Detailed Project Report

on

Automated Time-Table Scheduling for IIIT Dharwad

Submitted by

Team: Byte_Me

Team members present for the lab session:

Padala Gnandeep 24BCS095

Phanishree N 24BCS101

Shriyans S S Sahoo 24BCS142

Syed Mahdee Husain 24BCS156

Under the guidance of

Dr. Vivekraj V K

Assistant Professor, IIIT Dharwad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

16/09/2025

Contents

List of Figures

List of Tables

1 Introduction

In many educational institutions, preparing class schedules is still done manually or with basic tools. This method takes a lot of time and often leads to mistakes. Errors such as scheduling conflicts, overbooked classrooms, or uneven faculty workloads are common. Automating this process can save time, reduce errors, and ensure smooth scheduling for both students and teachers.

Creating a timetable involves scheduling lectures, labs, and tutorials for multiple sections, professors, and classrooms. It also requires considering factors like professor availability, lab requirements, self-study slots, and breaks. Doing this manually is a slow and error-prone process. There is scope of making mistakes in such a system. The aim of this project is to develop a timetable automation software that can create fully optimized and conflict-free schedules based on given inputs and institutional rules.

There are various constraints under which the software must create the time table. The classes start at 9 a.m. and end by 6 p.m. All the regular classes must be scheduled in this slot. The minor slots are scheduled either from 7:30 to 9:00 a.m. or after 6 p.m. The lunch break is from 1:15 p.m. till 2:00 p.m.

The software will take in details such as course codes, course names, professors, list of available classrooms and laboratories as input from the backend. Using this data, it will be able to generate separate timetables for each batch, section and year. The goal is to make a system that not only automates scheduling but also ensures it can be easily implemented in a real college environment.

This LaTeX[4] document lists down the various requirements that need to be kept in mind while designing this software, i.e, what the software will be able to do once it is built and fed the right inputs [2].

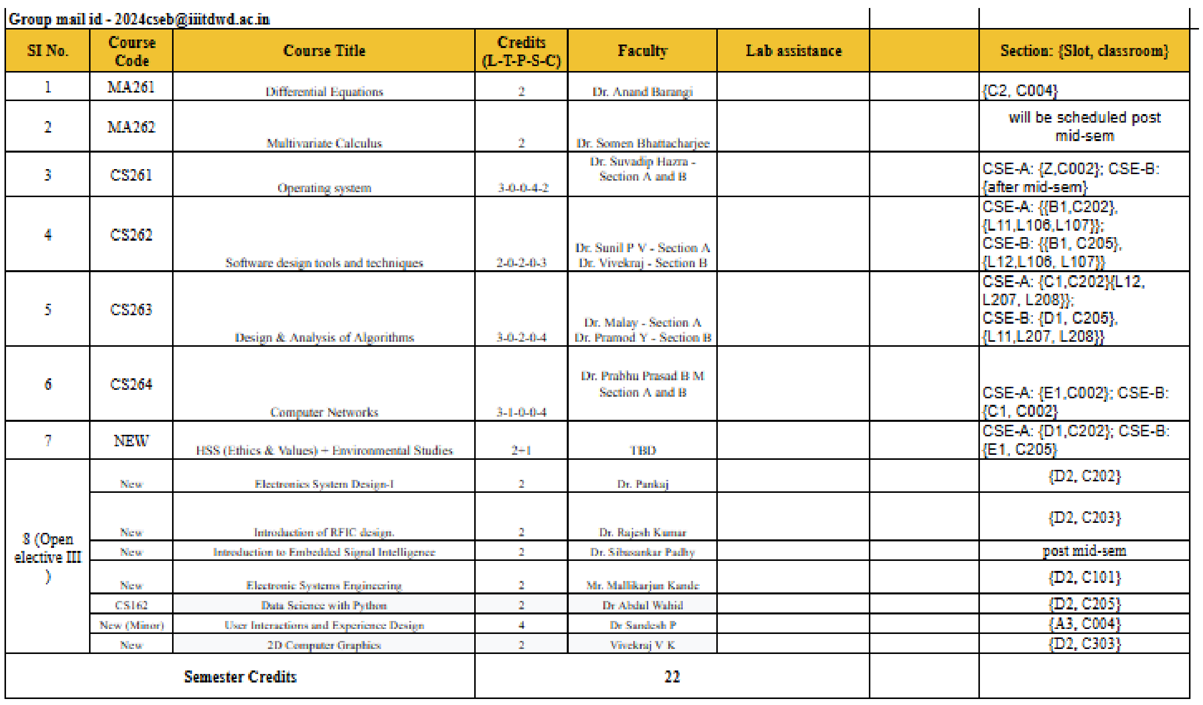
2 Existing System

Currently, the college timetable is generated automatically and presented as a single general timetable that uses slot codes (e.g., A1, E1, etc.) to represent lecture periods. These codes are then linked to separate pages for different semesters, where students can find the specific courses assigned to each slot.

The time-table operates on a slot-based scheduling system, where each day of the week is divided into predefined slots. A common slots tab shows the timing of each slot across the week, while the course mapping pages display which subject corresponds to each slot for a given semester [1].

