will cultivate team spirit, leadership, communication skills, ethics, and social values, enabling them to apply their understanding of the societal impacts of computer applications effectively.

Program Specific Outcomes

- **PSO I**: Apply advanced technologies through innovations to enhance the efficiency of design development.
- **PSO II**: Apply the principles of computing to analyze, design and implement sustainable solutions for real world challenges.

MAPPING OF PO-PSO-SDG

1. MAPPING WITH PROGRAM OUTCOMES (POs):-

SL.NO	POs ADDRESSED	RELEVANCE TO PROJECT
1	PO1	Applied knowledge of computer science and engineering principles to design and implement a full-stack web application using Django, React, and MySQL for automated timetable generation.
2	PO2	Identified and analyzed challenges in manual timetable preparation and formulated an efficient algorithmic approach to eliminate scheduling conflicts.
3	PO3	Designed and developed a complete software solution that generates optimized, conflict-free academic timetables while adhering to real institutional constraints.
4	PO4	Conducted research on various scheduling techniques and algorithms such as rule-based and genetic algorithms to derive an appropriate approach for automated scheduling.
5	PO5	Utilized modern tools and technologies including React.js for frontend design, Django REST Framework for backend APIs, and MySQL for structured data management.
6	PO6	Understood the academic and administrative needs of educational institutions and provided a digital tool that benefits faculty and departments by improving efficiency and accuracy.

7	PO11	Applied project management principles to plan timelines, allocate resources, and meet milestones throughout system development.
8	PO12	Demonstrated self-learning by adopting new frameworks (React.js, Django REST API) and continuously updating technical skills to adapt to evolving industry tools and technologies.

LIST OF PROGRAM OUTCOMES (POs):

PO1 – Engineering Knowledge :Apply knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to solve complex engineering problems.

PO2 – Problem Analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 – Design/Development of Solutions: Design solutions for complex engineering problems and design systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

PO4 – Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of information to provide valid conclusions.

PO5– Modern Tool Usage : Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to complex engineering activities with an understanding of the limitations.

PO6 – The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO7 – Environment and Sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of, and need for sustainable development.

PO8 – Ethics : Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO9 – Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 – Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11– Project Management and Finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 – Lifelong Learning: Recognize the need for, and have the ability to engage in independent and life-long learning in the broadest context of technological change.

2. MAPPING WITH PROGRAM SPECIFIC OUTCOMES (PSOs):

SL.NO	PSOs ADDRESSED	RELEVANCE TO PROJECT
1	PSO 1	Implemented advanced web technologies such as Django REST Framework, React.js, and MySQL to create an innovative and efficient digital timetable generator that automates complex scheduling tasks.
2	PSO 2	Designed and implemented a real-world academic management solution that automates complex scheduling tasks, ensuring efficiency, sustainability, and accuracy.

LIST OF PROGRAM SPECIFIC OUTCOMES (PSOs):

PSO 1: Apply advanced technologies through innovations to enhance the efficiency of design development.

PSO 2: Apply the principles of computing to analyze, design and implement sustainable solutions for real world challenges.

3. MAPPING WITH SUSTAINABLE DEVELOPMENT GOALS (SDGs):

SDG NO	SDGs ADDRESSED	RELEVANCE TO PROJECT
SDG 4	Quality Education	Ensures inclusive and equitable quality education by simplifying academic planning, improving transparency, and providing digital accessibility to institutional timetables.

SDG 9	Industry, Innovation, and Infrastructure	Encourages innovation by using modern technologies to build resilient digital infrastructure for academic institutions, promoting digital transformation in education.
SDG 13	Climate Action	Supports environmental sustainability by reducing paper usage through digital timetable creation, storage, and sharing, thereby minimizing the carbon footprint of institutional operations.

SUSTAINABLE DEVLOPMENT GOALS (SDGs):

SDG 1 – No Poverty-End poverty in all its forms everywhere.

SDG 2 – Zero Hunger-End hunger, achieve food security and improved nutrition, and promote sustainable agriculture.

SDG 3 – Good Health and Well-Being-Ensure healthy lives and promote well-being for all at all ages.

SDG 4 – Quality Education-Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

SDG 5 – Gender Equality-Achieve gender equality and empower all women and girls.

SDG 6 – Clean Water and Sanitation-Ensure availability and sustainable management of water and sanitation for all.

SDG 7 – Affordable and Clean Energy-Ensure access to affordable, reliable, sustainable, and modern energy for all.

SDG 8 – Decent Work and Economic Growth-Promote sustained, inclusive, and sustainable economic growth, full and productive employment, and decent work for all.

SDG 9 – Industry, Innovation, and Infrastructure-Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG 10 – Reduced Inequality-Reduce inequality within and among countries.

SDG 11 – Sustainable Cities and Communities-Make cities and human settlements inclusive, safe, resilient, and sustainable.

SDG 12 – Responsible Consumption and Production-Ensure sustainable consumption and production patterns.

SDG 13 – Climate Action-Take urgent action to combat climate change and its impacts.

SDG 14 – Life Below Water-Conserve and sustainably use the oceans, seas, and marine resources.

SDG 15 – Life on Land -Protect, restore, and promote sustainable use of terrestrial ecosystems, manage forests sustainably, combat desertification, halt and reverse land degradation, and halt biodiversity loss.

SDG 16 – Peace, Justice, and Strong Institutions- Promote peaceful and inclusive societies, provide access to justice for all, and build effective, accountable, and inclusive institutions.

SDG 17 – Partnerships for the Goals -Strengthen the means of implementation and revitalize the global partnership for sustainable development.

MANGALAM COLLEGE OF ENGINEERING, ETTUMANOOR
DEPARTMENT OF COMPUTER APPLICATIONS
OCTOBER 2025

<u>DECLARARTION</u>

I hereby certify that the work which is being presented in the project entitled "CLASSMATE TIMETABLE GENERATOR" submitted in the ***DEPARTMENT OF COMPUTER APPLICATIONS*** *is an authentic record of my own work carried under the supervision of DIVYA S.B,* ***HOD & ASSOCIATE PROFFESSOR.*** *This study has not been submitted to any other institution or university for the award of any other degree. This report has been checked for plagiarism by the college and the similarity index is within permissible limits set by the college.*

Name & Signature of Student

Date:

Place:

MANGALAM COLLEGE OF ENGINEERING, ETTUMANOOR

DEPARTMENT OF COMPUTER APPLICATIONS

OCTOBER 2025

विश्वं शास्त्रे प्रतिष्ठितम्।

CERTIFICATE

*This is to certify that the Project titled **"CLASSMATE TIMETABLE GENERATOR"** is the bonafide record of the work done by **ASWIN CHACKO JEEMON (MLM24MCA-2020)** of MCA in Computer Applications towards the partial fulfilment of the requirement for the award of MASTER OF COMPUTER APPLICATIONS **by APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**, during the academic year 2025-26.*

Internal Examiner

Project Coordinator

Ms. Banu Sumayya s

Assistant Professor

Department Of Computer

Project Guide

Ms. Divya SB

HOD & Associate Professor

Department Of Computer Applications

Head of the Department

Ms. Divya SB

Associate Professor

Department of Computer Ap

ACKNOWLEDGEMENT

I am greatly indebted to the authorities of Mangalam College of Engineering for providing the necessary facilities to successfully complete my Project on the topic “Disaster reporting application”.

I express my sincere thanks to **Dr. Vinodh P Vijayan**, Principal, Mangalam College of Engineering for providing the facilities to complete my Project successfully.

I thank and express my solicit gratitude to **Ms. Divya S B,** HOD, Department of Computer Applications, Mangalam College of Engineering, for her invaluable help and support which helped me a lot in successfully completing this Project work.

I express my gratitude to my Internal Guide, **Ms. Divya SB**, Associate professor, Department of Computer Applications for the suggestions and encouragement which helped in the successful completion of our Project.

Furthermore, I would like to acknowledge with much appreciation the crucial role of the faculties especially Project coordinator, **Ms.Banu Sumayya S**, Department of Computer Application, Mangalam College of Engineering, who gave the permission to use all the required equipment and the necessary resources to complete the presentation & report.

Finally, I would like to express my heartfelt thanks to my parents who were very supportive both financially and mentally and for their encouragement to achieve my goal.

ASWIN CHACKO JEEMON (MLM24MCA-2020)

ABSTRACT

Timetable creation in colleges is often a complex, time-consuming, and error-prone process, particularly when dealing with multiple classes, shared faculty, and limited teaching hours. Manually preparing timetables requires extensive coordination and repeated adjustments to prevent overlapping schedules, which often leads to inefficiencies and inaccuracies. To address these challenges, ClassMate has been developed as a web-based automated timetable generation system designed specifically for academic departments such as MCA and BCA.

The proposed system enables administrators or faculty members to generate weekly timetables automatically for multiple classes without any scheduling conflicts. ClassMate uses an intelligent scheduling logic that ensures proper allocation of faculty, subjects, and classrooms while considering institutional constraints. It effectively handles lab sessions spanning continuous periods and tutorial hours thereby maintaining institutional time discipline.

Developed using React.js for the frontend, Django REST Framework for the backend, and MySQL for the database, the system provides a modern, responsive, and scalable platform. Faculty members can easily add, edit, or update subject and class details, generate timetables, and download them in printable formats such as PDF or Excel. The system's modular architecture ensures smooth communication between components, and its automated conflict detection eliminates human error during schedule creation.

By significantly reducing manual effort and improving accuracy, ClassMate enhances administrative productivity and ensures effective utilization of faculty and classroom resources. It demonstrates the potential of full-stack web development in automating academic scheduling tasks, providing educational institutions with a reliable, efficient, and user-friendly solution for smart timetable management.

Mapping with Sustainable Development Goals	**Industry, Innovation and Infrastructure**
Quality Education	Ensures inclusive and equitable quality education by simplifying academic planning, improving transparency, and providing digital accessibility to institutional timetables.

Decent Work and Economic Growth	Promotes efficiency in academic operations by automating manual processes, reducing workload, and enabling faculty to focus more on teaching and research activities, thus supporting productive and balanced work environments.
Industry, Innovation and Infrastructure	Encourages innovation by using modern web technologies such as Django, React, and MySQL to develop resilient digital infrastructure for academic institutions, promoting digital transformation in education.
Climate Action	Supports environmental sustainability by reducing paper consumption through digital timetable generation, storage, and sharing, thereby minimizing the carbon footprint of institutional operations.

TABLE OF CONTENT

LIST OF FIGURES

List of Abbreviations

ABBREVIATION	FULL FORM
DFD	Data Flow Diagram
UML	Unified Modeling Language
AI	Artificial Intelligence
CRUD	Create, Read, Update, Delete
DBMS	Database Management System
DRF	Django REST Framework
ER	Entity Relationship
GA	Genetic Algorithm
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LMS	Learning Management System
MVC	Model View Controller
ORM	Object Relational Mapping
PDF	Portable Document Format
REST	Representational State Transfer

SDG	Sustainable Development Goal
SQL	Structured Query Language
UAT	User Acceptance Testing
UI	User Interface
UX	User Experience
API	Application Programming Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
URL	Uniform Resource Locator

CHAPTER 1
INTRODUCTION

1.1Background

In every academic institution, effective timetable management plays a crucial role in ensuring smooth and uninterrupted academic operations. The process of creating a timetable, however, is one of the most challenging and time-consuming administrative tasks. It requires precise coordination between faculty availability, classroom resources, and subject distribution. As the number of departments, courses, and faculty members increases, the difficulty in manually preparing a conflict-free timetable also rises exponentially. Traditionally, department coordinators or faculty members generate timetables manually using spreadsheets or handwritten charts, which often leads to scheduling errors, clashes between faculty assignments, and inefficient resource allocation.

The increasing digitalization of educational institutions and the growing adoption of web technologies have opened the door to automated solutions that simplify academic administration. To address the limitations of manual timetable generation, the ClassMate Timetable Generator project proposes a web-based automated system that can generate conflict-free, organized, and easily editable timetables for multiple classes and semesters. The system is specifically designed for colleges offering multi-year programs such as MCA and BCA, where multiple batches and faculty members are managed simultaneously. ClassMate automates the process of scheduling classes by intelligently distributing subjects, ensuring that no faculty or class experiences overlapping sessions.

ClassMate is developed using React.js for the frontend, Django for the backend and MySQL for data storage. The application provides a centralized digital platform that allows administrators and faculty members to generate, view and modify timetables seamlessly. It supports special requirements such as continuous-period lab sessions and tutorial hours. By leveraging modern web technologies and logical scheduling algorithms, the system reduces manual workload, minimizes human error, and enhances accuracy. The platform also allows users to download or print generated timetables, making it a complete end-to-end scheduling solution.

The development of ClassMate marks a significant shift from traditional timetable preparation methods toward intelligent automation in academia. By integrating real-time data management, rule-based allocation, and a responsive user interface, the project aims to redefine how educational timetables are managed, ensuring transparency, scalability, and operational efficiency for departments and faculty alike.

1.2 Problem Statement

Timetable generation in colleges is typically performed manually, a method that becomes increasingly unsustainable as the number of courses, departments, and teaching staff grows. Manual scheduling not only consumes valuable time but also introduces several challenges, including subject overlaps, inconsistent allocation of teaching hours, and the inability to handle complex cases such as lab sessions or multi-department faculty assignments. These inefficiencies can lead to disruptions in academic schedules, resource mismanagement, and dissatisfaction among both faculty and students.

The key issue lies in the absence of an automated, rule-based system that can efficiently allocate subjects and faculty across different classes while avoiding scheduling conflicts. Additionally, most traditional methods do not offer flexibility for adjustments if one faculty member's availability changes, the entire timetable may need to be reworked manually. There is also a lack of visualization tools for faculty and administrators to easily view or modify the schedule in real time.

The ClassMate Timetable Generator system is developed to overcome these issues by introducing a fully automated, intelligent, and easy-to-use digital platform. The system considers various constraints such as faculty workload, lab requirements, and fixed breaks, ensuring that no two classes or faculty schedules overlap. With features like automatic generation, conflict detection, and timetable export, ClassMate simplifies the scheduling process, reduces the potential for human error, and allows departments to produce accurate and consistent timetables efficiently.

1.3 Motivation

The motivation for developing ClassMate arises from the common difficulties experienced by college departments during the timetable preparation process. At the start of every semester, faculty members spend several days manually arranging schedules, often facing repeated errors and the need for continuous adjustments. The existing methods are heavily dependent on human effort, with little or no assistance from technology. This repetitive and time-consuming process motivated the idea of developing an automated system that could eliminate these challenges and streamline academic scheduling.

Furthermore, the evolution of web development technologies such as React.js and Django presents an opportunity to build robust and interactive systems that can handle complex logic efficiently. The motivation behind ClassMate is not only to automate the scheduling process but also to introduce a structured, intelligent, and scalable approach to timetable management that can adapt to multiple departments and semesters.

In addition, the project is inspired by the growing movement toward digital transformation in education. Colleges are increasingly adopting online platforms for attendance, learning management, and assessment. However, the timetable, one of the most essential components of academic planning, often remains manual. ClassMate bridges this gap by bringing automation to the core of academic scheduling, thus aligning with the concept of smart campus development. The system's motivation also lies in promoting efficiency, transparency, and digital empowerment in academic operations, making timetable generation faster, simpler, and more reliable.

1.4Scope

The scope of ClassMate encompasses the design and implementation of a comprehensive, automated timetable generation system that caters to the academic scheduling needs of institutions offering multi-semester programs such as MCA and BCA. The system focuses on providing an integrated, user-friendly platform for administrators and faculty to manage all aspects of timetable creation—from defining subjects and classes to generating conflict-free schedules automatically.

ClassMate supports the following key features within its functional scope:

- Faculty and Subject Management**:** Admins can add, update, and delete faculty members, along with the subjects they handle.
- Automated Timetable Generation**:** The system intelligently assigns subjects and faculty members to time slots while preventing conflicts across classes or departments.
- Multi-Period Handling**:** Labs and tutorials that require consecutive periods are automatically grouped.
- Predefined Breaks**:** The timetable includes fixed break periods that are uniformly applied across all days.
- Download and Print Options**:** Users can export timetables for departmental use or notice board display.
- Role-Based Access**:** Different privileges are assigned to administrators and faculty members for security and clarity.

From a technical perspective, the project utilizes a React.js frontend for interactive visualization, Django REST Framework for backend logic and API creation, and MySQL for reliable data management. This stack ensures scalability, performance, and ease of integration. The system's architecture allows for future enhancements such as integration with attendance modules, faculty substitution management, and AI-based optimization for resource utilization.

In summary, the scope of ClassMate extends beyond timetable generation, it represents a step toward academic automation and digital efficiency. The project aims to assist institutions in adopting smarter solutions for everyday administrative challenges, thereby promoting better time management, reduced workload, and improved academic coordination.

CHAPTER 2

LITERATURE REVIEW

2.1 AI-Driven Timetable Generator Using Constraint Programming

[Ankit Narang, Anshul Sharma, Anmol Tyagi, Aman Sharma, Md. Shahid, Tuijin Jishu/ Journal of Propulsion Technology, 2025]

This paper explores AI-based Constraint Programming (CP) and Machine Learning techniques for timetable optimization. The system uses Decision Trees and Clustering Algorithms to analyze teacher workload and classify subjects for balanced scheduling.

Unlike purely genetic approaches, CP models encode all constraints directly into the algorithm, reducing conflicts from the start. The authors propose a feedback-based learning mechanism to improve future scheduling accuracy.

While still a theoretical framework, the paper presents a promising direction for intelligent, adaptive scheduling systems that learn from historical data to optimize future timetables.

Key Aspects:

- Constraint Programming Framework: The CP model ensures that all hard constraints are satisfied before generating solutions, reducing conflict rates.
- Workload Balancing: Machine learning models like Decision Trees help distribute faculty loads evenly, improving schedule fairness.
- Predictive Optimization: The system learns from past data to predict potential clashes, thus preventing scheduling errors before they occur.
- Adaptability: The adaptive mechanism allows the system to evolve and improve over multiple scheduling cycles

2.2 Automatic Timetable Generation Using Genetic Algorithm

[Shraddha Thakare, Tejal Nikam, and Prof. Mamta Patil, IJERT, 2020]

This paper proposes a fully automated timetable generation system using Genetic Algorithms (GA) to handle the complex nature of scheduling in educational institutions. The research recognizes that timetable generation is a non-polynomial (NP) complete problem, where finding an optimal solution through brute-force methods is computationally infeasible. The GA approach mimics natural selection to iteratively evolve a population of possible solutions until an optimal or near-optimal timetable is achieved.