Issues in the Existing Approach

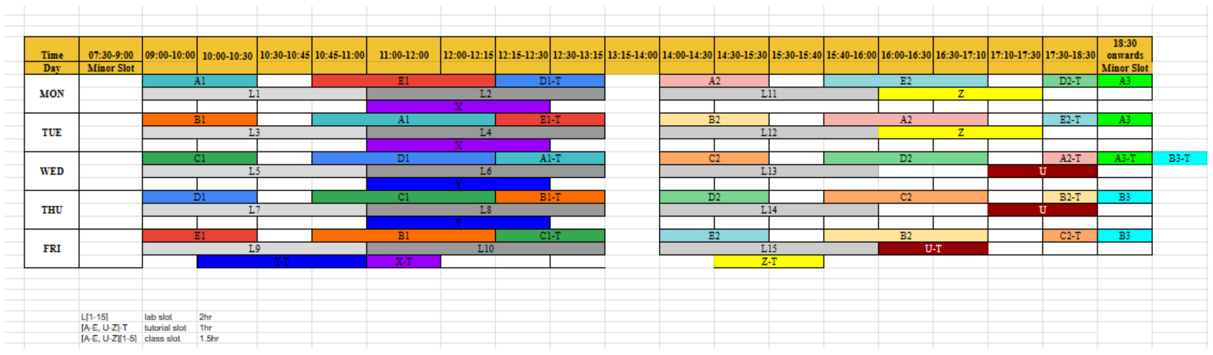
- Complex Access Process – Students must navigate between multiple tabs/-pages (common slot timings and course mapping) to know their schedule, which is inconvenient.
- Time-Consuming Interpretation – Decoding slot codes and cross-referencing them with separate semester pages slows down the process of checking daily classes.
- Prone to Errors – Students may misread slot codes or overlook changes, leading to missed or late attendance in classes.
- Lack of Personalization – The timetable is not directly customized for individual students; everyone receives the same general schedule, even if only a subset applies to them.
- Poor Real-Time Updates – Any last-minute changes (e.g., class rescheduling, faculty absence) require manual checking or external notifications, which can be easily missed.
- Not Mobile-Friendly – Accessing the timetable on mobile devices is cumbersome, as navigating between multiple tabs is less efficient on smaller screens.



Group mail id - 2024cseb@iiitdwd.ac.in

Sl No.	Course Code	Course Title	Credits (L-T-P-S-C)	Faculty	Lab assistance		Section: {Slot, classroom}
1	MA261	Differential Equations	2	Dr. Anand Barangi			{C2, C004}
2	MA262	Multivariate Calculus	2	Dr. Somen Bhattacharjee			will be scheduled post mid-sem
3	CS261	Operating system	3-0-0-4-2	Dr. Suvadip Hazra - Section A and B			CSE-A: {Z,C002}; CSE-B: {after mid-sem}
4	CS262	Software design tools and techniques	2-0-2-0-3	Dr. Sunil P V - Section A Dr. Vivekraj - Section B			CSE-A: {{B1,C202}, {L11,L106,L107}}; CSE-B: {{B1, C205}, {L12,L106, L107}}
5	CS263	Design & Analysis of Algorithms	3-0-2-0-4	Dr. Malay - Section A Dr. Pramod Y - Section B			CSE-A: {C1,C202}{L12, L207, L208}}; CSE-B: {D1, C205}, {L11,L207, L208}}
6	CS264	Computer Networks	3-1-0-0-4	Dr. Prabhu Prasad B M Section A and B			CSE-A: {E1,C002}; CSE-B: {C1, C002}
7	NEW	HSS (Ethics & Values) + Environmental Studies	2+1	TBD			CSE-A: {D1,C202}; CSE-B: {E1, C205}
8 (Open elective III)	New	Electronics System Design-I	2	Dr. Pankaj			{D2, C202}
	New	Introduction of RFIC design.	2	Dr. Rajesh Kumar			{D2, C203}
	New	Introduction to Embedded Signal Intelligence	2	Dr. Sibasankar Padhy			post mid-sem
	New	Electronic Systems Engineering	2	Mr. Mallikarjun Kande			{D2, C101}
	CS162	Data Science with Python	2	Dr Abdul Wahid			{D2, C205}
	New (Minor)	User Interactions and Experience Design	4	Dr Sandesh P			{A3, C004}
	New	2D Computer Graphics	2	Vivekraj V K			{D2, C303}
Semester Credits			22				

Figure 1. 3rd Sem Decoding Page



Time / Day	07:30-9:00 Minor Slot	09:00-10:00	10:00-10:30	10:30-10:45	10:45-11:00	11:00-12:00	12:00-12:15	12:15-12:30	12:30-13:15	13:15-14:00	14:00-14:30	14:30-15:30	15:30-15:40	15:40-16:00	16:00-16:30	16:30-17:10	17:10-17:30	17:30-18:30	18:30 onwards Minor Slot	
MON		A1			E1			D1-T			A2			E2				D2-T	A3	
		L1				L2					L11				Z					
						X														
TUE		B1			A1			E1-T			B2			A2				E2-T	A3	
		L3				L4					L12				Z					
						X														
WED		C1			D1			A1-T			C2			D2				A2-T	A3-T	B3-T
		L5				L6					L13						U			
						Y														
THU		D1			C1			B1-T			D2			C2				B2-T	B3	
		L7				L8					L14						U			
						Y														
FRI		E1			B1			C1-T			E2			B2				C2-T	B3	
		L9				L10					L15				U-T					
			Y-T			X-T						Z-T								

L[1-15]	lab slot	2hr
[A-E, U-Z]-T	tutorial slot	1hr
[A-E, U-Z][1-5]	class slot	1.5hr

Figure 2. General Timetable with List of Slots

3 Requirements Modeling

The list of requirements for the proposed software and the UML Use-Case Diagram, describing all it's stakeholders is given in this section.

3.1 List of Requirements

List of all the requirements for the software and their description in a tabular format [3].

Requirement	**Description**
User Authentication for different Users	The system should allow secure login system for different roles (e.g administrator, professor, student etc.) and have edit access given to administrator
Admin Management Panel	Admin should be able to add/edit/delete faculty, subjects, rooms and time slots.
Faculty Availability Input	Faculty should be able to login and select unavailable slots and the system must consider it while scheduling the classes
Course and Resource Details	The system must allow complete entry of course information, including course code, course name, L-T-P-S-C structure, instructor, semester, and number of registered students. It should also store room/lab capacities to avoid overbooking. (L-T-P-S-C = Lecture - Tutorial - Practical - Self-Study - Credits)
Automatic Timetable Generation	The software should generate conflict-free, optimized timetables automatically, based on all inputs and constraints. This includes: • No clashes for faculty, classrooms, labs, or lab assistants. • One lecture per subject per day. • Proper 10-minute breaks between lectures. • Dedicated lunch break aligned with mess timings.
Lab and Theory Scheduling Rules	If lab is there then preferably a class of same subject should be before lab on same day.

Self-Study and Breaks	Allocate dedicated self-study hours for students. Ensure no faculty or student has continuous classes without a break, and lunch is always included in the schedule.
Faculty Workload Constraints	Allow setting limits such as maximum hours per day/per week* for faculty.Breaks between lectures (ideally 3 hours) for preparation.
Electives and Half-Semester Courses	Different elective courses that are meant for the same group of students should always be in the same time slot
Manual Adjustments	After the time table is generated, there should be an option for the admin to manually modify the time table if needed.
Different Views	The software must generate different time tables for students professors.
Visual Presentation	The generated time table should have color-coded slots for easy identification of subjects, labs and electives.
Exporting and Sharing	The generated time table should be downloadable in PDF or Excel format for the ease of access to students.
Integration with Google Calendar	The time table that has been generated should also integrate with the student's Google Calendar where the scheduled classes are visible and a reminder is sent to the student automatically before each class.
Notifications and Updates	Any changes, conflicts or updates to the time table should be notified to the students/relevant users.
Exam Timetable Generation	The system should automatically generate a conflict-free exam timetable for all courses, ensuring no overlapping exams for any student, providing adequate gaps, and scheduling theory and lab exams appropriately.
Classroom and Seating Allocation	The system should allocate classrooms and labs for exams based on capacity, ensuring proper distribution of students and seating such that no two students of the same exam sit next to each other.
Invigilator Assignment	The system should assign invigilators fairly across all exams while respecting faculty workload limits and avoiding conflicts with their own course exams.

Seating Plan Generation	The software should generate detailed seating plans with roll numbers mapped to seats, ensuring separation rules are followed, and export them in printable formats for admin and invigilators.
Final Product	The final product should include the classes and examination timetable generator, all program files, proper documentation and a user manual with output samples.

Table 1
Requirements Table

3.2 Use-Case Diagram

This is the Use-Case Diagram for the Automated Time-Table Scheduling Software. It shows the use case of this software for various stakeholders, like Administrator, Faculty, Students and Technical Support.

This use case diagram was created using the software, StarUML. StarUML is a software modeling tool used to design and visualize software systems. It supports creating UML (Unified Modeling Language) diagrams, which help developers and designers understand, document, and plan software architecture [5].