The process includes chromosome encoding to represent teacher-subject assignments, a fitness function to measure conflict levels, and genetic operators such as selection, crossover, and mutation to produce improved generations. The algorithm efficiently handles constraints like faculty workload, classroom availability, and subject distribution.

The study demonstrated that GA-based systems outperform manual and heuristic methods in terms of accuracy and efficiency. However, it noted drawbacks such as sensitivity to parameter tuning (population size, mutation rate) and lack of real-time adaptability.

This paper forms a foundational reference for modern algorithmic timetable systems, showcasing the feasibility of AI-driven optimization.

Key Aspects:

- Constraint Satisfaction: The study defines timetable generation as a constraint satisfaction problem that involves fulfilling both hard constraints (e.g., no overlapping classes) and soft constraints (e.g., teacher preferences). This helps in maintaining scheduling accuracy and fairness.
- Optimization through Genetic Operators: The use of GA operators such as selection, crossover, and mutation allows continuous refinement of solutions, gradually eliminating conflicts and improving timetable quality.
- Efficiency and Accuracy**:** By replacing manual scheduling with an algorithmic approach, the system improves time efficiency and minimizes human error, ensuring consistent results across multiple scheduling cycles.

- Parameter Sensitivity: The study notes that GA performance depends on fine-tuning parameters such as mutation rate and population size, which can affect convergence speed and overall accuracy.

2.3 Automated Timetable Generation Using Machine Learning Algorithms

[Shraddha Ambhore et al., IRJESM, 2020]

This research integrates Machine Learning algorithms such as Evolutionary Algorithm, Tabu Search, and Simulated Annealing to automate the timetable generation process. The model focuses on optimizing faculty workload distribution and ensuring efficient use of classrooms and time slots. It formulates the scheduling task as a constraint satisfaction problem (CSP), where each variable (subject, teacher, room) must satisfy both hard and soft constraints.

The system generates an initial population of random timetables, evaluates their fitness based on conflict-free arrangements, and uses iterative optimization to reach a feasible solution. The authors emphasize that ML-based scheduling not only automates allocation but also learns institutional preferences over time, improving results with repeated usage.

The research further includes modules for user registration, timetable generation, and visualization. Despite its success, it lacks a web interface and real-time updates, making deployment challenging in dynamic educational settings. Nevertheless, this study strongly supports hybrid algorithmic approaches to academic scheduling.

Key Aspects:

- Hybrid Algorithmic Design: The use of multiple optimization algorithms allows the system to balance global search (Evolutionary Algorithms) and local refinement (Tabu Search), improving overall solution quality.
- Multi-Course Scheduling: The system handles several departments or semesters simultaneously, ensuring proper teacher allocation and workload management.
- Learning-Based Improvement: The inclusion of machine learning techniques enables the system to learn from past data, leading to better scheduling decisions in future runs.
- User Interaction: The project features login and visualization modules, allowing administrators to view and modify generated timetables, improving usability and control.

2.4 AI-Based Automatic Timetable Generator Using React and Firebase

[Aviraj Latpate, Nihal Sayyad, Chaitanya Bargal, Adish Sawant, J.S Choudhari, IJCRT, 2024]

This paper introduces an AI-powered web application built using ReactJS and Firebase, aiming to automate timetable creation while enhancing user interaction and data management. The system employs optimization algorithms, likely heuristic or genetic, to allocate subjects, classrooms and faculty members efficiently.

The React-based frontend provides an intuitive and responsive interface, while Firebase ensures real-time database management and cloud storage. The model addresses both hard constraints (conflicts, resource availability) and soft constraints (teacher preferences, consecutive lectures).

Testing demonstrated significant reductions in human workload and scheduling errors. The system's scalability allows simultaneous timetable generation for multiple departments. Limitations include dependency on internet access and initial complexity in database setup. Overall, this paper bridges AI optimization and modern web development, supporting the vision of fully digital academic scheduling systems.

Key Aspects:

- Modern Web Framework: ReactJS ensures a responsive, interactive interface that enhances user experience, especially for administrators managing large datasets.
- Real-Time Data Handling: Firebase enables instant data storage and retrieval, allowing immediate updates to all connected users when changes occur.
- Constraint Management: The system effectively handles hard constraints (faculty availability) and soft constraints (preferred timings) to produce balanced schedules.
- Scalability and Collaboration: Its architecture supports multi-user access and simultaneous editing, making it suitable for large institutions.

2.5 Explicit Artificial Intelligence Timetable Generator for Colleges and Universities

[Lawrence A. Farinola, Mahougnon B. M. Assogba, OJAPPS, 2025]

This research presents a robust AI-driven timetable generation system for higher education institutions. It leverages Genetic Algorithms combined with rule-based scheduling techniques to handle diverse academic structures. The system is implemented using PHP, MySQL, and JavaScript, allowing for easy deployment in web environments.

The system incorporates both hard constraints (faculty availability, room size, time limits) and soft constraints (subject sequencing, preferred time slots). An adaptive mutation technique is used to improve convergence rates. The system produces multiple timetable views faculty-wise, room-wise, and department-wise and ensures uniform resource utilization.

The study highlights significant reductions in time and manpower compared to manual scheduling. However, it notes challenges such as limited mobile integration and occasional database redundancy. The authors propose extending the system with AI-based prediction for future scheduling cycles.

Key Aspects:

- Web-Based System Design: The use of PHP and MySQL ensures accessibility and database reliability, allowing institutions to deploy the system on their intranet or web server.

- AI Integration: By combining GA with heuristic logic, the system dynamically generates optimized timetables even with complex constraints.

- Multi-View Output: The system produces timetables categorized by faculty, room, or course, improving usability for different stakeholders.

- Resource Optimization: It maximizes the utilization of resources such as classrooms and labs, ensuring balanced workload distribution among faculty.

2.6 A Study on Automatic Timetable Generator

[Parkavi A., IJIRG, 2018]

One of the earliest foundational works in this field, this paper formulates timetable generation as a constraint satisfaction problem and evaluates algorithms such as Genetic Algorithm, Simulated Annealing, and Heuristic Scheduling. It classifies constraints as "hard" (must be satisfied) and "soft" (preferable but optional).

The authors implement a system that accepts inputs like the number of subjects, teachers, and their workloads, and automatically produces an optimized timetable. The approach demonstrated high accuracy and adaptability for small to medium-sized institutions.

While the research established key algorithmic frameworks, it lacked a digital interface and database integration. Nevertheless, it remains a seminal reference, illustrating how AI techniques could transform traditional scheduling into an automated, intelligent process.

Key Aspects:

- Constraint Classification: The research categorizes constraints as “hard” (mandatory) and “soft” (optional), helping algorithms prioritize resource allocation.
- Heuristic Techniques**:** Heuristic algorithms are used to find feasible solutions quickly by making logical, rule-based scheduling decisions.
- Time Efficiency: The automation process significantly reduces manual scheduling time while maintaining flexibility for future updates.
- Practical Relevance: The system demonstrates that even simple algorithmic models can outperform manual scheduling in medium-scale institutions.

2.7 Automatic Timetable Generation Using Genetic and Heuristic Algorithms

[Daxesh Brahmbhatt, Harsh Patel, Kenil Prajapati, Jalak Gevariya, Disha George, JETIR, 2022]

This paper combines Genetic Algorithm (GA) and Heuristic Search to improve scheduling accuracy. The hybrid approach leverages the exploration ability of GA and the refinement strength of Heuristic Search, ensuring faster convergence toward conflict-free timetables.

The algorithm applies stochastic search methods and uses performance indicators to assess efficiency. Although the model provides high accuracy, it requires careful parameter adjustment and can be computationally expensive for large datasets.

The research concludes that hybrid AI models are more flexible and efficient than single-method algorithms, particularly for complex institutional scheduling.

Key Aspects:

- Hybrid Optimization: The integration of heuristic guidance into GA helps the algorithm converge faster by focusing on more promising timetable configurations.
- Efficiency and Accuracy: This hybrid approach improves performance, producing solutions that meet both institutional and individual faculty requirements.
- Scalability**:** The system can be adapted for large institutions handling numerous subjects and staff members simultaneously.
- Computational Cost: Although accurate, the algorithm requires high processing time due to iterative optimization steps.

2.8 Timely Trigger: A Smart Timetable Generator

[Dipesh Mahajan, Garima Malakar, Rishab Lath, Taniya More, Dr. Dinesh Jain, IJNRD, 2024]

The Timely Trigger project introduces a cloud-based timetable management system that incorporates AI for conflict detection and automated updates. The system is designed for administrators and faculty to access, modify, and publish timetables online.

It uses Genetic Algorithm for optimization and cloud synchronization to ensure that any modification instantly updates across all connected devices. The platform includes user authentication, push notifications, and Excel import/export capabilities.

The study demonstrates high user convenience, real-time communication, and reduction in scheduling delays. However, limitations include internet dependency and high storage demands on cloud servers. This research underscores the growing role of AI-integrated cloud applications in education technology.

Key Aspects:

- Real-Time Cloud Integration: The use of Firebase ensures that any change made by an administrator is immediately reflected across all devices.
- Notification System: The system includes push notifications that alert teachers or students to timetable modifications, improving communication efficiency.
- AI Optimization**:** The AI module analyses existing schedules to detect and resolve conflicts automatically.
- Accessibility and Usability: With features such as Excel import/export and user roles, it enhances the accessibility and control of the scheduling process.

2.9 Web-Based Timetable Generation System for Higher Education Institutions

[Henry Techie-Menson & Paul Nyagorme, IJERIS, 2021]

This research introduces a Web-Based Timetable System using the Rapid Application Development (RAD) model. The focus is on developing an interactive system that supports quick timetable creation and modification based on user feedback.

Developed with PHP, MySQL and jQuery, the system enables real-time editing, collaboration, and version tracking. It is particularly useful for multi-department institutions with overlapping faculty schedules. The RAD approach ensures faster development cycles and easy adaptability. The study highlights benefit such as reduced scheduling time and improved user participation but points out limitations like limited offline accessibility and potential database bottlenecks under heavy usage.

Key Aspects:

- RAD Methodology: RAD ensures quick development and continuous improvement through frequent user input, increasing system relevance.
- Web Technologies: Built with PHP and jQuery, the system provides an easy-to-use interface accessible through any browser.
- User Collaboration: Multiple users can edit or review timetables simultaneously, promoting teamwork among faculty and administrators.
- Limitations: The study highlights issues related to offline usage and system downtime due to internet dependency.

2.10 Heuristic and Genetic Algorithm-Based Timetable Optimization System

[Akshay Puttaswamy, H. M. Arshad Ali Khan, Chandan S. V., Parkavi A., International Journal of Science and Innovative Engineering & Technology (IJSIET), 2018]

This paper presents a comprehensive study of metaheuristic and heuristic approaches for automating timetable generation in academic institutions. The authors focus on integrating Genetic Algorithms (GA) with Simulated Annealing (SA) and Tabu Search techniques to handle complex scheduling constraints efficiently.

The system models timetable creation as a constraint satisfaction and optimization problem, where both hard constraints (faculty availability, classroom limits) and soft constraints (faculty preferences, consecutive lecture avoidance) are encoded. By combining GA's evolutionary search with local optimization heuristics, the system aims to generate near-optimal, conflict-free schedules that balance resource utilization and fairness.

Although primarily theoretical, the study highlights the importance of hybrid algorithm design in achieving scalability and adaptability for large institutions.

Key Aspects:

- Hybrid Metaheuristic Framework:

  The integration of Genetic Algorithms with Simulated Annealing and Tabu Search enables broader exploration of the solution space while refining feasible schedules through local optimization.

- Constraint Modeling:

  The paper categorizes scheduling rules into hard and soft constraints, ensuring feasibility first before optimizing for comfort and preference-based adjustments.

- Evolutionary Search Mechanism:

  Genetic operators such as crossover, mutation, and selection are used to evolve successive generations of timetable solutions until an optimal fitness score is achieved.

- Heuristic Adaptability:

  The hybrid approach allows the system to adapt to various institutional environments, accommodating changes in course offerings, faculty, and room capacities with minimal reconfiguration.

- Performance Challenge:

  While effective for smaller datasets, the heuristic model may experience computational overhead with large-scale timetabling due to the exponential growth of constraint combinations.

CHAPTER 3
PROPOSED SYSTEM

3.1 Users

Users of the ClassMate system are primarily faculty members and administrators who interact with the web application to manage academic timetables. The administrator has complete control over the system, including the ability to add faculty details, assign subjects, and generate timetables for various departments and semesters. Faculty members can view, verify, and download their schedules once they are generated. The system ensures a simple and user-friendly interface that allows both technical and non-technical users to perform operations efficiently without requiring prior technical expertise.

3.2 Login

The login module provides authentication and secure access to the system. Administrators can log in using their predefined credentials to access the management dashboard. This module is developed using Django's built-in authentication system, which includes password hashing and session management. The frontend React interface communicates with the Django backend via REST API calls to verify credentials. Unauthorized users are restricted from accessing any administrative functionalities. This ensures data confidentiality and maintains system integrity.

3.3 Faculty and Subject Management

After logging in, administrators can perform Create, Read, Update, and Delete (CRUD) operations on faculty and subject data. Each faculty record contains information such as name, subjects handled, and department association. The admin can also add subjects and map them to specific faculty members. This data acts as the foundation for timetable generation. The system validates every entry to prevent duplicate subject assignments or inconsistencies. Through a simple and interactive React interface, administrators can manage all academic information efficiently.

3.4 Timetable Generation

This module is the **core component** of the ClassMate system. Once all faculty and subjects are entered, the admin can initiate the automatic timetable generation process. The backend Django logic applies rule-based algorithms to allocate subjects and faculty across available periods and weekdays while ensuring that:

- No faculty or subject appears in two places at the same time.
- Lab sessions are allocated for continuous periods.
- Tutorial hours are appropriately assigned.

The timetable generation process guarantees balanced faculty workload distribution and non-overlapping schedules. Once generated, the timetable is automatically stored in the database and displayed on the frontend.

3.5 Editing and Regeneration

ClassMate provides flexibility to modify existing timetables. Administrators can make manual adjustments by editing or regenerating timetables as needed. This feature is useful when changes occur, such as new faculty joining, class rescheduling, or subject replacement. The updated data is automatically validated and saved in the database, ensuring consistency across all records. The regeneration process allows the system to reapply constraints while maintaining existing assignments wherever possible.

3.6 Lab Handling System

The system intelligently handles predefined breaks and lab hours. Break times are automatically integrated into the timetable and excluded from class assignments. Lab sessions, which often span three consecutive periods, are allocated as continuous blocks without overlapping with regular theory periods. The system's backend algorithm recognizes and reserves these slots to maintain accurate scheduling across all days. This ensures that academic schedules follow institutional standards and provide a realistic daily structure for both students and faculty.

3.7 Data Storage and Management

All the data in ClassMate including faculty details, subjects, classes and generated timetables is securely stored in a MySQL relational database. Django's ORM ensures structured and efficient data handling, reducing the chances of redundancy. Each table is linked through primary and foreign keys for easy retrieval and management. The system supports scalability, allowing data for multiple programs (like MCA and BCA) and semesters to be maintained simultaneously. Regular backups can be configured to ensure data reliability and protection against unexpected failures.

3.8 Security and Authentication

Security is a key focus of the ClassMate system. User credentials are protected through Django's password hashing mechanism using the PBKDF2 algorithm, ensuring secure storage and authentication. Session management and permission control restrict access based on user roles only administrators can modify or generate timetables, while faculty members can view them. All communications between the frontend and backend occur through HTTPS-based REST APIs**,** ensuring secure data transfer. Validation mechanisms prevent duplicate data entries and unauthorized access attempts, maintaining both data integrity and confidentiality.

3.9 Download and Reporting Module

Once a timetable is generated and verified, the administrator can download or print the schedule in an organized tabular format. The export feature allows saving timetables as PDF or Excel files, which can be displayed on departmental notice boards or shared digitally with faculty members. This reporting feature enhances transparency and accessibility while providing a professional presentation of academic schedules.

CHAPTER-4

METHODOLOGY

4.1 Introduction

The methodology adopted for the ClassMate Timetable Generator focuses on developing a structured, modular, and scalable web-based system that efficiently automates the academic timetable generation process. The approach ensures seamless integration between the frontend, backend, and database layers using a three-tier architecture. The system was developed following the Agile Software Development Model, which promotes iterative progress, continuous feedback, and flexibility to incorporate changes during the development lifecycle. Each iteration involved planning, development, testing, and deployment of specific modules such as faculty management, timetable generation, and reporting.

4.2 Software Development Model

The Agile Model was selected for developing ClassMate due to its adaptability and incremental approach. The model allows the project to be divided into smaller, manageable components that can be developed, tested, and improved independently. Each sprint cycle focused on completing one or more functionalities — for example, in the first sprint, the database and faculty management modules were implemented; in subsequent sprints, timetable generation and export modules were developed.

This iterative approach helped ensure that issues were detected early, user feedback was incorporated promptly, and the final system aligned closely with the institution’s real-world needs.

4.3 System Design Process

The design process of the ClassMate system was based on a three-layered architecture consisting of the presentation layer, application layer, and data layer. The frontend was developed using React.js, offering a responsive and intuitive interface for administrators and faculty. The backend, built using Django and Django REST Framework, handled all business logic, timetable generation algorithms, and database operations. The database layer, powered by MySQL, managed persistent storage for faculty, subject, and timetable data.

During the design phase, Data Flow Diagrams (DFDs), Entity Relationship Diagrams (ERDs), and Use Case Diagrams were prepared to visualize system interactions. These diagrams guided the development of modular and reusable components that ensured smooth communication between the frontend and backend via RESTful APIs.

4.4 System Workflow

The workflow of the ClassMate Timetable Generator begins with the administrator logging into the system. Once authenticated, the admin can enter faculty, subject, and class details. After the necessary data is entered, the timetable generation process can be initiated. The system's backend applies constraint-based logic to assign subjects and faculty across available slots. The generated timetable is displayed in a tabular format on the frontend, where the admin can review and download it.
If any inconsistencies are detected, the system allows editing and regeneration without affecting existing data. The overall workflow ensures accuracy, automation, and data consistency throughout the process.