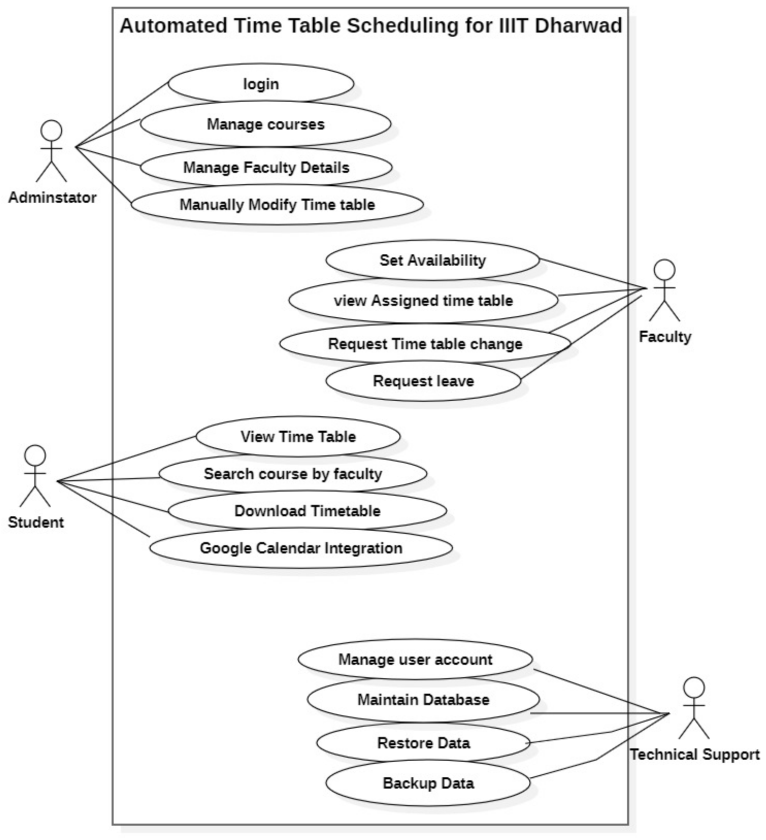
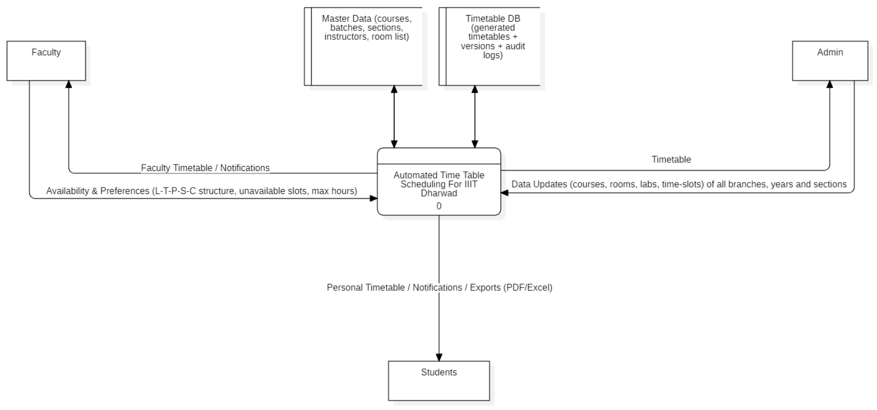


Figure 3. Use Case Diagram

4 Software Design



Master Data (courses, batches, sections, instructors, room list)
Timetable DB (generated timetables + versions + audit logs)
Faculty
Admin
Faculty Timetable / Notifications
Availability & Preferences (L-T-P-S-C structure, unavailable slots, max hours)
Automated Time Table Scheduling For IIIT Dharwad
0
Timetable
Data Updates (courses, rooms, labs, time-slots) of all branches, years and sections
Personal Timetable / Notifications / Exports (PDF/Excel)
Students

Figure 4. Level 0 - Context Diagram

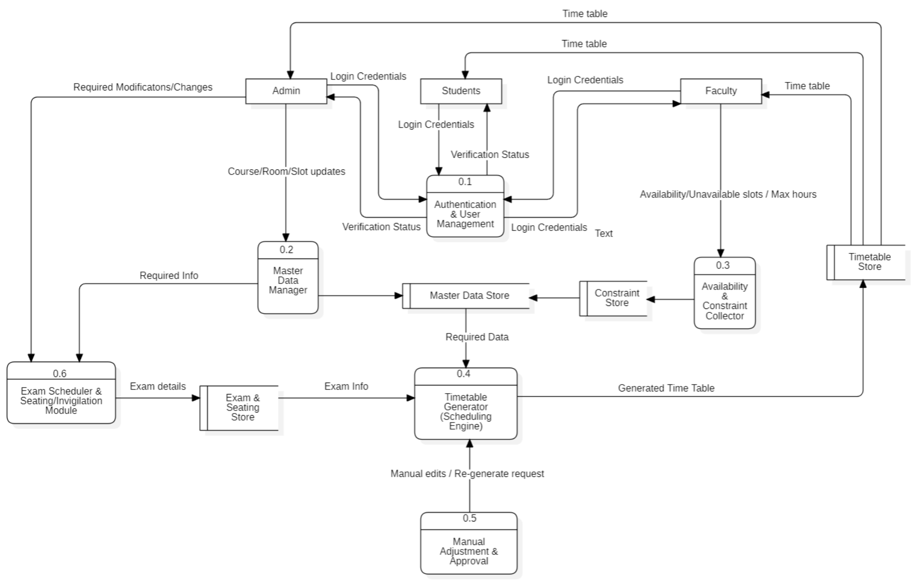
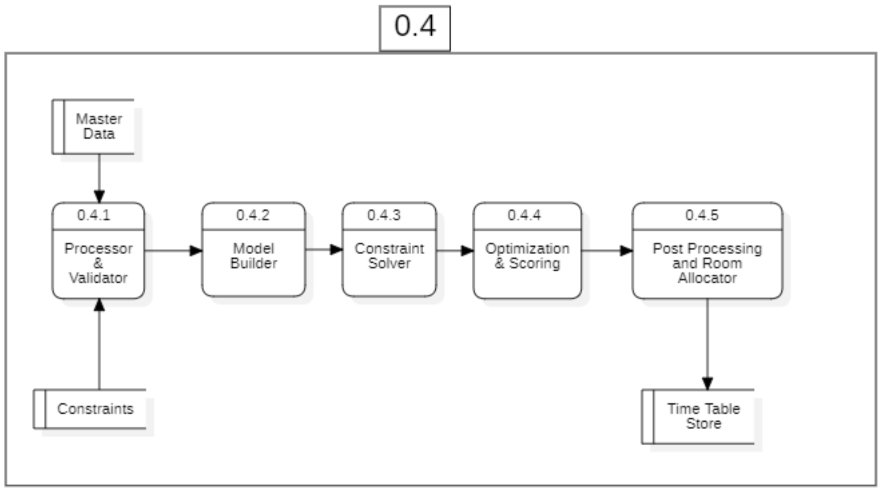


Figure 5. Level 1 DFD Diagram



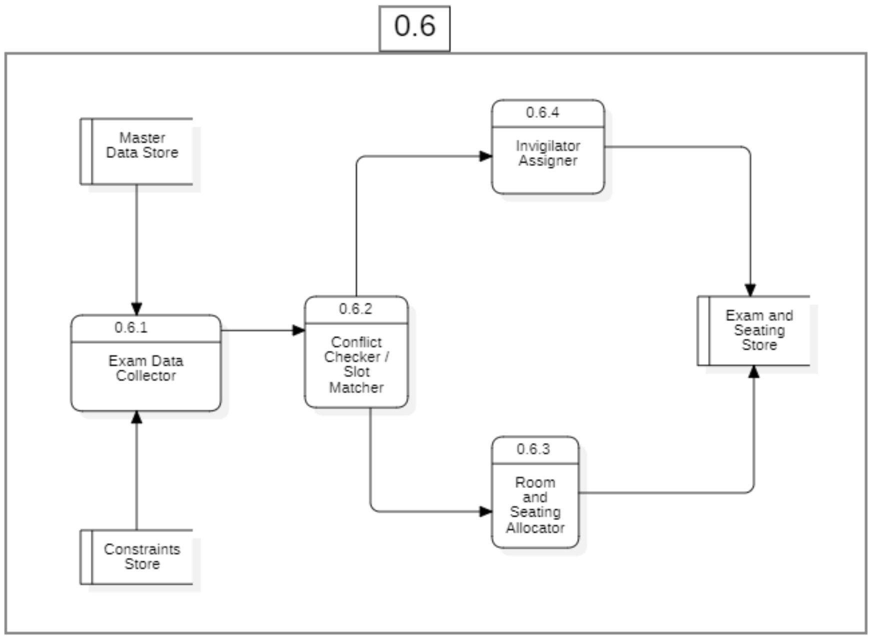


Figure 6. Level 2 DFD Diagram

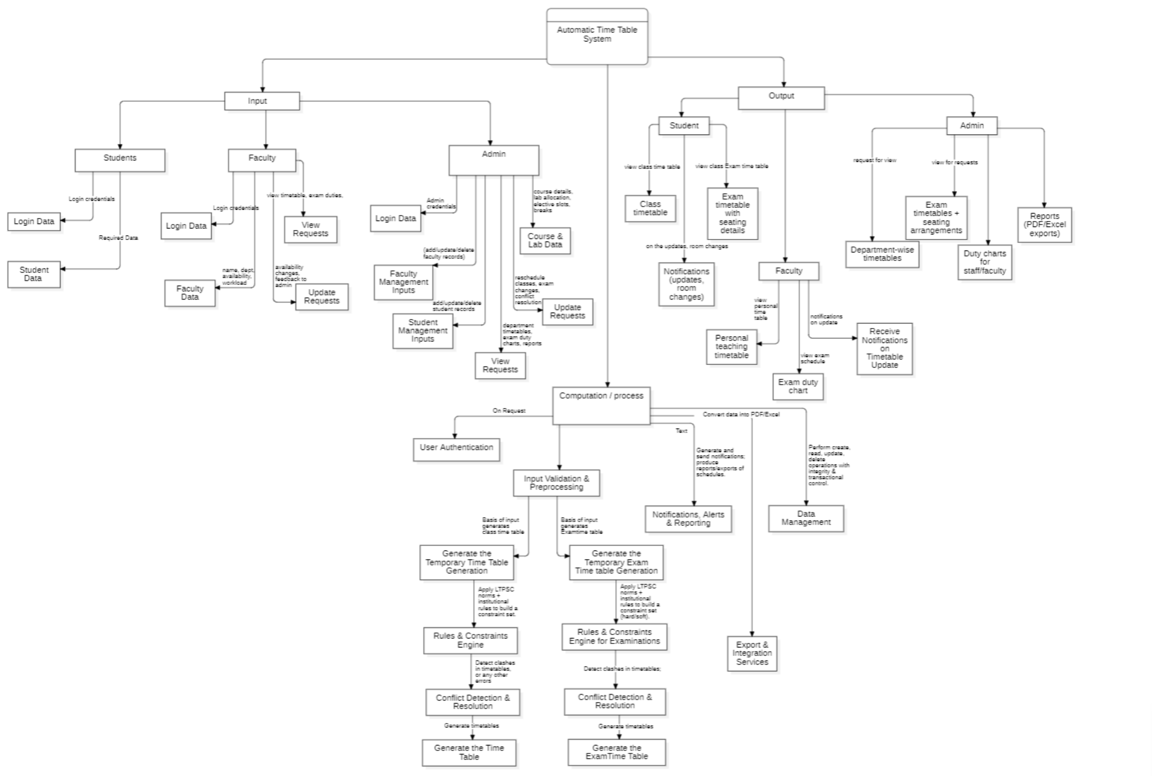
4.1 Data Dictionary

```
Course = {id, code, title, lectures_per_week, lab_hours_per_week,
preferred_room_type, student_groups}
Faculty = {id, name, max_slots_per_day, unavailable_slots, preferred_slots}
Room = {id, room_type, capacity, location}
StudentGroup = {id, size, courses}
Slot = {id, day, start_time, end_time}
Assignment = {slot_id, room_id, faculty_id, course_id, group_id}
Timetable = {assignments: list(Assignment), score: float}
```