4.5 Algorithm and Logic

The core algorithm used in ClassMate is a rule-based timetable allocation system that ensures no conflicts occur between faculty or subjects. The algorithm follows these key steps:

1. Retrieve all faculty, subject, and class data from the MySQL database.
2. Define constraints such as working hours, lab durations, and predefined breaks.
3. Assign subjects to periods ensuring:
    - No faculty teaches in two classes simultaneously.
    - Each subject appears the required number of times per week.
    - Lab and tutorial sessions are grouped consecutively.
4. Validate the generated timetable for conflicts.
5. Store and display the final timetable on the frontend.

Future versions of the system may integrate a Genetic Algorithm (GA) or PyGAD optimization to improve timetable efficiency and workload distribution.

4.6 Implementation

The implementation phase involved converting the system design into functional modules using modern frameworks and tools.

- **Frontend Implementation**:

The frontend was built using React.js with JSX syntax, providing a component-based architecture. The user interface includes forms for faculty and subject entry, a dynamic timetable view, and buttons for generating or downloading timetables. Axios is used for sending API requests to the Django backend.

- **Backend Implementation**:

The backend was implemented using Django and Django REST Framework. It manages faculty, subject, and timetable models and provides API endpoints for CRUD operations. Django's ORM ensures efficient interaction with the MySQL database, while built-in authentication secures access to administrative operations.

- **Database Implementation**:

MySQL serves as the backend database, storing structured data in relational tables such as Faculty, Subject, ClassRoom, and TimetableEntry. Each record is linked via primary and foreign keys, ensuring referential integrity and quick retrieval.

CHAPTER-5

SYSTEM ARCHITECTURE

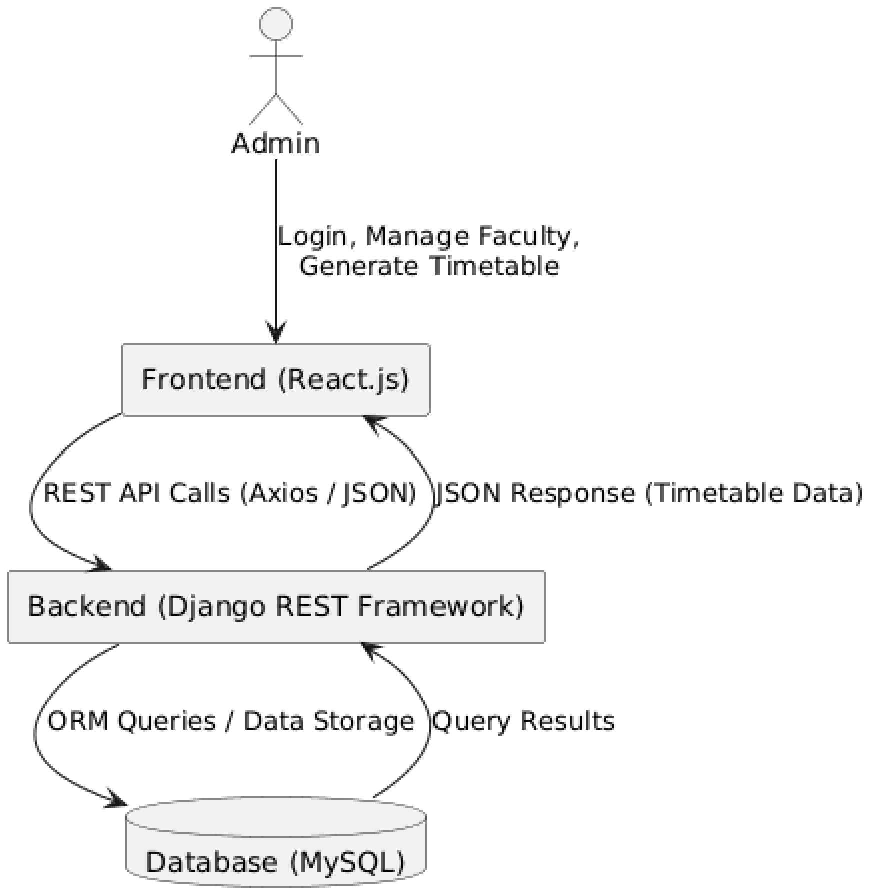

Admin
Login, Manage Faculty, Generate Timetable
Frontend (React.js)
REST API Calls (Axios / JSON)
JSON Response (Timetable Data)
Backend (Django REST Framework)
ORM Queries / Data Storage
Query Results
Database (MySQL)

Figure 5.1 System architecture

The system architecture of the ClassMate Timetable Generator defines the overall structure and interaction between its major components the frontend, backend, and database. The architecture follows a three-tier model, ensuring separation of concerns and modularity. This architecture allows independent development and maintenance of each layer, which improves scalability, performance, and flexibility.

The frontend layer manages user interactions and visualization, the backend layer handles business logic and constraint-based timetable generation, and the database layer is responsible for data storage and retrieval. Together, these components create a robust, secure, and efficient academic scheduling platform.

The ClassMate architecture is designed as a three-tier client-server system:

1. **Presentation Layer** (React.js): Handles user interface and interaction.
2. **Application Layer** (Django REST Framework): Manages business logic, API processing, and timetable generation.
3. **Data Layer** (MySQL): Stores and retrieves data such as faculty details, subject information, and generated timetables.

Each layer communicates through well-defined RESTful APIs. This modular structure simplifies debugging, enhances security, and allows parallel development across the frontend and backend teams.

5.1 Frontend Architecture (Presentation Layer)

The **frontend** of ClassMate is developed using **React.js**, which provides a dynamic and responsive user interface. React components handle tasks such as:

- Displaying faculty, subjects, and timetable data
- Managing forms for data entry and updates
- Allowing the admin to initiate timetable generation
- Rendering tabular outputs for easy readability

Data is fetched from the Django backend using **Axios** through **HTTP requests (GET, POST, PUT, DELETE)**. The frontend uses **state management** to ensure smooth data flow and instant UI updates. This layer enhances the user experience through simplicity and real-time responsiveness.

5.2 Backend Architecture (Application Layer)

The **backend** serves as the core processing engine of ClassMate. It is built using **Django** and **Django REST Framework**, which provide a reliable and scalable platform for API-based web applications.

Key responsibilities of the backend layer include:

- Handling CRUD operations for faculty, subjects, and timetable data.
- Implementing timetable generation logic based on predefined constraints.
- Validating that no subject or faculty overlaps occur.
- Managing authentication and secure data transactions.

The backend receives requests from the frontend, processes them, interacts with the database, and returns structured JSON responses. It also uses **serializer classes** for converting Django model instances into easily readable JSON data formats.

5.3 Database Architecture (Data Layer)

The **database layer** is implemented using **MySQL**, a relational database management system known for its speed and reliability. The following core tables are used in ClassMate:

- **Faculty Table:** Stores faculty information such as ID, name, and subjects handled.
- **Subject Table:** Contains subject names, codes, and mappings to respective faculty.
- **ClassRoom Table:** Maintains information about each class, semester, and department.
- **TimetableEntry Table:** Stores generated timetable entries linked to faculty, class, day, and period.

Django's **Object-Relational Mapper (ORM)** simplifies database operations by allowing developers to interact with tables using Python code instead of raw SQL queries. Foreign key relationships ensure data consistency between tables.

5.4 Data Flow in the System

This flow ensures smooth communication between all system layers, maintaining efficiency and reliability The data flow within the ClassMate system follows a structured pattern:

1. The admin interacts with the **React.js frontend** to input faculty and subject details.
2. The frontend sends API requests to the **Django backend**.
3. The backend processes the request, validates inputs, and stores the data in **MySQL**.
4. When timetable generation is initiated, the backend retrieves all relevant data, applies scheduling logic, and generates a valid timetable.
5. The generated data is returned to the frontend in JSON format, where it is displayed in a tabular view.
6. The admin can then export the timetable to a **PDF or Excel** format for record-keeping**.**

This flow ensures smooth communication between all system layers, maintaining efficiency and reliability.

5.5 System Components

The major components of the ClassMate system include:

1. **AdminPanel:**
   Provides full control over data management and timetable generation.
2. **FacultyModule:**
   Allows the storage and management of faculty and subject details.
3. **TimetableGenerator:**
   Automatically assigns subjects and faculty to time slots, avoiding clashes and including breaks/labs.
4. **DatabaseManagementModule:**
   Handles all database operations, including insertions, updates, and retrievals.
5. **DownloadandReportModule:**
   Generates printable reports and downloadable timetable files.

Each component operates independently but communicates through APIs to achieve seamless functionality.

5.6 Security Architecture

Security plays a critical role in ClassMate's architecture. Django provides built-in security mechanisms, including:

- **Hashed Passwords:** User credentials are encrypted using PBKDF2.
- **CSRF Protection:** Prevents cross-site request forgery.
- **Authentication and Authorization:** Only logged-in administrators can modify data.
- **Secure API Communication:** HTTPS and token-based authentication ensure safe data transmission.

Additionally, input validation and error handling are implemented to prevent data corruption or SQL injection attacks.

CHAPTER-6

MODULES

The ClassMate system is divided into several functional modules, each responsible for a specific operation within the application. This modular design approach ensures clarity, maintainability, and scalability of the overall system. The key modules include Admin Login**,** Faculty Management**,** Class Management**,** Subject Management**,** Timetable Generation**,** Lab Handling**,** Download and Reporting and Database Management**.**

Each module interacts seamlessly with the others through a well-defined flow, managed by the Django REST API and rendered through the React.js frontend. The modular separation also allows individual testing, debugging, and future enhancements without affecting the rest of the system.

6.1 Admin Login Module

The Admin Login module is the entry point to the ClassMate system. It ensures that only authorized personnel can access the application's functionalities. The admin logs in using secure credentials managed through Django's built-in authentication system. Passwords are encrypted using Django's PBKDF2 hashing algorithm to prevent unauthorized access. Once logged in, the admin is redirected to the main dashboard, where they can manage faculty, subjects, and classes, and initiate timetable generation.
This module also manages session handling to maintain secure and consistent user sessions during operation.

6.2 Faculty Management Module

The Faculty Management module allows the administrator to add, view, edit, and delete faculty records. Each faculty member's name, department, and subjects handled are stored in the database. This module ensures that no duplicate entries are made and that each faculty record is unique. The frontend React interface provides a simple form for adding and editing faculty details, which are sent to the Django backend through REST API requests. The backend then validates and stores the data in the MySQL database using Django's ORM.

This module forms the foundation for timetable generation since faculty information is directly linked to subject allocation and scheduling.

6.3 Class Management Module

The Class Management module manages data related to various academic programs such as MCA and BCA. It stores class names, semesters, and batch information. The administrator can create, edit, or delete class records as required. Each class is mapped to multiple subjects and faculty members. The React frontend provides a simple interface for adding or modifying class data, while the Django backend ensures that all relationships between faculty, subjects, and classes are properly maintained. This module ensures that timetables are generated accurately for each class and semester without overlaps.

6.4 Subject Management Module

The Subject Management module handles information related to all subjects offered in the institution. Each subject is linked to a specific faculty member and class. The administrator can add new subjects, assign them to faculty, or remove outdated entries. This module also supports tutorials and lab hour designations for each subject, which are used during timetable generation. By maintaining a well-organized subject database, the system ensures accurate allocation and prevents duplication of subjects across departments or classes.

6.5 Timetable Generation Module

This is the **core module** of the ClassMate system. Once all faculty, class, and subject data are entered, the administrator can generate the timetable with a single click. The Django backend executes a rule-based scheduling algorithm that allocates subjects, faculty, and labs across available slots in a conflict-free manner.

The key features of this module include:

- Ensuring no faculty or subject overlaps occur.
- Automatic handling of labs that span of each consecutive periods.
- Balanced subject distribution across weekdays.

Once generated, the timetable is stored in the MySQL database and displayed dynamically on the frontend in tabular form. The admin can review the schedule and make manual adjustments if necessary.

6.6 Lab Handling Module

The Lab Handling module automatically incorporates predefined breaks and multi-period labs into the timetable. Breaks are reserved slots that cannot be assigned to any subject. Similarly, lab sessions that require continuous hours are grouped together automatically during generation. This module ensures that institutional timing standards are strictly followed and that practical sessions are accurately represented. It improves timetable readability and eliminates the possibility of manual oversight in scheduling long-duration labs.

6.7 Download and Reporting Module

The Download and Reporting module enables administrators to export generated timetables in various formats, such as PDF or Excel. This feature allows the timetable to be printed, displayed on departmental notice boards, or shared digitally among faculty.

The module formats the timetable neatly for presentation and archiving purposes. It also supports regeneration and re-exporting if any changes are made later. This ensures that the system remains user-friendly and aligned with institutional reporting needs.

6.8 Database Management Module

The Database Management module is responsible for storing and maintaining all system data in the MySQL database. Django's ORM layer is used to perform all database operations securely and efficiently. The database contains interconnected tables for Faculty, Subjects, Classes, and TimetableEntries, ensuring relational integrity.

This module provides data reliability, quick access, and scalability. Regular database backups and validation ensure data consistency. All CRUD operations performed from the frontend are automatically synchronized with the database through REST APIs.

The modular architecture of ClassMate enables efficient development and smooth operation of the entire system. Each module performs a specific function, contributing to the automation of timetable generation and management. Together, they ensure that the system is accurate, secure, and user-friendly. The clear separation of modules also facilitates maintenance and allows easy addition of new features, such as substitution management or AI-based scheduling, in future versions of the system.

CHAPTER 7

DIAGRAMS

7.1 Entity Relationship Diagram

The entity relationship diagram of the ClassMate Timetable Generator illustrates the logical data model of the system by defining the major entities, their attributes, and interrelationships. The key entities are Faculty**,** Subject**,** Class**,** Timetable**,** and Timetable Slot. The Faculty entity includes attributes such as faculty ID, name, email, department, and designation. It is connected to the Subject entity through the Teaches relationship, signifying that each faculty member can handle one or more subjects. The Class entity stores class-specific information like class ID, name, semester, and batch, and is related to the Subject entity through the Has relationship, representing that each class can have multiple subjects. The Timetable entity is linked to the Class entity via the Generates relationship, indicating that a timetable is generated for each class. The Timetable Slot entity connects Faculty, Subject, and Timetable, representing individual periods within a schedule. Each slot has attributes such as slot ID, timetable ID, faculty ID, subject ID, period, and day. The relationships between these entities help maintain referential integrity and ensure that timetable data is efficiently managed and conflict-free.

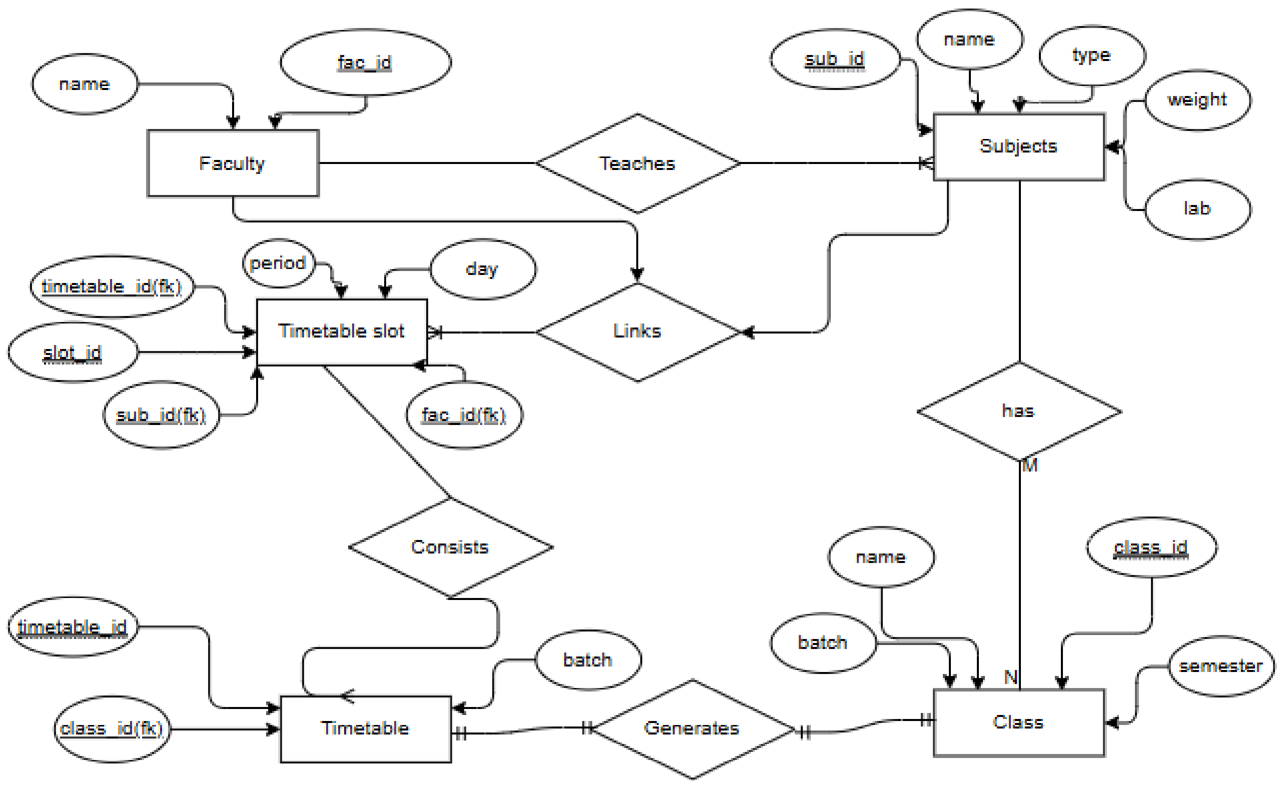


Figure 7.1 ER Diagram

7.2 Data Flow Diagrams (DFD)

A data flow diagram (DFD) is a graphical representation that shows how data moves within a system or organization. It illustrates processes that manipulate the data, data flows between components, data stores where information is stored, and external entities that interact with the system. DFDs are used to understand, analyze, and communicate information flow. They can be decomposed into different levels for a detailed view. The DFD is also called a data flow graph or bubble chart. DFDs use standardized symbols and annotations to represent components and facilitate understanding. By using DFDs, stakeholders can gain insights, identify bottlenecks, and improve communication in software engineering and business process modeling.