5 Module Structure

5.1 High Level Design

Since this is a very complex diagram, it is not clearly visible. Please refer to the .mdj file for the High Level Design.



Automatic Time Table System
Input
Students
Login credentials
Login Data
Required Data
Student Data
Faculty
Login credentials
Login Data
view timetable, exam duties,
View Requests
name, dept, availability, workload
Faculty Data
availability changes, feedback to admin
Update Requests
Admin
Admin credentials
Login Data
course details, lab allocation, elective slots, breaks
Course & Lab Data
(add/update/delete faculty records)
Faculty Management Inputs
add/update/delete student records
Student Management Inputs
reschedule classes, exam changes, conflict resolution
Update Requests
department timetables, exam duty charts, reports
View Requests
Output
Student
view class time table
Class timetable
view class Exam time table
Exam timetable with seating details
on the updates, room changes
Notifications (updates, room changes)
Faculty
view personal time table
Personal teaching timetable
view exam schedule
Exam duty chart
notifications on update
Receive Notifications on Timetable Update
Admin
request for view
Department-wise timetables
view for requests
Exam timetables + seating arrangements
Duty charts for staff/faculty
Reports (PDF/Excel exports)
Computation / process
On Request
User Authentication
Input Validation & Preprocessing
Basis of input generates class time table
Generate the Temporary Time Table Generation
Apply LTPSC norms + institutional rules to build a constraint set.
Rules & Constraints Engine
Detect clashes in timetables, or any other errors
Conflict Detection & Resolution
Generate timetables
Generate the Time Table
Basis of input generates Examtime table
Generate the Temporary Exam Time table Generation
Apply LTPSC norms + institutional rules to build a constraint set (hard/soft).
Rules & Constraints Engine for Examinations
Detect clashes in timetables;
Conflict Detection & Resolution
Generate timetables
Generate the ExamTime Table
Text
Generate and send notifications; produce reports/exports of schedules.
Notifications, Alerts & Reporting
Convert data into PDF/Excel
Export & Integration Services
Perform create, read, update, delete operations with integrity & transactional control.
Data Management

Figure 7. High Level Design

5.2 Low-Level Design — Function Declaration

5.2.1 Authentication & User Management

Responsible for secure login, session handling, and user administration.

- **login(user_id, password)**
  Authenticates the user credentials. If valid, creates a session and grants role-based access (Admin, Faculty, Student).
  Inputs: User ID, Password
  Outputs: Success/Failure, Session Token

- **logout(user_id)**
  Ends the active session of a user. Ensures data security and prevents unauthorized access.
  Inputs: User ID
  Outputs: Confirmation of logout

- **createUser(user_data)**
  Creates a new user in the system with profile details.
  Inputs: User details (name, email, role, etc.)
  Outputs: Unique User ID

- **updateUser(user_id, updated_data)**
  Allows modification of user details.
  Inputs: User ID, Updated Data
  Outputs: Confirmation of update

- **deleteUser(user_id)**
  Removes a user's account and access rights.
  Inputs: User ID
  Outputs: Confirmation of deletion

5.2.2 Admin Module

Provides full control for course management, room management, and timetable modifications.

- **addCourse(course_data)**
  Registers a new course.

Inputs: Course details
Outputs: Course ID

- **updateCourse(course_id, data)**
  Modifies existing course details.
  Inputs: Course ID, Updated Data
  Outputs: Confirmation of update
- **deleteCourse(course_id)**
  Removes a course permanently.
  Inputs: Course ID
  Outputs: Success/Failure
- **addRoom(room_data)**
  Adds a new classroom or lab.
  Inputs: Room details (capacity, type, availability)
  Outputs: Room ID
- **updateRoom(room_id, data)**
  Edits classroom details.
  Inputs: Room ID, Updated Data
  Outputs: Confirmation of update
- **deleteRoom(room_id)**
  Removes a classroom/lab.
  Inputs: Room ID
  Outputs: Confirmation of deletion
- **manualAdjustments(timetable_id, changes)**
  Overrides auto-generated timetable.
  Inputs: Timetable ID, Change details
  Outputs: Updated Timetable

5.2.3 Faculty Module

Enables faculty to manage availability and view schedules.

- **setFacultyAvailability(faculty_id, unavailable_slots)**
  Faculty specifies unavailable slots.
  Inputs: Faculty ID, Slots
  Outputs: Confirmation

- **getFacultyAvailability(faculty_id)**
  Retrieves availability.
  Inputs: Faculty ID
  Outputs: Slots list
- **viewTimetable(faculty_id)**
  Provides faculty timetable.
  Inputs: Faculty ID
  Outputs: Timetable
- **viewExamDuties(faculty_id)**
  Displays exam invigilation duties.
  Inputs: Faculty ID
  Outputs: Duties list

5.2.4 Student Module

Provides students with timetables, exam schedules, and notifications.

- **viewTimetable(student_id)**
  Shows timetable for a student.
  Inputs: Student ID
  Outputs: Timetable
- **viewExamSchedule(student_id)**
  Provides exam schedule.
  Inputs: Student ID
  Outputs: Exam timetable
- **selectElectives(student_id, elective_list)**
  Registers electives.
  Inputs: Student ID, Electives
  Outputs: Confirmation
- **getNotifications(student_id)**
  Retrieves alerts and updates.
  Inputs: Student ID
  Outputs: Notifications

5.2.5 Core System Module

Central engine for timetable and exam scheduling.

- **generateTimetable(batch_id)**
  Creates conflict-free timetables.
  Inputs: Batch ID
  Outputs: Timetable
- **validateTimetable(timetable_id)**
  Checks clashes and compliance.
  Inputs: Timetable ID
  Outputs: Valid/Invalid
- **generateExamSchedule(courses, constraints)**
  Creates an exam schedule.
  Inputs: Courses, Constraints
  Outputs: Exam timetable
- **allocateRooms(exam_id, rooms)**
  Assigns classrooms/labs for exams.
  Inputs: Exam ID, Rooms
  Outputs: Allocation list
- **generateSeatingPlan(exam_id, students, room_id)**
  Produces seating arrangements.
  Inputs: Exam ID, Students, Room ID
  Outputs: Seating plan
- **assignInvigilators(exam_id, faculty_list)**
  Assigns invigilators.
  Inputs: Exam ID, Faculty list
  Outputs: Assignment list

5.2.6 Output & Integration Module

Handles outputs, exports, and system integrations.

- **exportTimetable(timetable_id, format)**
  Exports timetable.

Inputs: Timetable ID, Format

Outputs: File download

- **exportExamSchedule(exam_id, format)**

  Exports exam schedule.

  Inputs: Exam ID, Format

  Outputs: Schedule file

- **integrateWithGoogleCalendar(user_id, timetable_id)**

  Syncs with Google Calendar.

  Inputs: User ID, Timetable ID

  Outputs: Calendar events

- **sendNotification(user_id, message)**

  Sends alerts.

  Inputs: User ID, Message

  Outputs: Confirmation

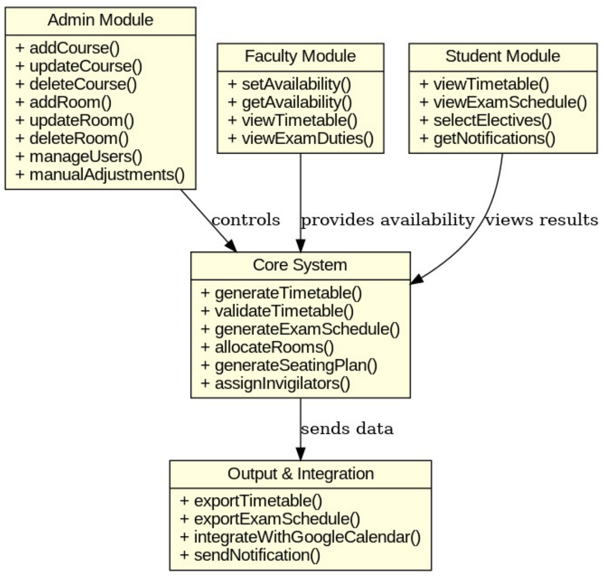


Figure 8. Low Level Design - Illustration

5.3 Low Level Design — Data Structures

The following data structures have been designed to efficiently store and manage the information required for the Automated Timetable Scheduling system.

- **Course**
  - Fields: `id, code, title, lectures_per_week, lab_hours_per_week, student_groups, preferred_room_type`
  - Description: Stores details about each course including scheduling preferences and associated student groups.
- **Faculty**
  - Fields: `id, name, max_slots_per_day, unavailable_slots, preferred_slots`
  - Description: Contains faculty scheduling constraints such as availability and preferred slots.
- **Room**
  - Fields: `id, room_type, capacity, location`
  - Description: Stores information regarding classrooms or laboratories used for scheduling.
- **StudentGroup**
  - Fields: `id, size, courses`
  - Description: Represents batches or sections of students and the courses they are enrolled in.
- **Slot**
  - Fields: `id, day, start_time, end_time`
  - Description: Defines the time slots available for scheduling lectures or labs.
- **Assignment**
  - Fields: `slot_id, room_id, faculty_id, course_id, group_id`
  - Description: Represents the allocation of a course to a specific slot, room, faculty member, and student group.