7.2.1 Context Level or LEVEL 0 DFD

A Level 0 DFD is also called Context Diagram. It provides a high-level overview of the system or organization, illustrating the major processes and their interconnections. It represents thetop- level view of data flow without delving into the internal workings of individual processes. The main purpose of a Level 0 DFD is to provide a conceptual understanding of how data moves through the system. It's important to note that a Level 0 DFD is often the starting point for creating more detailed DFDs. As the analysis progresses, additional levels (such as Level 1, Level 2, and so on) can be developed to further decompose the main process into sub-processes and provide a more detailed representation of the system's functionality.

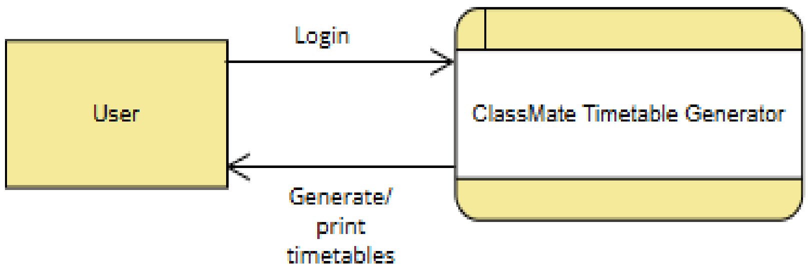


Figure 7.2 DFD Level 0

Here the Level 0 DFD represents the system as a single high-level process titled **"ClassMate Timetable Generator."**

The only external entity is the **User** (administrator or faculty).

Description:

- The user first **logs in** to the system by providing credentials.
- After successful validation, the user can **generate** or **print** timetables.
- The ClassMate Timetable Generator processes all the logic internally and provides the required output back to the user.

Data Flow Explanation:

1. The **Login** data from the user is passed to the system for validation.
2. After authentication, the user can initiate the **Generate / Print Timetables** request.
3. The generated timetable is returned as an output to the user.

Significance:
The context diagram defines the system boundary and clarifies how external entities communicate with the ClassMate application at the highest level.

7.2.2 LEVEL 1 DFD

A Level 1 DFD provides a more detailed view of the system or organization compared to the Level 0 DFD. It decomposes the processes identified in the Level 0 DFD into sub-processes, showing the data flows between them. Here, the main functions carried out by the system are highlighted as we break into its sub-processes. The purpose of a Level 1 DFD is to provide a more granular understanding of how data moves and is processed within the system. Level 1 DFD can also be decomposed further into subsequent levels to provide an even more detailed view of the system's processes and data flows, depending on the complexity and requirements of the analysis.

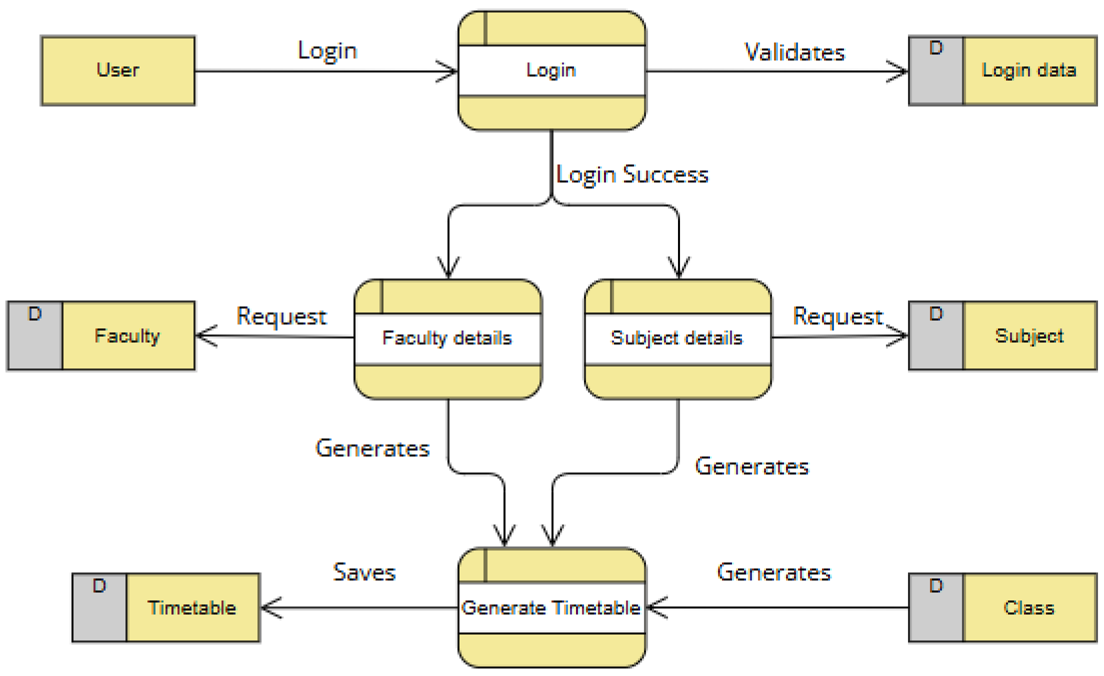


Figure 7.3 DFD Level 1

Here the Level 1 DFD expands the single process shown in the context diagram into multiple interrelated sub-processes. It details the internal working of ClassMate Timetable Generator and identifies all major data stores.

Processes:

1. **Login (1.0)**

   The user enters login credentials which are validated against stored data.

- Input: Username and Password
- Data Store Used: Login Data
- Output: Login Success / Failure Message

2. **Faculty Details (2.0)**

   After successful login, the user can request faculty information.

- Input: Request for Faculty Data

- Data Store Used*:* Faculty
- Output: List of Faculty Details

3. **Subject Details (3.0)**

   The system retrieves subject information along with assigned faculty.

- Input: Request for Subject Data
- Data Store Used*:* Subject
- Output: Subject Details Displayed

4. **Generate Timetable (4.0)**

   The system combines faculty, subject, and class information to produce the final timetable.

- Inputs: Faculty Details, Subject Details, Class Information.
- Data Stores Used: Class, Timetable.
- Output: Generated Timetable (saved and displayed to user).

7.3 Use Case Diagram

The use case diagram of the ClassMate Timetable Generator represents the various functional interactions between the system and its main actor, the Faculty (Administrator). The Faculty interacts with the system to perform key operations such as Login, Generate Timetable, Edit Timetable, Add Periods, Handle Lab Hours, and Handle Tutorial Hours. The process begins when the Faculty logs into the system through secure authentication. Once logged in, the user can generate a timetable automatically by providing inputs such as class, subject, and faculty details. The system then processes these inputs to produce an optimized schedule while preventing any Faculty Clash. Further, the Faculty can modify existing schedules using the Edit option or add new periods when additional sessions are required. The system also allows handling of specialized sessions such as labs and tutorials separately to ensure they are scheduled without overlapping with regular lectures. The Prevent Faculty Clash functionality runs throughout the process to ensure that no faculty member is assigned multiple periods at the same time. Overall, the use case diagram effectively captures the scope and functionality of the ClassMate system, demonstrating how automation and validation together streamline the complex task of academic timetable generation.

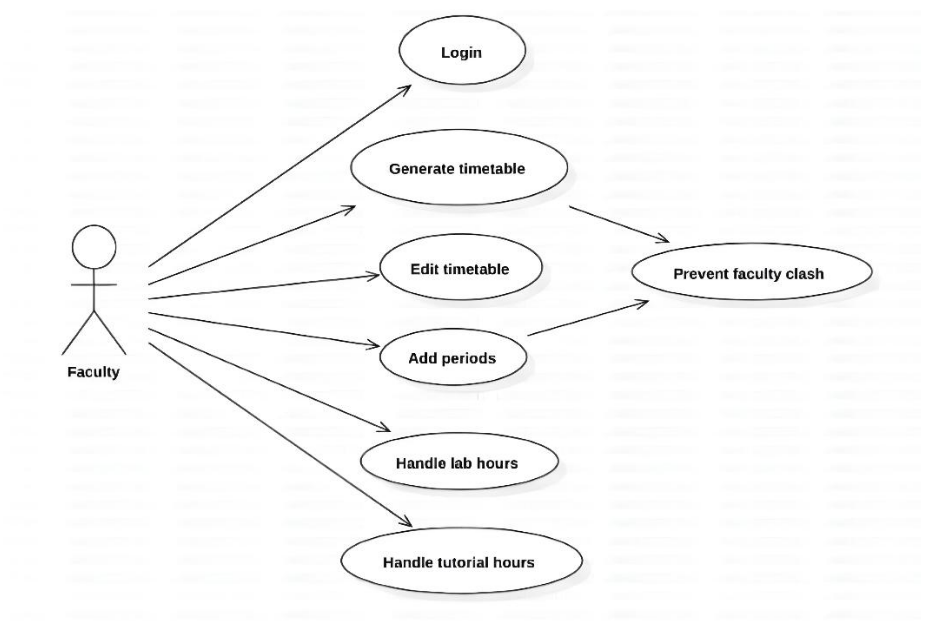


Figure 7.4 Use Case Diagram

7.4 Class Diagram

The class diagram of the ClassMate Timetable Generator illustrates the static structure of the system by showing the major classes, their attributes, methods, and relationships. The key classes include Faculty, Subject, Class, Timetable, and Timetable Slot. The Faculty class holds attributes such as faculty ID, name, email, department, and designation, and includes operations like login(), generateTimetable(), and viewTimetable(). The Subject class stores information about each course, such as subject ID, name, type, total hours, lab, and tutorial requirements. It also contains methods to retrieve details and determine whether lab or tutorial sessions are needed. The Class class represents the student group, storing data like class ID, semester, name, and batch. The Timetable class generates and manages the complete schedule, containing methods to generate(), editSlot(), addSlot(), and downloadTT(). The Timetable Slot class acts as the bridge linking Faculty, Class, and Subject by defining the actual schedule entries, with attributes for slot ID, timetable ID, faculty ID, subject ID, day, and period. Relationships between the classes show how a Faculty teaches a

Subject, a Class has Subjects, and a Timetable consists of multiple Timetable Slots. This class structure clearly outlines how the system components collaborate to generate, manage, and store timetables efficiently.

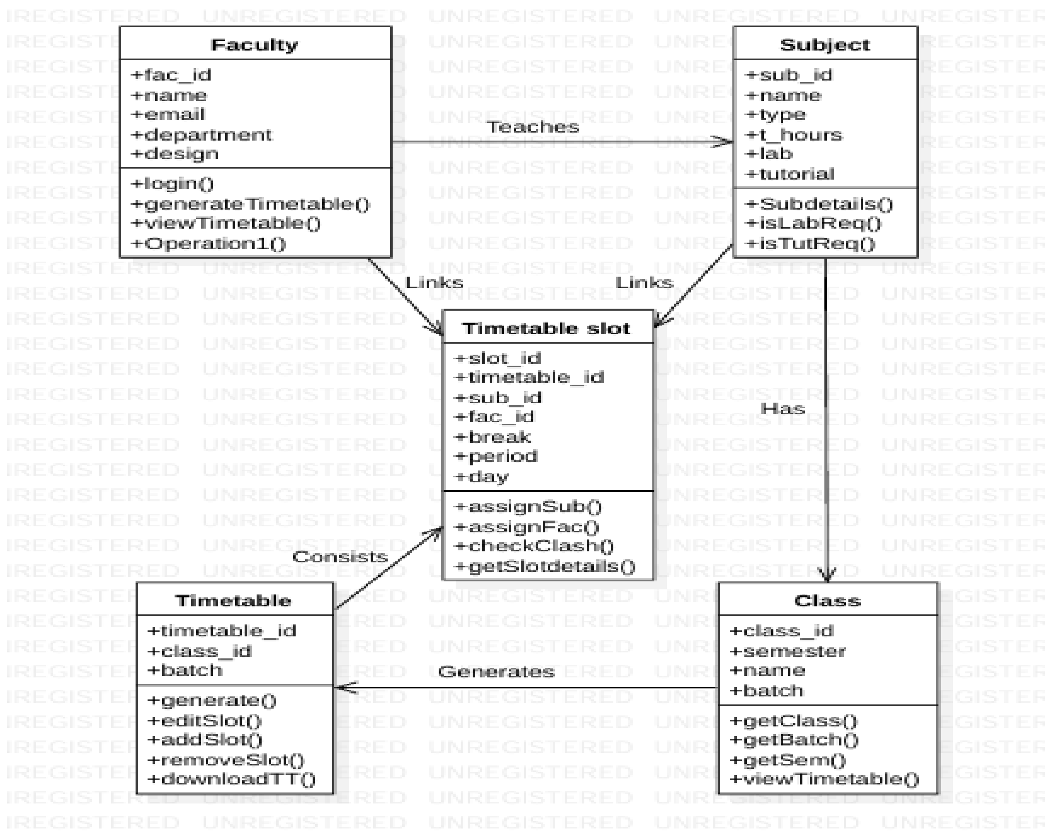


Figure 7.5 Class Diagram

7.5 State Diagram

The activity diagram illustrates the sequential flow of actions in the timetable management system, beginning with the user login. Once the user successfully logs in, the dashboard is loaded, allowing the user to input necessary details. After providing the details, the user selects preferences, which the system uses to generate a timetable. At this stage, the generated timetable undergoes validation; if the preferences are invalid, the system redirects the user back to the preference selection step, ensuring accuracy and consistency. If the timetable is valid, it is displayed to the user for review. Following this, the user has the option to download the timetable for future reference. Finally, the process concludes with the user logging out of the system. This diagram clearly depicts the logical sequence of actions and decision points within the system, highlighting how validation and feedback loops ensure that the timetable generated meets the user’s requirements.

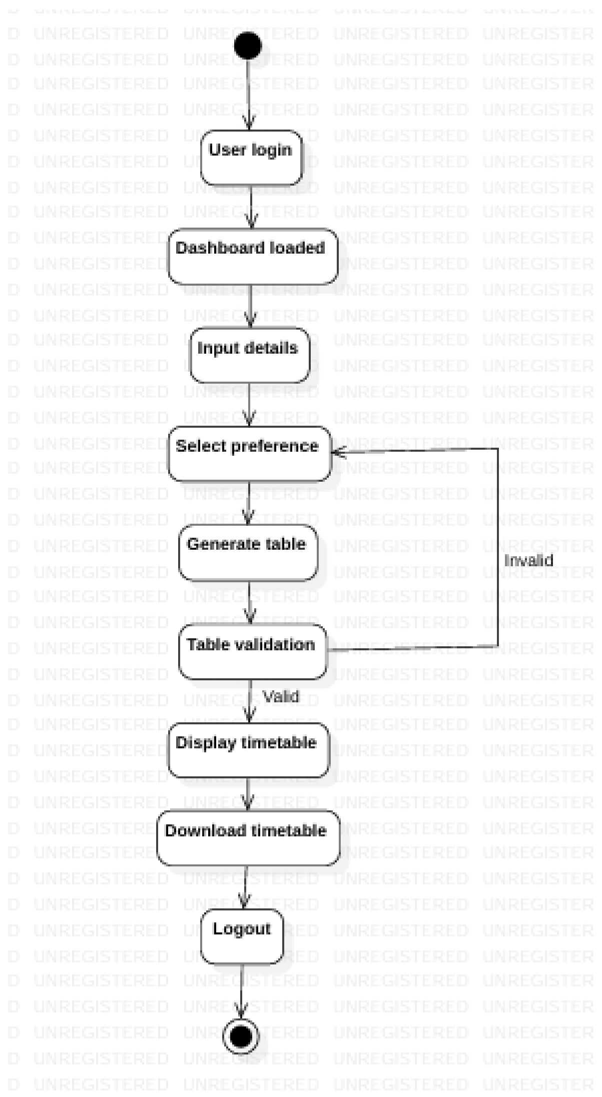


Figure 7.6 State Diagram

7.6 Sequence Diagram

The sequence diagram of the ClassMate Timetable Generator demonstrates the flow of interactions among system components during the process of timetable generation. The primary actor is the Faculty, who communicates with three main components: the User Interface (UI), the Period Allocator, and the Database. The interaction begins when the Faculty logs in and selects the required class and subject details through the UI. The UI then sends a request to the Period Allocator, which retrieves existing schedules and constraints from the Database. Using this information, the Period

Allocator applies rules to allocate periods, handle lab and tutorial constraints, and prevent faculty clashes. Once a valid timetable is generated, it is saved to the Database, and a success response is returned to the UI. Finally, the UI displays the generated timetable to the Faculty user. This diagram captures the dynamic behavior of the system by showing how control flows sequentially between different components. It highlights how the scheduler module and database interact in real time to ensure that the generated timetable adheres to all academic and faculty constraints.

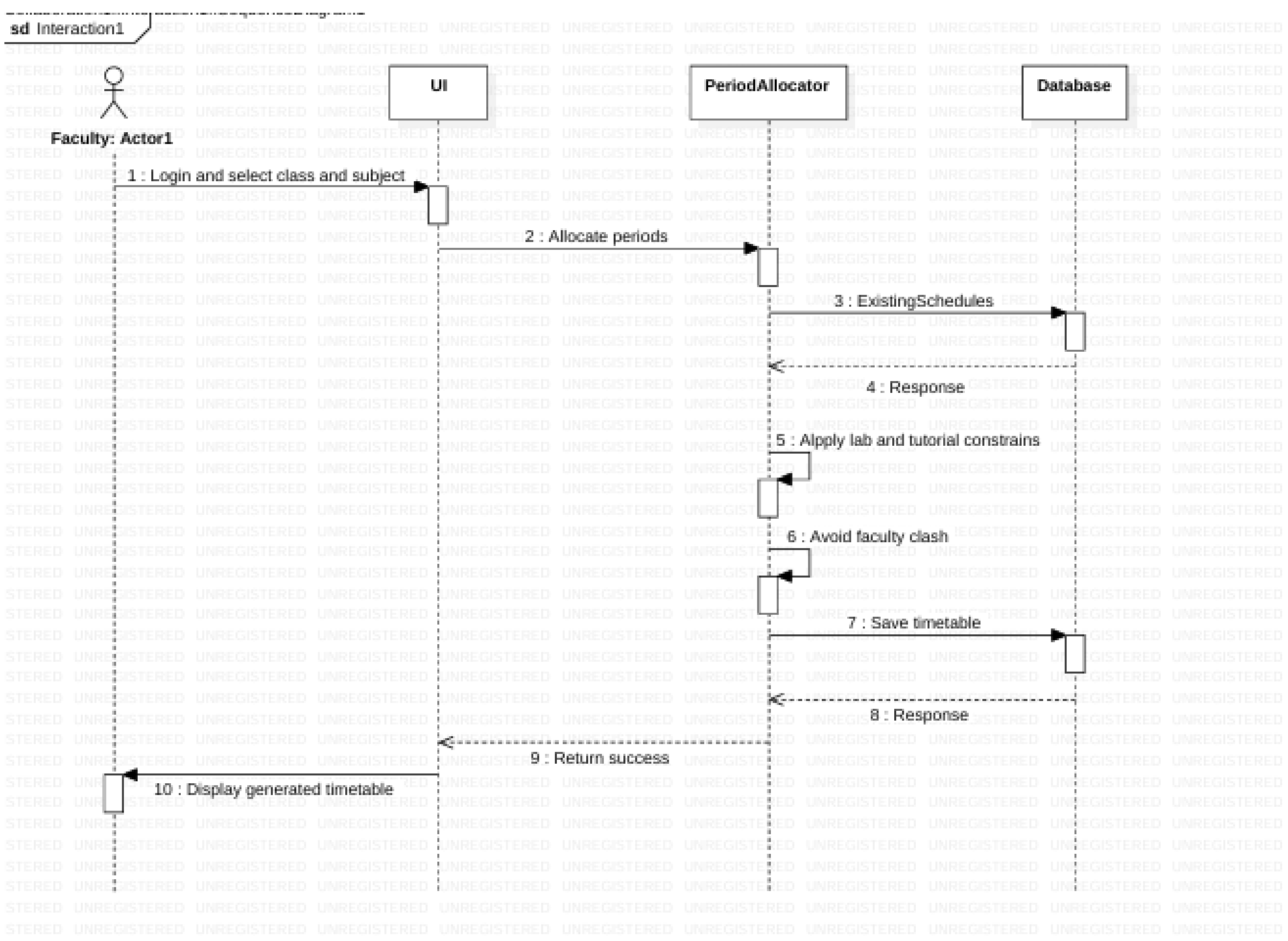


Figure 7.7 Sequence Diagram

CHAPTER 8

TESTING

Testing is one of the most critical phases in software development. It ensures that the developed system meets all functional and non-functional requirements, operates reliably, and is free from defects. The ClassMate Timetable Generator underwent multiple levels of testing to validate that each module performs as expected and integrates seamlessly with others. The main objective of testing was to detect and correct errors, verify functionality, ensure data consistency, and validate the correctness of the automatic timetable generation logic.
A systematic testing approach was followed to ensure that the system is secure, accurate, and user-friendly before deployment.

8.1 Testing Objectives

The key objectives of testing in the ClassMate project are:

1. To verify that each module (Faculty, Subject, Class, and Timetable) performs correctly.
2. To ensure that the system correctly generates non-conflicting timetables.
3. To validate data accuracy and integrity across the Django backend and MySQL database.
4. To confirm that the frontend and backend communicate effectively through REST APIs.
5. To ensure secure authentication and session handling for admin users.
6. To check the usability and responsiveness of the React.js frontend.
7. To ensure the system meets all specified requirements under normal and extreme conditions.

8.2 Testing Strategy

A bottom-up testing strategy was used, beginning with individual module testing and gradually progressing to full system testing. The testing process included Unit Testing, Integration Testing, System Testing, and User Acceptance Testing (UAT).
This structured approach ensured that errors were detected early and that integration between modules was smooth and reliable. The Django backend APIs, MySQL database queries, and React components were tested independently before being combined for system-wide validation.

8.3 Types of Testing

8.3.1 Unit Testing

Unit testing was performed to validate individual components of the system. Each function, model, and API endpoint in Django was tested to ensure correctness. For example, CRUD operations on the Faculty and Subject modules were tested to verify proper database insertion, retrieval, and updates. Django's built-in testing framework was used to automate some of these tests, ensuring that all functionalities behave as expected in isolation.

8.3.2 Integration Testing

Integration testing verified that the communication between the frontend (React.js) and backend (Django REST Framework) functioned correctly. Tools like Postman were used to send HTTP requests to the Django APIs, ensuring that data was transmitted and received correctly. The frontend components were tested using Axios to ensure the system fetched and displayed accurate data without lag or errors.

8.3.3 System Testing

System testing was conducted to verify that all components worked together as a complete, integrated system. During this phase, the entire process from faculty registration to timetable generation and download was tested in real-world scenarios. The primary focus was on verifying timetable accuracy, absence of subject/faculty clashes, proper handling of labs, and inclusion of predefined breaks.

8.3.4 User Acceptance Testing (UAT)

The final testing phase involved evaluating the system from the end-user's perspective. The admin user (representing the department coordinator) tested the ClassMate system in a simulated academic environment. Feedback was collected to confirm that the system was intuitive, accurate, and met the institutional needs. Minor UI adjustments were made based on the feedback to enhance clarity and usability.

8.4 Test Environment

Testing was performed in a controlled environment to replicate actual usage conditions.

Hardware Configuration:

- Processor: Intel Core i5
- RAM: 8 GB

- Storage: 256 GB SSD

Software Configuration:

- Operating System: Windows 10
- Frontend: React.js (Node.js environment)
- Backend: Django with REST Framework
- Database: MySQL
- Tools Used: Postman, Visual Studio Code, PyCharm, Google Chrome

The test environment was designed to closely match the real-world deployment scenario, ensuring accurate test results.

8.5 Test Cases

Test Description	**Input**	**Expected Output**	**Actual Result**	**Status**
Login validation	Correct credentials	Login successful	Successful	Pass
Login with invalid credentials	Wrong password	Access denied	Access denied	Pass
Add new faculty	Faculty name, subject	Data stored in DB	Successfully added	Pass
Timetable generation	Class & subject data	Non-conflicting timetable	Generated accurately	Pass
Download timetable	Export request	Timetable file (PDF/Excel)	File generated	Pass

8.6 Bug Identification and Fixing

During initial testing, a few minor issues were identified:

- Issue 1: Duplicate subjects were occasionally assigned to the same faculty.
  Fix: Added backend validation to prevent duplicate assignments.
- Issue 2: React UI did not refresh automatically after data edits.
  Fix: Implemented state updates using React hooks for real-time UI updates.
- Issue 3: Timetable export format misaligned during download.
  Fix: Adjusted table structure and applied consistent formatting before export.

After fixing these issues, all modules were re-tested and confirmed to be working correctly.

8.7 Test Results and Evaluation

The results of all testing phases indicated that the ClassMate system performs accurately and meets its intended objectives. The timetable generation logic produced correct, conflict-free outputs under various test datasets. The user interface was responsive and intuitive, and the system exhibited excellent performance even with large data entries.
Overall, the testing process validated that ClassMate is a robust and efficient solution for automated timetable management.

The testing phase successfully demonstrated the stability, accuracy, and usability of the ClassMate system. Each component was thoroughly tested to ensure smooth interaction between the frontend, backend, and database. With all major functionalities verified and validated, the system is ready for deployment in a real academic environment. The comprehensive testing process also ensures that the system can be easily maintained and scaled in future versions.

CHAPTER 9

ADVANTAGES AND DISADVANTAGES

Every software system, regardless of its complexity or sophistication, possesses both advantages and limitations. The effectiveness of a system is measured by the degree to which it fulfills its intended objectives while minimizing its shortcomings. The ClassMate Timetable Generator was designed with the primary aim of automating the academic timetable creation process, thereby eliminating the manual workload faced by administrators.

This chapter highlights the significant advantages that ClassMate provides to educational institutions, followed by a discussion of its few limitations that can be addressed in future enhancements.

9.1 Advantages

The ClassMate system offers several key benefits that improve efficiency, accuracy, and usability in academic scheduling. Some of the notable advantages are discussed below:

9.1.1 Automation of Manual Work

ClassMate automates the tedious process of timetable creation, which is traditionally done manually using spreadsheets or paper charts. This automation drastically reduces the time required to generate schedules and minimizes human error.

9.1.2 Elimination of Conflicts

The system ensures that no two classes or faculty members are assigned overlapping time slots. Its rule-based scheduling logic automatically validates all constraints, producing a fully conflict-free timetable.

9.1.3 Multi-Period Lab and Tutorial Handling

One of the unique features of ClassMate is its ability to handle labs and tutorials that span multiple consecutive periods. This ensures that longer sessions are scheduled correctly without breaking continuity.

9.1.4 Predefined Break Integration

The system automatically integrates fixed institutional breaks such as 10:50–11:00 AM, 12:45–01:30 PM, and 03:10–03:15 PM into the timetable. This eliminates the need for manual adjustments and ensures uniformity across all schedules.

9.1.5 User-Friendly Interface

Developed using React.js, the frontend interface provides an intuitive and responsive user experience. Even users with minimal technical knowledge can easily navigate through the system, manage data, and generate timetables with a few clicks.

9.1.6 Efficient Data Management

All data — including faculty details, subjects, and timetable entries — is securely stored in a centralized MySQL database. The use of Django ORM ensures data consistency, fast retrieval, and easy scalability for future expansions.

9.1.7 Download and Reporting Feature

Administrators can export the generated timetables as PDF or Excel files for official records or departmental notice boards. This makes data sharing and archival more efficient and professional.

9.1.8 Scalability and Flexibility

The modular design and use of web technologies make ClassMate highly scalable. New features, such as AI-based optimization or multi-department synchronization, can be integrated in the future without altering the system's core structure.

9.1.9 Reduced Human Error

Manual timetable creation is prone to mistakes such as double allocations or missed slots. ClassMate's automated algorithm ensures accuracy in scheduling, thereby reducing the risk of errors

9.1.10 Accessibility

Being a web-based system, ClassMate can be accessed from any device with an internet connection. This allows the admin to manage or regenerate timetables remotely, promoting flexibility and convenience.

9.2 Disadvantages

Despite its numerous benefits, ClassMate also has a few limitations that can be improved upon in future versions:

9.2.1 Requires Initial Data Setup

The system relies on accurate input data for faculty, subjects, and classes. Setting up this data for the first time can be time-consuming and requires attention to detail.

9.2.2 Internet Dependency

As ClassMate is a web-based application, it requires a stable internet connection for operation. Network issues may temporarily affect performance or accessibility.

9.2.3 Limited User Roles

The current version supports only a single user role — the administrator. While this simplifies control, it restricts the participation of faculty or department heads who might need access to generate or review timetables.

9.2.4 Limited Optimization Algorithm

The timetable generation logic currently uses rule-based scheduling. Although accurate, it does not include advanced optimization features like **Genetic Algorithms** or **AI-based resource balancing**, which could further enhance efficiency.

9.2.5 No Mobile Application Support

At present, ClassMate is optimized for web browsers only. A dedicated mobile app could improve accessibility and convenience for administrators on the go.

The ClassMate Timetable Generator provides a modern, automated, and reliable approach to academic scheduling. It simplifies the work of administrators, ensures conflict-free timetables, and saves valuable time during each semester's planning phase. While the system performs efficiently in its current version, future enhancements such as multi-user access, mobile integration, and AI optimization can make it even more powerful.

Overall, the advantages of ClassMate far outweigh its limitations, making it a highly beneficial tool for modern educational institutions aiming for digital transformation.

CHAPTER 10

RESULT

This chapter contains the output results and snapshots obtained from the implementation of the project **“ClassMate Timetable Generator.”**

The records presented in this section demonstrate how the proposed system performs its functions, starting from user authentication to automated timetable generation and storage. Each screen shows a distinct part of the working model and validates the successful development of the application.

The **ClassMate System** was developed with a user-friendly interface using modern web technologies. The interface ensures easy navigation, quick data handling, and clear visualization of generated timetables. The following subsections describe the major records and outputs captured from the live execution of the system.

10.1 Login Module

The **Login Module** is the entry point of the application. It authenticates the credentials of the user before allowing access to the dashboard.

Administrators or authorized faculty members enter their username and password. The credentials are verified against the data stored in the login database.

Upon successful verification, users are redirected to the system dashboard; otherwise, an appropriate error message is displayed.

The interface is designed for simplicity and ease of use, incorporating input validation and secure session handling.

Key Functions:

- Validates user credentials.
- Restricts unauthorized access.
- Redirects verified users to the dashboard.
- Provides error messages for incorrect input.

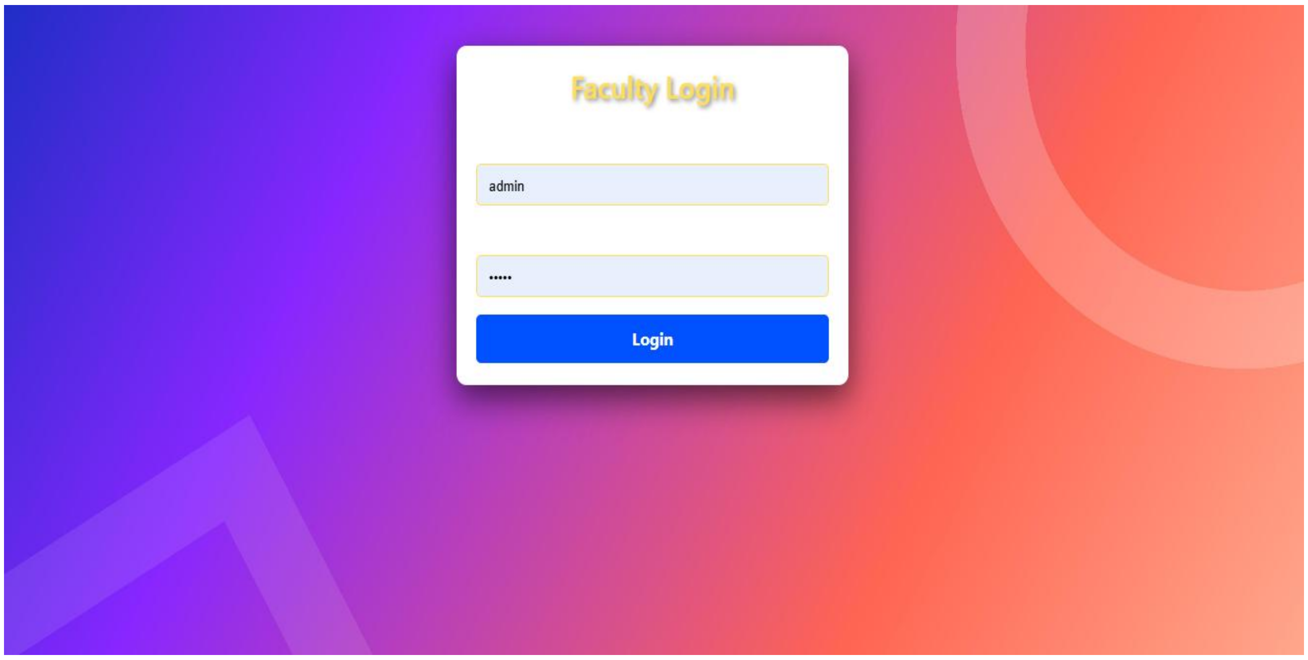


Figure 10.1 Login Page

10.2 Dashboard – Faculty and Subject Management

Once the user logs in, the Dashboard Module provides access to the faculty and subject management system.
Here, the administrator can add new faculty details, register subjects, and link each subject with the respective faculty and class.

The dashboard enables editing and deletion of records, providing complete control over institutional data.

The dashboard is divided into two sections:

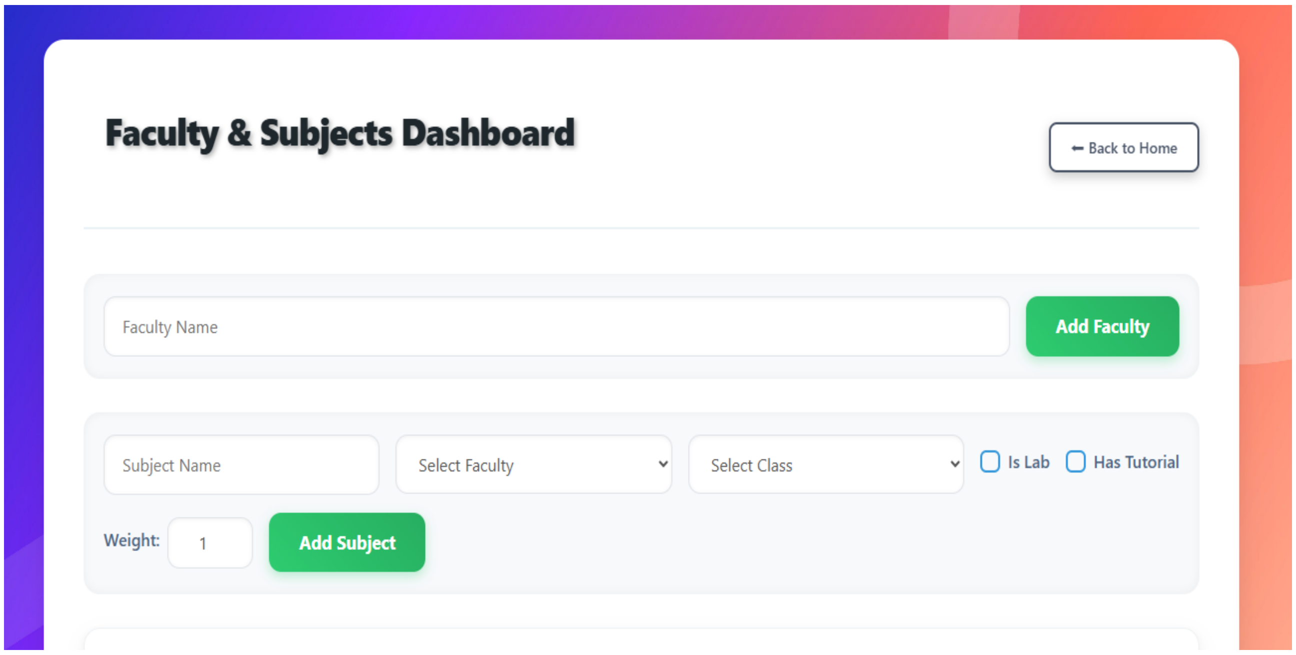
1. Faculty & Subjects Form: For entering and updating data.
2. Faculty Listing Table: Displays existing faculty–subject records.

The system automatically validates entries and ensures that no duplicate faculty or subject is created.