- **Timetable**
  - Fields: `assignments, score`
  - Description: Contains a list of assignments and a score to evaluate the quality of the timetable.

Supporting Structures

- Dictionaries/maps are used to quickly access data such as courses, rooms, faculties, and slots by their IDs.
- Sets are employed for constraints such as unavailable and preferred slots, allowing efficient lookups.
- Lists are used to maintain ordered schedules and assignments.

These data structures ensure that the timetable scheduling system is efficient, scalable, and adaptable to changes in constraints and data size.

5.4 Low Level Design — Algorithms

The algorithms below describe the key scheduling processes used in the Automated Timetable Scheduling system. These algorithms focus on conflict resolution, constraint checking, and optimization.

Generate Timetable Algorithm

This algorithm assigns courses to available slots and rooms using a recursive backtracking approach while optimizing for schedule constraints.

Algorithm 1 Generate Timetable

GenerateTimetablebatch_id $timetable \leftarrow$ empty timetable $courses \leftarrow$ get courses for batch_id $slots \leftarrow$ get available slots $rooms \leftarrow$ get available rooms **define** recursive function AssignCourses(index) index equals number of courses $timetable.score \leftarrow$ calculate score True $course \leftarrow courses[index]$ slot in slots room in rooms can assign course to slot and room add assignment to timetable AssignCourses(index + 1) True remove assignment from timetable False

Validate Timetable Algorithm

This algorithm ensures that there are no scheduling conflicts and that all constraints are satisfied.

Algorithm 2 Validate Timetable

ValidateTimetabletimetable assignment in timetable.assignments slot is not free or room capacity exceeded or faculty unavailable False True

Allocate Rooms for Exams

This algorithm assigns rooms based on available capacity and student count.

Algorithm 3 Allocate Rooms

AllocateRoomsexam_id, rooms $students \leftarrow$ get students for exam_id sort rooms by capacity in descending order $allocation \leftarrow$ empty dictionary $i \leftarrow 0$ student in students $room \leftarrow rooms[i \mod length(rooms)]$ assign student to room $i \leftarrow i + 1$ allocation

Supporting Functions

- **can_assign(course, slot, room, timetable)**: Checks if the assignment satisfies constraints like availability and capacity.

- **calculate_score(timetable)**: Evaluates the timetable's efficiency based on conflict minimization and preferences.
- **is_slot_free(assignment)**: Verifies that no other assignment occupies the same slot.
- **is_room_capacity_ok(assignment)**: Ensures the room can accommodate the assigned group.
- **is_faculty_available(assignment)**: Checks if the faculty is available at the assigned time.

These algorithms ensure the timetable is conflict-free, efficient, and optimized for the needs of both students and faculty while being easily extendable for future requirements.

6 Coding/Implementation

This section details the technology stack and implementation choices for the "Automated Time-table Scheduling" project. The initial development focuses on setting up the project structure, data parsing, and a foundational scheduling logic.

6.1 Technology Stack

The project will be developed using the following technologies:

- **Python 3.13:** Chosen for its extensive libraries, particularly for data handling (Pandas) and constraint programming (Google OR-Tools), and its rapid development capabilities.
- **CSV/JSON:** Utilized as the primary format for input data, defining courses, faculty, rooms, and constraints. This format is human-readable and easy to manage.
- **pytest:** A robust testing framework used to implement unit tests. This ensures the reliability and correctness of individual components, such as data parsing and constraint validation.
- **Git and GitHub:** Employed for version control and collaborative development. This allows for effective team coordination, code management, and tracking project history. GitHub Link : https://github.com/shriyanssahoo/Byte_Me

7 Conclusion

The proposed Automated Timetable System aims to overcome the limitations of the existing slot-based timetable by providing a more efficient, user-friendly, and personalized solution. By eliminating the need for students to manually cross-reference slot codes with course lists, the system will significantly reduce the time and effort required to check schedules.

With features such as direct, individualized timetables, conflict-free scheduling, quick updates, and mobile-friendly access, the new system will streamline timetable management for both students and faculty. It will also improve communication, adaptability to sudden changes, and overall academic organization within the college.

Ultimately, this solution will not only save time but also enhance the academic experience, ensuring that timetable access is instant, accurate, and hassle-free [3].

References

[1] Dr. Vivekraj V K. 2025 AD CSE TT, 2025. URL https://docs.google.com/spreadsheets/d/1OFJvLCslq0ALMjQzeHuc1PcpFqDh_wGf/edit?rtpof=true&sd=true&gid=366542899.

[2] Rajib Mall. *Fundamentals of Software Engineering*. PHI Learning Pvt. Ltd., 2018.

[3] OpenAI. ChatGPT, 2025. URL https://chatgpt.com.

[4] Overleaf. Documentation - Overleaf, 2025. URL https://www.overleaf.com/learn.

[5] StarUML. StarUML, 2025. URL https://docs.staruml.io.