Key Functions:

- Add, edit, or delete faculty records.
- Assign subjects and workloads to faculty.
- Display all stored faculty data in a tabular view.

- Maintain database consistency with automatic validation.



Faculty & Subjects Dashboard

Faculty Name

Subject Name

Select Faculty

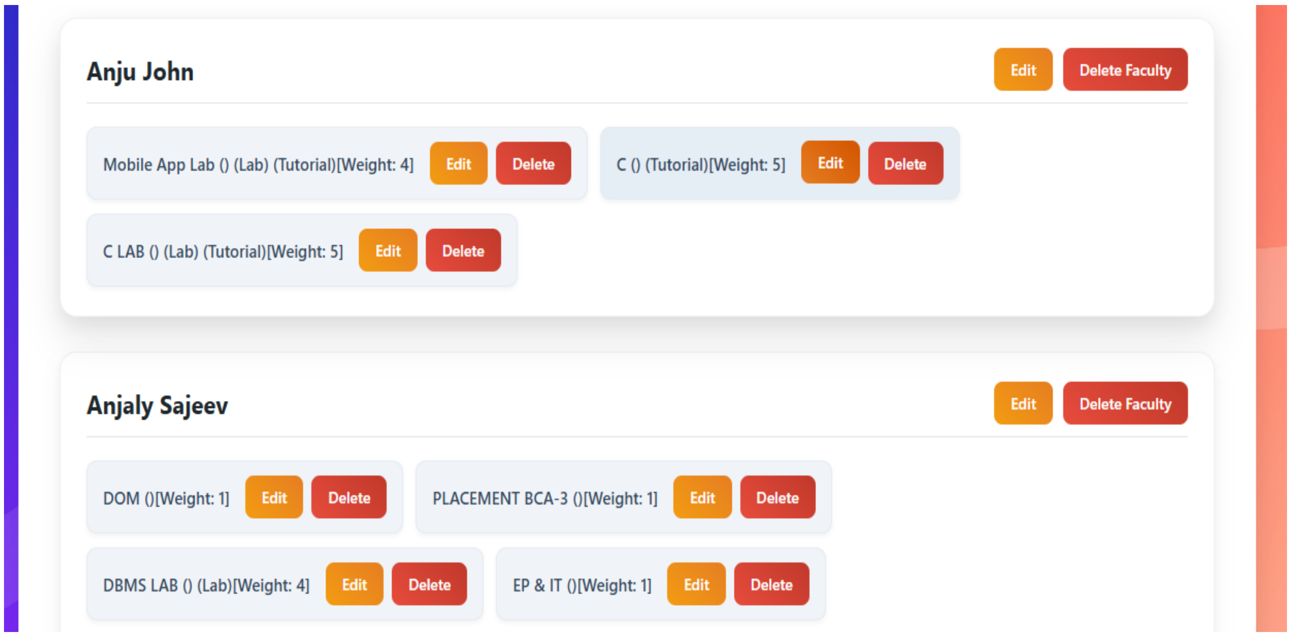
Select Class

Is Lab

Has Tutorial

Weight: 1

Figure 10.2 Faculty and Subjects Dashboard-1



Anju John

Mobile App Lab () (Lab) (Tutorial)[Weight: 4]

C () (Tutorial)[Weight: 5]

C LAB () (Lab) (Tutorial)[Weight: 5]

Anjaly Sajeev

DOM ()[Weight: 1]

PLACEMENT BCA-3 ()[Weight: 1]

DBMS LAB () (Lab)[Weight: 4]

EP & IT ()[Weight: 1]

Figure 10.3 Faculty and Subjects Dashboard-2

10.3 Timetable Generation Module

The **Timetable Generation Module** is the core component of the ClassMate system. It automatically generates conflict-free timetables using the stored faculty and subject data, considering the number of classes, hours per week, and subject types (lecture/lab/tutorial).

Users can select the desired class and semester, specify tutorial or lab requirements, and then initiate the generation process.

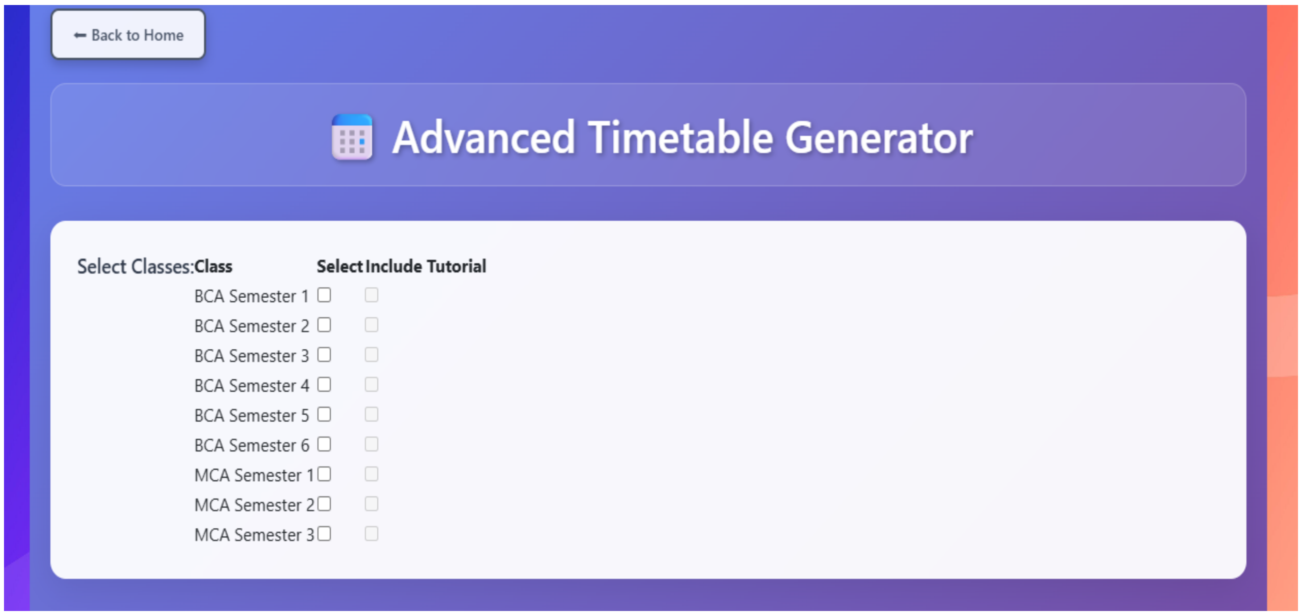
Behind the interface, the system applies scheduling logic to ensure that:

- No two subjects overlap in the same time slot.
- Faculty are not assigned multiple classes simultaneously.
- Lab sessions are properly grouped in extended time slots.

The process concludes by storing the generated timetable in the database for viewing, printing, or exporting.

Key Functions:

- Retrieve all relevant faculty and subject data.
- Apply scheduling algorithm to allocate slots.
- Prevent overlapping sessions and conflicts.
- Store generated timetable records automatically.



Advanced Timetable Generator

Select Classes:

Class	Select	Include Tutorial
BCA Semester 1	☐	☐
BCA Semester 2	☐	☐
BCA Semester 3	☐	☐
BCA Semester 4	☐	☐
BCA Semester 5	☐	☐
BCA Semester 6	☐	☐
MCA Semester 1	☐	☐
MCA Semester 2	☐	☐
MCA Semester 3	☐	☐

Figure 10.4 Timetable Generation Screen

10.4 Generated Timetable Record

After processing, the system displays the **Generated Timetable** in a well-structured tabular format. Each timetable lists subjects, faculty names, and corresponding time slots arranged across days of the week.

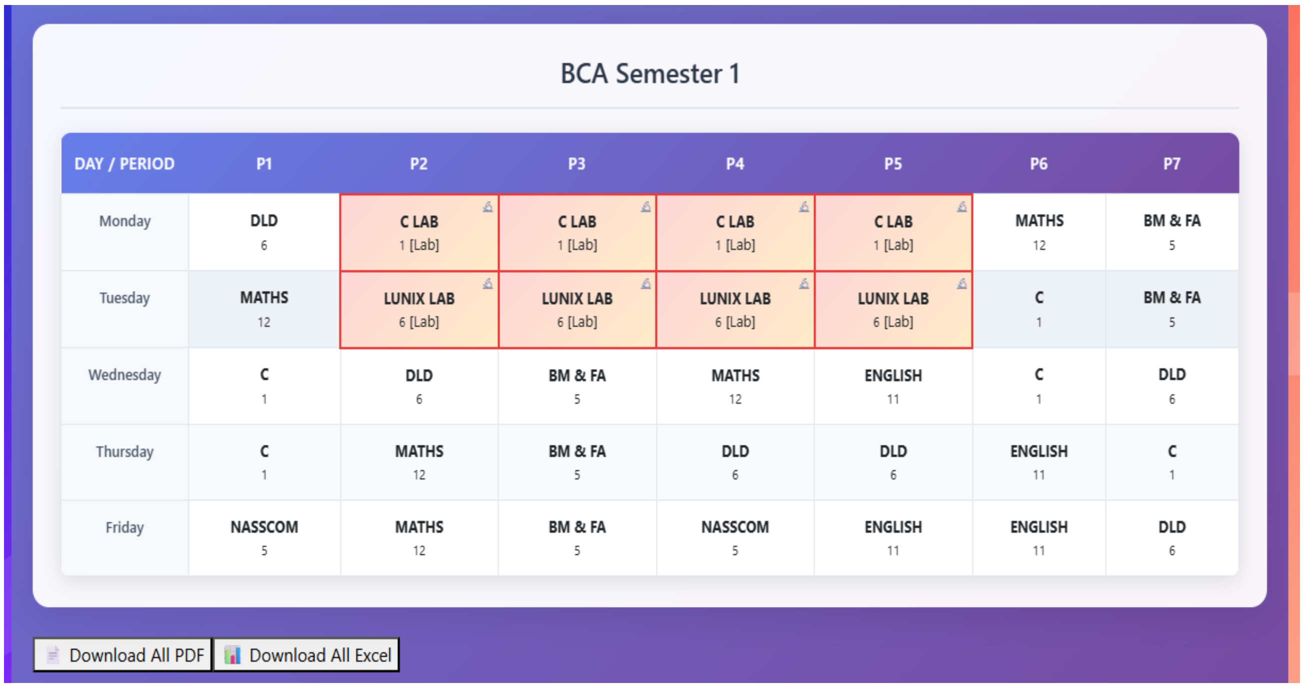
Practical and tutorial sessions are highlighted distinctly for better clarity.

Users can preview, print, or export the timetable in various formats (PDF / Excel).

The generated timetable is also stored in the database to ensure future retrieval and modification if required.

Key Functions:

- Displays final timetable in a structured format.
- Allows export and print options.
- Differentiates lectures, labs, and tutorials.
- Saves timetable data for future reference.



BCA Semester 1

DAY / PERIOD	P1	P2	P3	P4	P5	P6	P7
Monday	DLD 6	C LAB 1 [Lab]	C LAB 1 [Lab]	C LAB 1 [Lab]	C LAB 1 [Lab]	MATHS 12	BM & FA 5
Tuesday	MATHS 12	LUNIX LAB 6 [Lab]	LUNIX LAB 6 [Lab]	LUNIX LAB 6 [Lab]	LUNIX LAB 6 [Lab]	C 1	BM & FA 5
Wednesday	C 1	DLD 6	BM & FA 5	MATHS 12	ENGLISH 11	C 1	DLD 6
Thursday	C 1	MATHS 12	BM & FA 5	DLD 6	DLD 6	ENGLISH 11	C 1
Friday	NASSCOM 5	MATHS 12	BM & FA 5	NASSCOM 5	ENGLISH 11	ENGLISH 11	DLD 6

Figure 10.5 Generated Timetable Record

CHAPTER 11

CONCLUSION

The ClassMate system successfully fulfills the goal of automating academic timetable creation by integrating intelligent scheduling logic with a user-friendly web interface. The system reduces the time and effort required by administrators to manually prepare schedules and eliminates conflicts such as overlapping subjects, double-booked faculty members, or missing slots.

Through this project, the traditional manual process of timetable generation has been transformed into a streamlined, digital process that can be completed within seconds.

The system's backend, developed using Django REST Framework, serves as the core logic engine. It processes input data related to faculty, subjects, and class structures, and generates optimized timetables based on defined constraints. The MySQL database ensures structured and secure storage of all institutional data, maintaining relationships between tables such as Faculty, Subjects, ClassRoom, and TimetableEntry. The React.js frontend offers an interactive and dynamic user experience, allowing the admin to easily navigate through modules, manage records, and visualize generated timetables.

During testing and evaluation, ClassMate demonstrated exceptional reliability and efficiency. The timetables generated were conflict-free, well-structured, and adhered to institutional policies regarding breaks and lab periods. The performance evaluation showed that what previously required several hours of manual effort could now be completed in a few seconds, significantly improving productivity.

Another notable achievement of the project is its modular architecture, which makes it easy to extend or modify the system. Each component — login, faculty management, subject management, class management, and timetable generation — operates independently, ensuring scalability and maintainability.

In conclusion, ClassMate stands as a robust, efficient, and user-oriented system that fulfills all major objectives of digital timetable automation. It proves the practical application of full-stack web technologies in academic administration and demonstrates how modern software solutions can effectively replace outdated manual procedures. The system provides an essential foundation for institutions to transition toward a fully digital scheduling environment.

Future Scope

Although the current version of ClassMate efficiently automates the timetable generation process and meets all the essential objectives, there is considerable scope for enhancement and expansion in future versions. As technology evolves and academic processes become increasingly digital, the system can be extended with additional features to improve performance, accessibility, and usability.

One of the most significant improvements envisioned for the future is the inclusion of multi-user access control. At present, the system allows only an administrative user to manage and generate timetables. Future versions can incorporate different user roles such as Faculty, Department Head, and Coordinator, each having their own access privileges. This would make the system more collaborative and better suited for larger institutions where multiple users are involved in academic planning and supervision.

Another important enhancement would be the integration of Artificial Intelligence (AI) and Genetic Algorithms (GA) into the timetable generation logic. While the current rule-based algorithm ensures accuracy and conflict-free scheduling, it does not perform advanced optimization. With AI-driven algorithms, the system could analyse faculty workloads, classroom availability, subject weightage, and historical data to generate the most efficient and balanced schedules. Such intelligent automation would significantly enhance system performance and adaptability.

The development of a mobile application for Android and iOS platforms is also a promising area for expansion. A dedicated mobile app would allow users to access timetables, receive updates, and perform minor edits on the go. This feature would improve convenience and accessibility, ensuring that administrators can manage schedules anytime and anywhere.

Additionally, notification and communication mechanisms can be integrated into the system to inform faculty members whenever a timetable is updated or modified. Automatic email or SMS alerts can ensure real-time communication and minimize confusion or delays caused by schedule changes. Similarly, the system could be enhanced to integrate with institutional ERP or Learning Management Systems (LMS) to synchronize data such as faculty assignments, subject details, and classroom schedules across platforms.

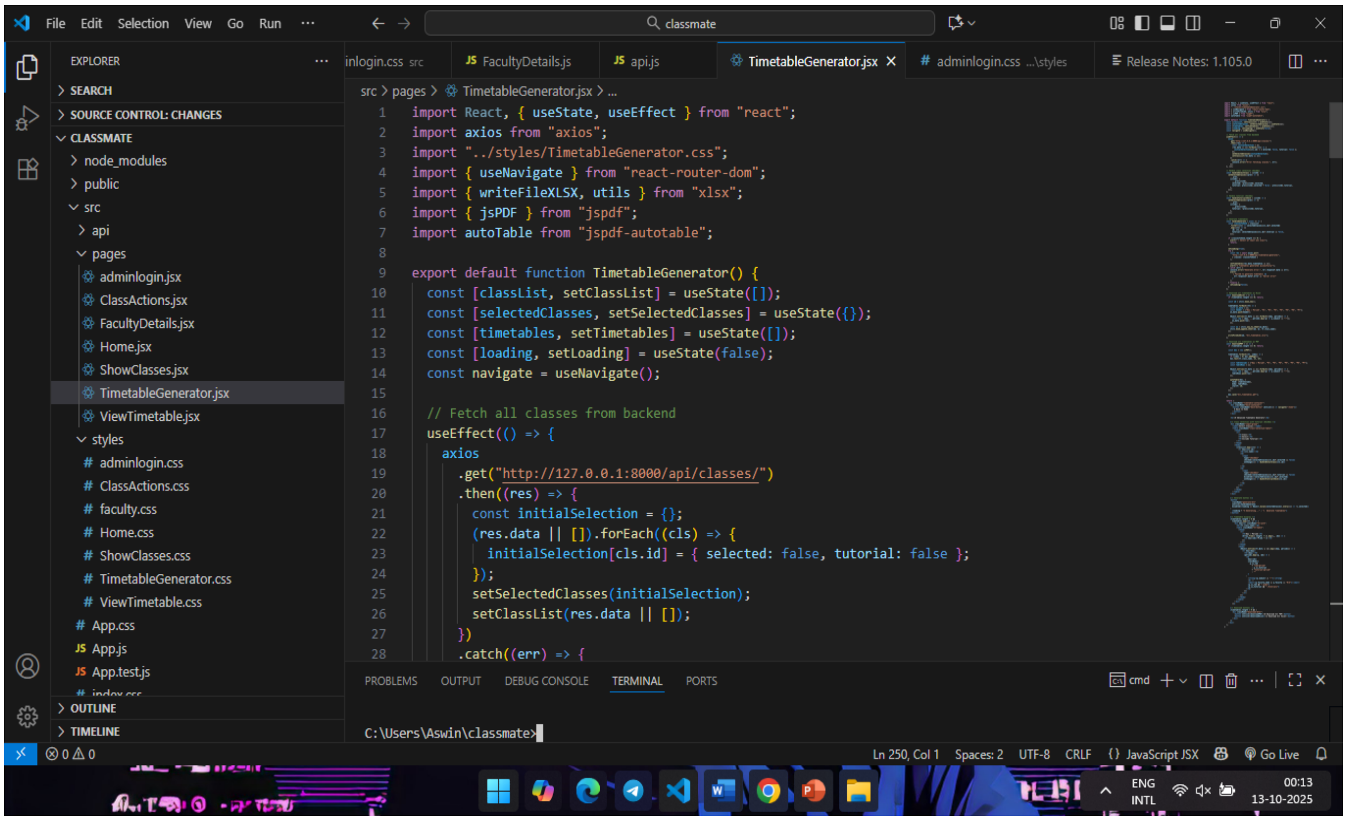
Deploying the system on a cloud infrastructure such as AWS, Azure, or Google Cloud could significantly improve scalability and availability. Cloud deployment would allow multiple departments to use the application concurrently, support online backups, and provide secure, centralized access to data from any location. Alongside this, future iterations could implement automated database backup and encryption mechanisms to safeguard academic data and prevent accidental loss or unauthorized access.

Furthermore, data analytics and visualization features can be added to generate reports on faculty workload, class utilization, and subject distribution. Such analytical dashboards would assist academic coordinators in making informed decisions about resource allocation and curriculum design. Another potential enhancement includes the introduction of faculty leave and substitution management, allowing the system to automatically reassign teachers and update the timetable in real time when a faculty member is unavailable.

Lastly, future versions of ClassMate can be expanded to support multi-department and multi-campus environments, enabling centralized academic scheduling across various programs like MCA, BCA, MBA, and Engineering streams. This would make ClassMate a comprehensive academic scheduling platform for institutions with complex structures.

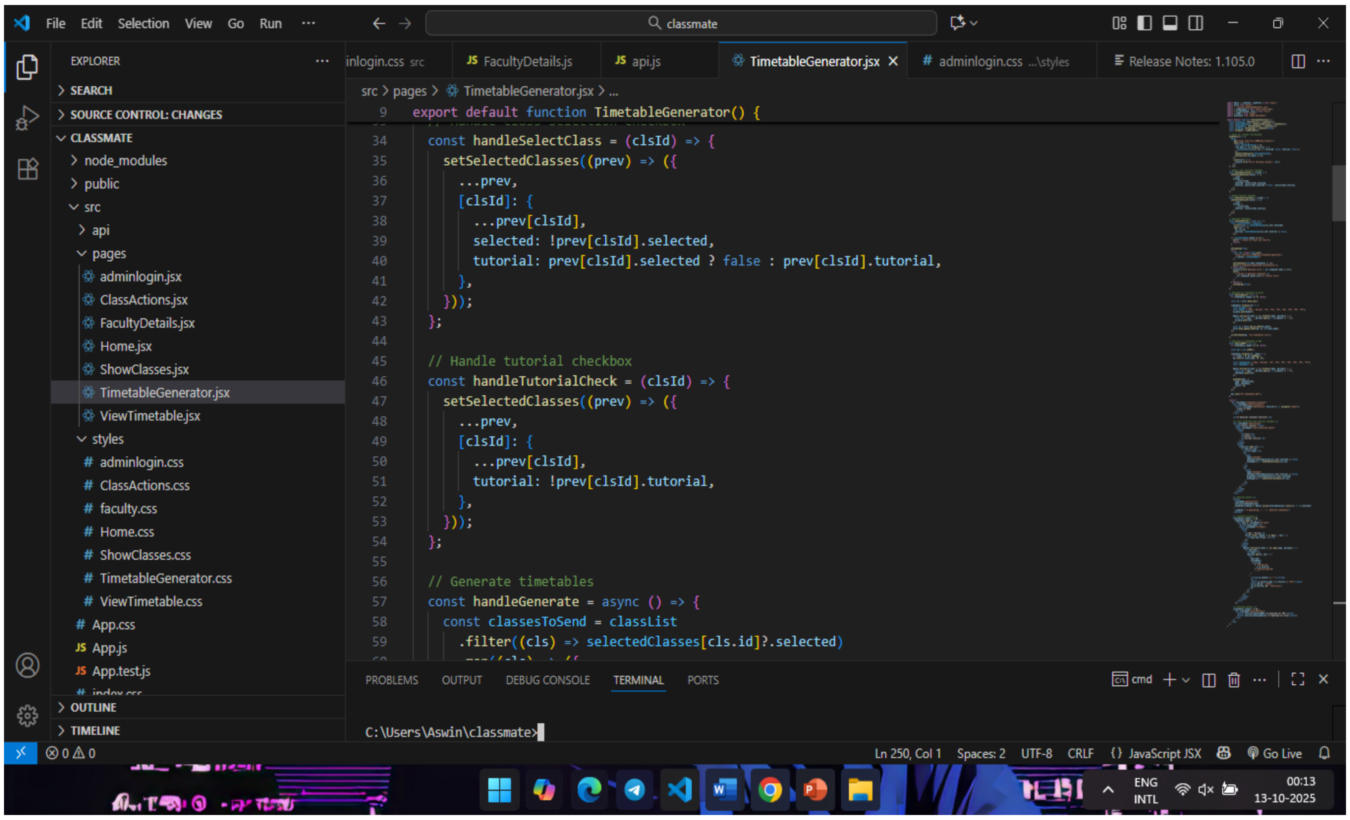
In conclusion, the future scope of ClassMate is extensive. With the integration of advanced technologies such as AI-based optimization, multi-user access, mobile support, and cloud computing, the system can evolve from a simple timetable generator into a complete academic scheduling and management solution. These enhancements will not only improve system efficiency and reliability but also align with the broader goal of creating smart, automated, and data-driven educational environments.

APPENDICES



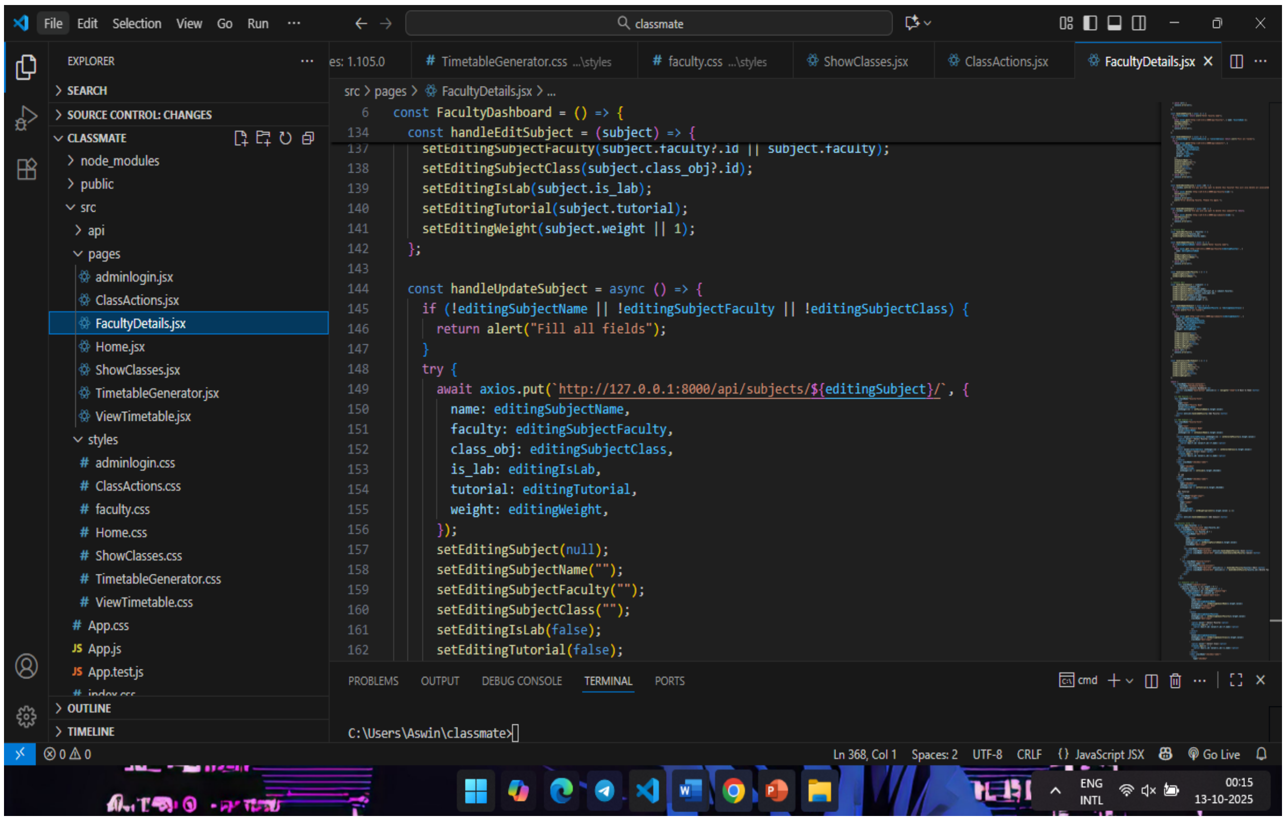
```jsx
import React, { useState, useEffect } from "react";
import axios from "axios";
import "../styles/TimetableGenerator.css";
import { useNavigate } from "react-router-dom";
import { writeFileXLSX, utils } from "xlsx";
import { jsPDF } from "jspdf";
import autoTable from "jspdf-autotable";

export default function TimetableGenerator() {
  const [classList, setClassList] = useState([]);
  const [selectedClasses, setSelectedClasses] = useState({});
  const [timetables, setTimetables] = useState([]);
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  // Fetch all classes from backend
  useEffect(() => {
    axios
      .get("http://127.0.0.1:8000/api/classes/")
      .then((res) => {
        const initialSelection = {};
        (res.data || []).forEach((cls) => {
          initialSelection[cls.id] = { selected: false, tutorial: false };
        });
        setSelectedClasses(initialSelection);
        setClassList(res.data || []);
      })
      .catch((err) => {
```

```
C:\Users\Aswin\classmate>
```



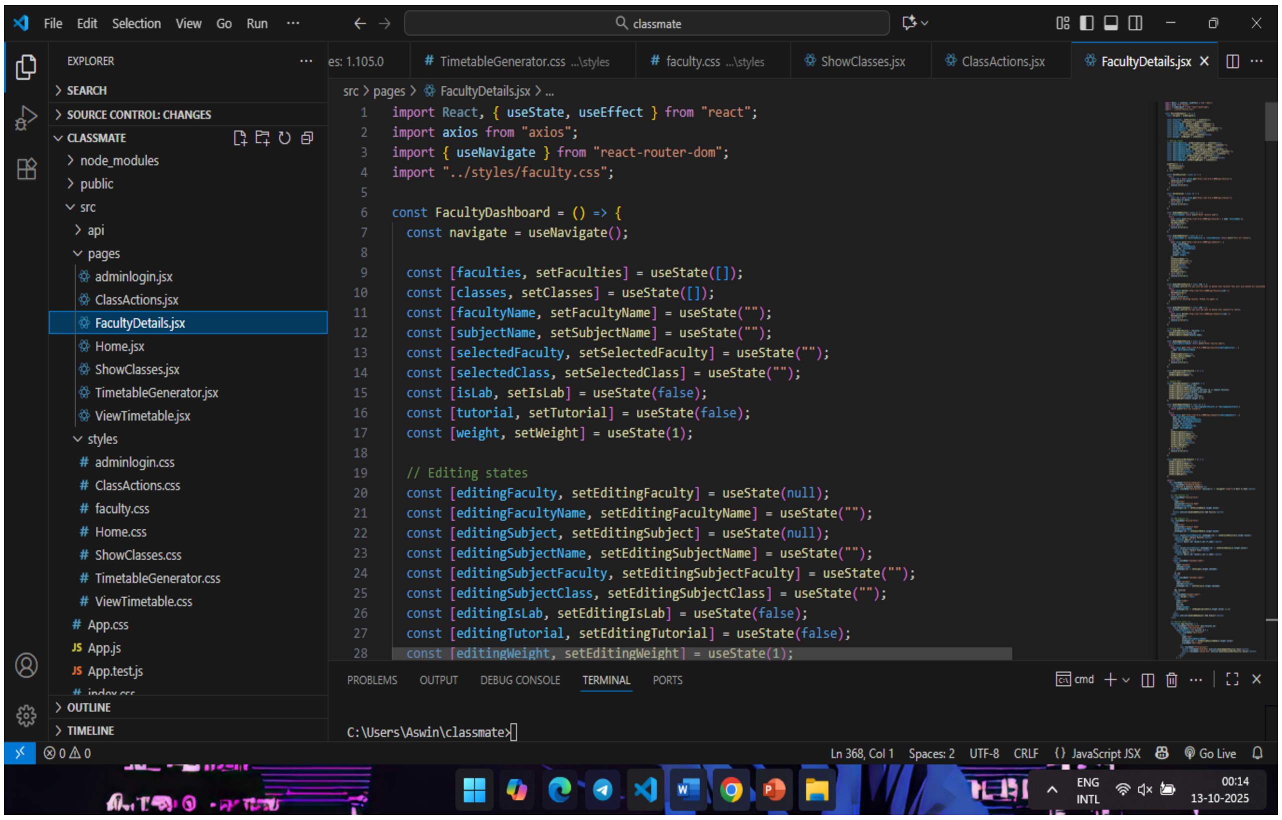
```jsx
export default function TimetableGenerator() {
  const handleSelectClass = (clsId) => {
    setSelectedClasses((prev) => ({
      ...prev,
      [clsId]: {
        ...prev[clsId],
        selected: !prev[clsId].selected,
        tutorial: prev[clsId].selected ? false : prev[clsId].tutorial,
      },
    }));
  };

  // Handle tutorial checkbox
  const handleTutorialCheck = (clsId) => {
    setSelectedClasses((prev) => ({
      ...prev,
      [clsId]: {
        ...prev[clsId],
        tutorial: !prev[clsId].tutorial,
      },
    }));
  };

  // Generate timetables
  const handleGenerate = async () => {
    const classesToSend = classList
      .filter((cls) => selectedClasses[cls.id]?.selected)
```

```
C:\Users\Aswin\classmate>
```



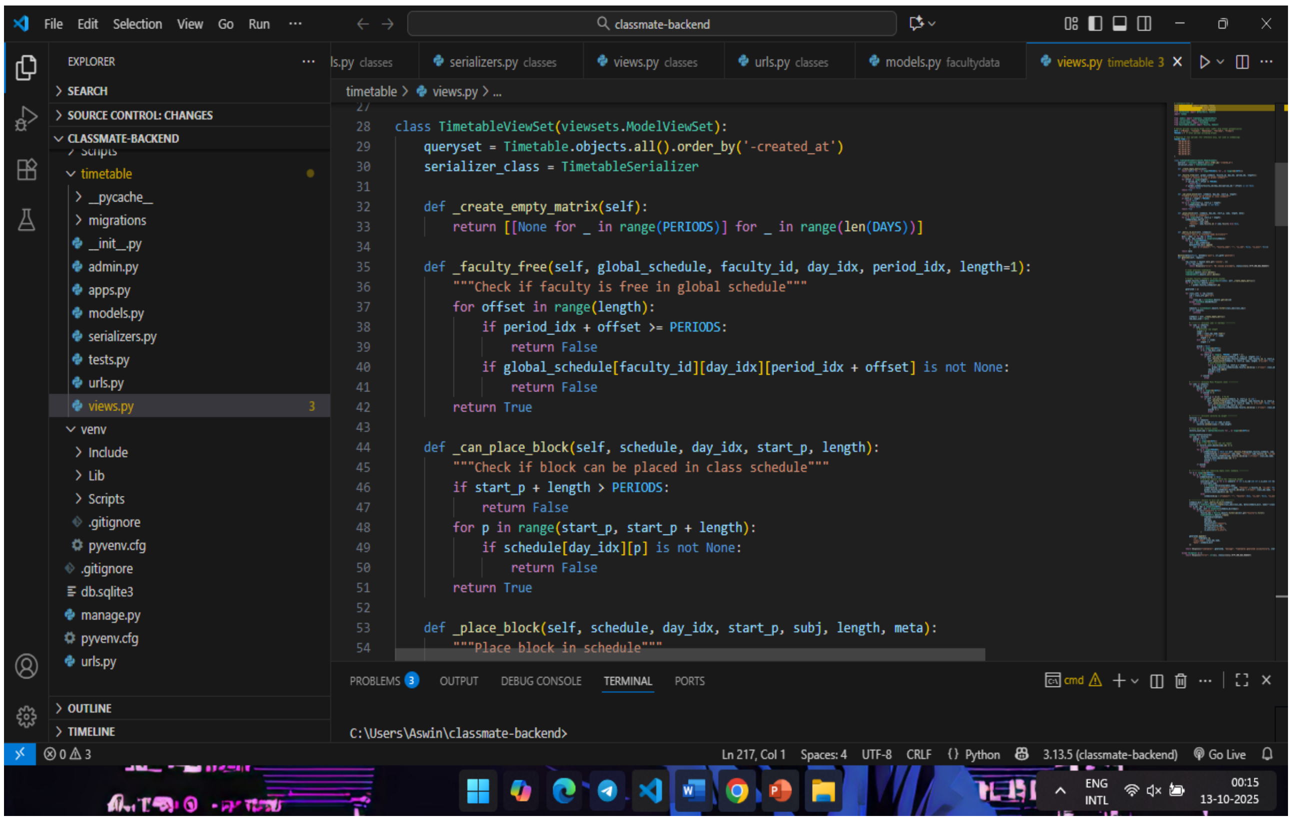
```
const FacultyDashboard = () => {
  const handleEditSubject = (subject) => {
    setEditingSubjectFaculty(subject.faculty?.id || subject.faculty);
    setEditingSubjectClass(subject.class_obj?.id);
    setEditingIsLab(subject.is_lab);
    setEditingTutorial(subject.tutorial);
    setEditingWeight(subject.weight || 1);
  };

  const handleUpdateSubject = async () => {
    if (!editingSubjectName || !editingSubjectFaculty || !editingSubjectClass) {
      return alert("Fill all fields");
    }
    try {
      await axios.put(`http://127.0.0.1:8000/api/subjects/${editingSubject}/`, {
        name: editingSubjectName,
        faculty: editingSubjectFaculty,
        class_obj: editingSubjectClass,
        is_lab: editingIsLab,
        tutorial: editingTutorial,
        weight: editingWeight,
      });
      setEditingSubject(null);
      setEditingSubjectName("");
      setEditingSubjectFaculty("");
      setEditingSubjectClass("");
      setEditingIsLab(false);
      setEditingTutorial(false);
```

```
C:\Users\Aswin\classmate>
```



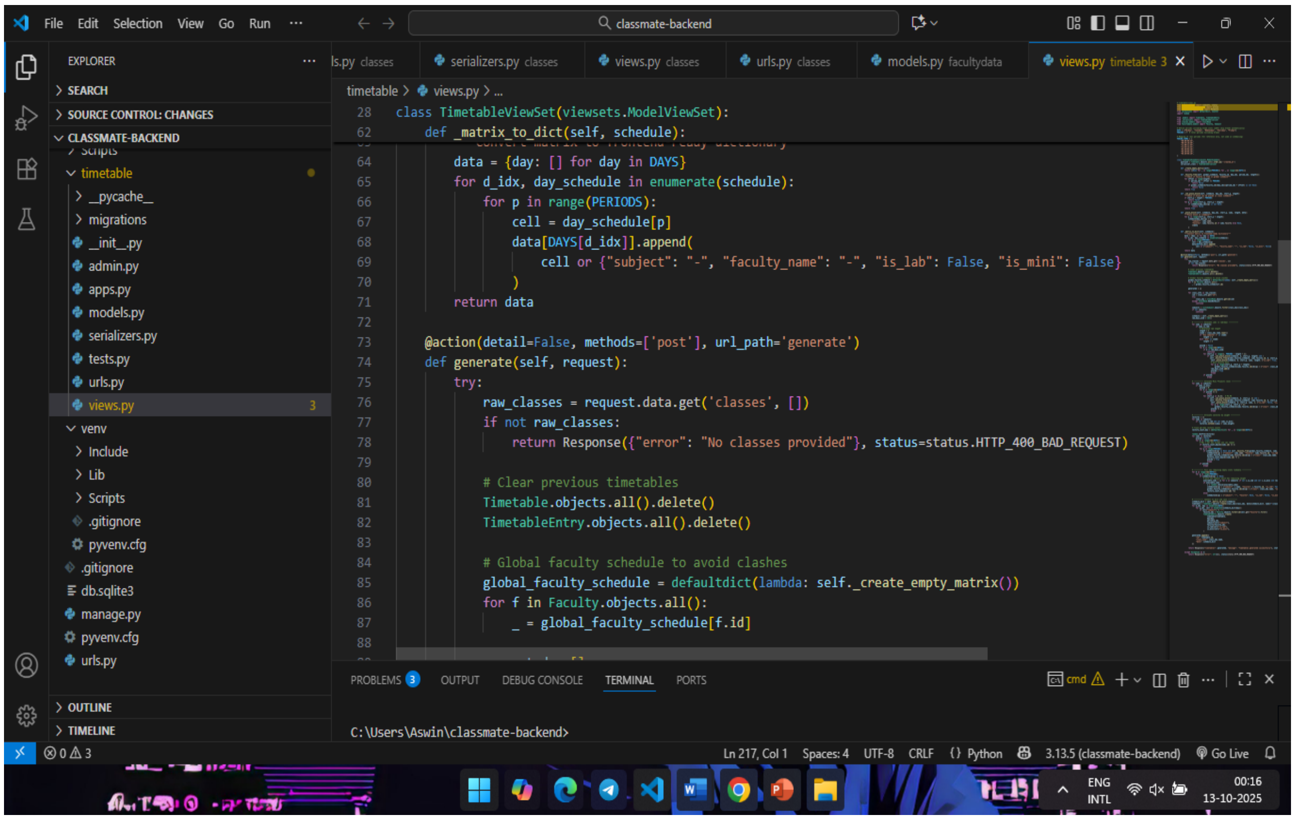
```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import "../styles/faculty.css";

const FacultyDashboard = () => {
  const navigate = useNavigate();

  const [faculties, setFaculties] = useState([]);
  const [classes, setClasses] = useState([]);
  const [facultyName, setFacultyName] = useState("");
  const [subjectName, setSubjectName] = useState("");
  const [selectedFaculty, setSelectedFaculty] = useState("");
  const [selectedClass, setSelectedClass] = useState("");
  const [isLab, setIsLab] = useState(false);
  const [tutorial, setTutorial] = useState(false);
  const [weight, setWeight] = useState(1);

  // Editing states
  const [editingFaculty, setEditingFaculty] = useState(null);
  const [editingFacultyName, setEditingFacultyName] = useState("");
  const [editingSubject, setEditingSubject] = useState(null);
  const [editingSubjectName, setEditingSubjectName] = useState("");
  const [editingSubjectFaculty, setEditingSubjectFaculty] = useState("");
  const [editingSubjectClass, setEditingSubjectClass] = useState("");
  const [editingIsLab, setEditingIsLab] = useState(false);
  const [editingTutorial, setEditingTutorial] = useState(false);
  const [editingWeight, setEditingWeight] = useState(1);
```

```
C:\Users\Aswin\classmate>
```



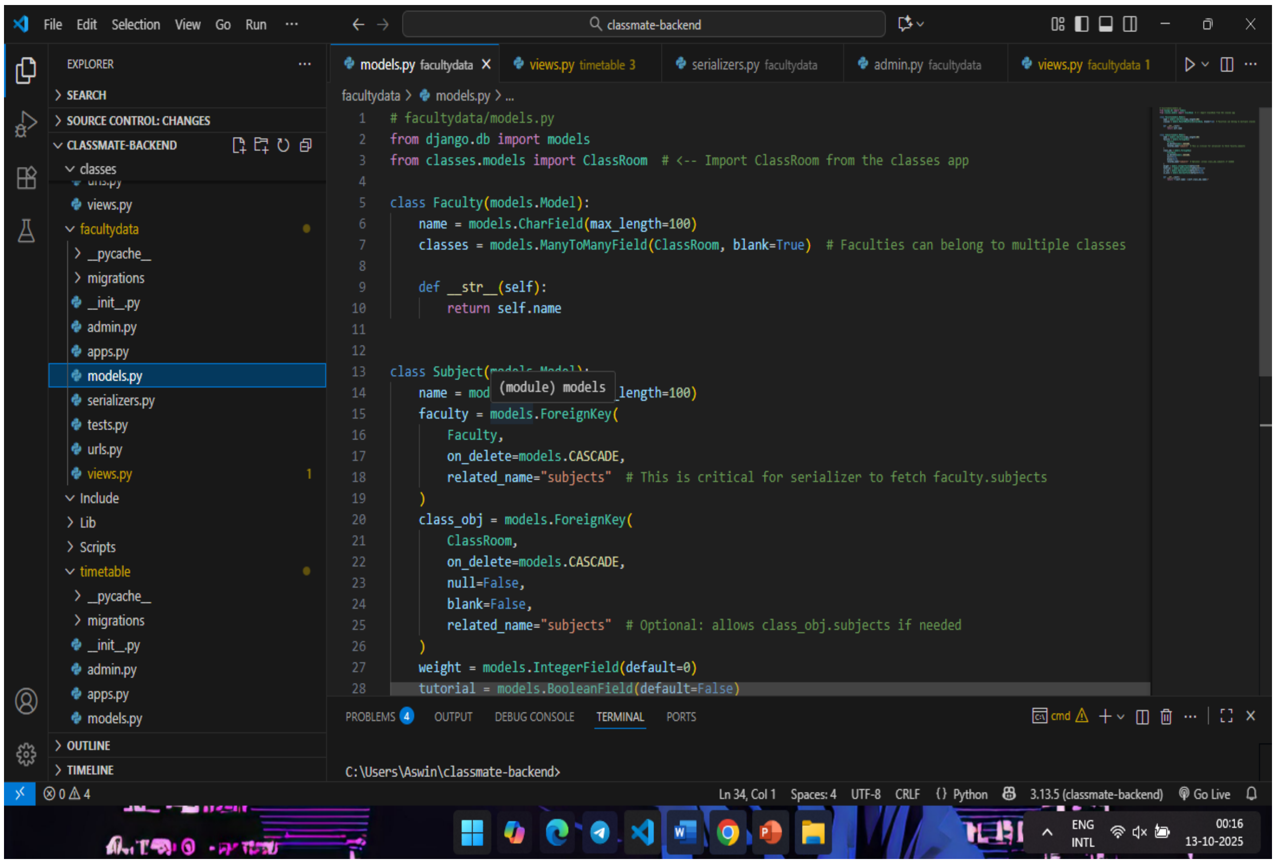
```python
class TimetableViewSet(viewsets.ModelViewSet):
    queryset = Timetable.objects.all().order_by('-created_at')
    serializer_class = TimetableSerializer

    def _create_empty_matrix(self):
        return [[None for _ in range(PERIODS)] for _ in range(len(DAYS))]

    def _faculty_free(self, global_schedule, faculty_id, day_idx, period_idx, length=1):
        """Check if faculty is free in global schedule"""
        for offset in range(length):
            if period_idx + offset >= PERIODS:
                return False
            if global_schedule[faculty_id][day_idx][period_idx + offset] is not None:
                return False
        return True

    def _can_place_block(self, schedule, day_idx, start_p, length):
        """Check if block can be placed in class schedule"""
        if start_p + length > PERIODS:
            return False
        for p in range(start_p, start_p + length):
            if schedule[day_idx][p] is not None:
                return False
        return True

    def _place_block(self, schedule, day_idx, start_p, subj, length, meta):
        """Place block in schedule"""
```

```
C:\Users\Aswin\classmate-backend>
```



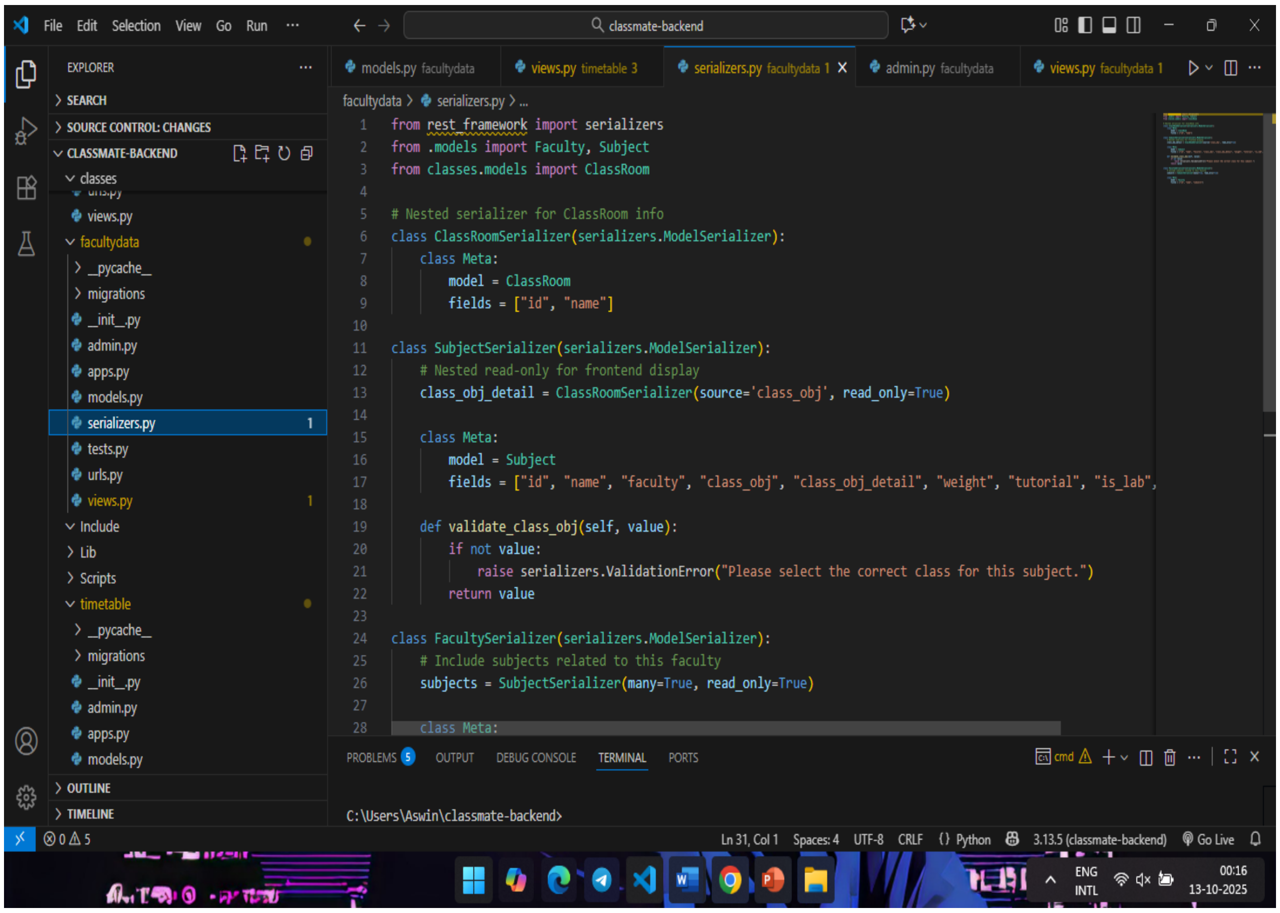
```python
class TimetableViewSet(viewsets.ModelViewSet):
    def _matrix_to_dict(self, schedule):
        data = {day: [] for day in DAYS}
        for d_idx, day_schedule in enumerate(schedule):
            for p in range(PERIODS):
                cell = day_schedule[p]
                data[DAYS[d_idx]].append(
                    cell or {"subject": "-", "faculty_name": "-", "is_lab": False, "is_mini": False}
                )
        return data

    @action(detail=False, methods=['post'], url_path='generate')
    def generate(self, request):
        try:
            raw_classes = request.data.get('classes', [])
            if not raw_classes:
                return Response({"error": "No classes provided"}, status=status.HTTP_400_BAD_REQUEST)

            # Clear previous timetables
            Timetable.objects.all().delete()
            TimetableEntry.objects.all().delete()

            # Global faculty schedule to avoid clashes
            global_faculty_schedule = defaultdict(lambda: self._create_empty_matrix())
            for f in Faculty.objects.all():
                _ = global_faculty_schedule[f.id]
```

```
C:\Users\Aswin\classmate-backend>
```



```python
# facultydata/models.py
from django.db import models
from classes.models import ClassRoom  # <-- Import ClassRoom from the classes app

class Faculty(models.Model):
    name = models.CharField(max_length=100)
    classes = models.ManyToManyField(ClassRoom, blank=True)  # Faculties can belong to multiple classes

    def __str__(self):
        return self.name


class Subject(models.Model):
    name = mod (module) models length=100)
    faculty = models.ForeignKey(
        Faculty,
        on_delete=models.CASCADE,
        related_name="subjects"  # This is critical for serializer to fetch faculty.subjects
    )
    class_obj = models.ForeignKey(
        ClassRoom,
        on_delete=models.CASCADE,
        null=False,
        blank=False,
        related_name="subjects"  # Optional: allows class_obj.subjects if needed
    )
    weight = models.IntegerField(default=0)
    tutorial = models.BooleanField(default=False)
```

```
C:\Users\Aswin\classmate-backend>
```



```python
from rest_framework import serializers
from .models import Faculty, Subject
from classes.models import ClassRoom

# Nested serializer for ClassRoom info
class ClassRoomSerializer(serializers.ModelSerializer):
    class Meta:
        model = ClassRoom
        fields = ["id", "name"]

class SubjectSerializer(serializers.ModelSerializer):
    # Nested read-only for frontend display
    class_obj_detail = ClassRoomSerializer(source='class_obj', read_only=True)

    class Meta:
        model = Subject
        fields = ["id", "name", "faculty", "class_obj", "class_obj_detail", "weight", "tutorial", "is_lab",

    def validate_class_obj(self, value):
        if not value:
            raise serializers.ValidationError("Please select the correct class for this subject.")
        return value

class FacultySerializer(serializers.ModelSerializer):
    # Include subjects related to this faculty
    subjects = SubjectSerializer(many=True, read_only=True)

    class Meta:
```

```
C:\Users\Aswin\classmate-backend>
```

REFERENCES

[1] A. Narang, A. Sharma, A. Tyagi, A. Sharma, and M. Shahid, "AI-Driven Timetable Generator Using Constraint Programming," *Tuijin Jishu / Journal of Propulsion Technology*, vol. 45, no. 2, pp. 112–118, 2025.

[2] S. Thakare, T. Nikam, and M. Patil, "Automatic Timetable Generation Using Genetic Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, issue 4, pp. 1324–1330, 2020.

[3] S. Ambhore, R. Deshmukh, M. Patil, and K. Ghodke, "Automated Timetable Generation Using Machine Learning Algorithms," *International Research Journal of Engineering Science and Management (IRJESM)*, vol. 5, issue 3, pp. 45–52, 2020.

[4] A. Latpate, N. Sayyad, C. Bargal, A. Sawant, and J. S. Choudhari, "AI-Based Automatic Timetable Generator Using React and Firebase," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 12, issue 4, pp. 985–992, 2024.

[5] L. A. Farinola and M. B. M. Assogba, "Explicit Artificial Intelligence Timetable Generator for Colleges and Universities," *Open Journal of Applied Sciences (OJAPPS)*, vol. 15, issue 1, pp. 211–218, 2025.

[6] P. A. Parkavi, "A Study on Automatic Timetable Generator," *International Journal of Innovative Research in Graduate Studies (IJIRG)*, vol. 5, issue 5, pp. 78–83, 2018.

[7] D. Brahmbhatt, H. Patel, K. Prajapati, J. Gevariya, and D. George, "Automatic Timetable Generation Using Genetic and Heuristic Algorithms," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 9, issue 6, pp. 1642–1648, 2022.

[8] D. Mahajan, G. Malakar, R. Lath, T. More, and D. Jain, "Timely Trigger: A Smart Timetable Generator," *International Journal of Novel Research and Development (IJNRD)*, vol. 9, issue 4, pp. 25–32, 2024.

[9] H. Techie-Menson and P. Nyagorme, "Web-Based Timetable Generation System for Higher Education Institutions," *International Journal of Engineering Research and Innovation Studies (IJERIS)*, vol. 8, issue 3, pp. 66–73, 2021.

[10] A. Puttaswamy, H. M. A. A. Khan, C. S. V., and P. A. Parkavi, "Heuristic and Genetic Algorithm-Based Timetable Optimization System," *International Journal of Science and Innovative Engineering & Technology (IJSIET)*, vol. 5, issue May 2018, pp. 21–28, 2018.

[11] A. Thakur, S. Menon, and J. Lobo, "Optimization of University Timetable Using Hybrid Genetic Algorithm," *International Journal of Computer Applications (IJCA)*, vol. 182, no. 25, pp. 15–20, 2023.