Detailed Project Report

on

Automated Timetable scheduling for IIIT Dharwad

Submitted by

Team: Semicolon

Shivam Kishore 24BCS140

Ritik Sinha 24BCS121

Mayank Sahu 24BCS126

P Maruthi 24BCS098

Under the guidance of

Dr. Vivekraj VK

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

12/08/2024

Contents

List of Figures **iii**

1 Introduction **1**

2 Existing System **4**

2.1 Present Semester Timetable 5

2.2 Previous Semester Timetables 6

2.3 Limitations of the Manual Approach 7

3 Requirements Modeling **13**

3.1 List of Requirements 13

4 Software Design **22**

5 Low Level Design **25**

6 Module **29**

7 Introduction to Data Dictionary **29**

8 Standard Notations in Data Dictionaries **29**

9 Application to the Time-Table Diagrams **30**

10 Coding / Implementation **32**

10.1 Programming Language 32

10.2 Frameworks and Libraries 32

10.3 Database 32

10.4 Front-End (Optional Extension) 33

10.5 Version Control and Collaboration 33

10.6 Testing and Quality Assurance 33

10.7 Deployment 33

10.8 Summary 33

11 Conclusion **35**
11.1 Final Thoughts 35

References **36**

List of Figures

1 Present semester time table 5
2 Previous semester time table 6
3 Previous semester seating arrangement 11
4 Previous semester supplementary seating arrangement 12
5 Use case diagram. 21
6 DFD Level 0 (Context Diagram) 22
7 DFD Level 1 23
8 DFD Level 2 24
9 Module 29

List of Tables

1. List of functional requirements 9

2. List of non-functional requirements 11

1 Introduction

Creating an efficient timetable in a modern academic setting is a challenging process that requires balancing multiple constraints while optimizing resource usage. At IIIT Dharwad, with a growing variety of courses, electives, and laboratory work, manual timetable creation is often time-consuming and prone to conflicts. An automated timetable system can solve these challenges by providing an organized, flexible, and adaptive scheduling framework for both students and faculty members.

The proposed system makes use of intelligent algorithms, real-time data, and modern technologies such as vision-based recognition and QR code access to streamline timetable generation and management. By considering academic rules, faculty preferences, and student availability, it delivers a conflict-free and visually accessible timetable. Provide a detailed introduction to the problem describing the purpose. Describe the constraints with which the system has to function.

- Tracking availability of professors, classrooms, laboratories, lab assistants, students registered for a course, and individual students.
- Taking user inputs for:

  Course Code: e.g., CS101

  Course Name: e.g., Data Structures and Algorithms
- LTPSC Structure: e.g., 3-1-2-0-4 (3 Lectures, 1 Tutorial, 2 Practical hours,

0 Self-study, 4 Credits) or 2-0-3-0-3 (2 Lectures, 0 Tutorials, 3 Practical hours, 0 Self-study, 3 Credits)

Course Instructor: e.g., Dr. A. Kumar

Course Offered To: e.g., B.Tech CSE, Semester 3

- Allocating dedicated study hours for better preparation time between lectures.
- Avoiding continuous classes for professors, ensuring a minimum 3-hour gap between any two lectures.
- Restricting one lecture/tutorial per day per course for balanced learning.
- Dynamic rescheduling in case of faculty absence, public holidays, or student unavailability.
- scheduling lunch breaks within the working hours of the mess to avoid clashes.
- Color-coded visual timetables for quick recognition (e.g., green for core courses, blue for labs, orange for electives).
- Scheduling labs on the same day as the related lecture to improve retention and application.
- With the automation of class timetables, the need for a parallel system to automate exam timetables has emerged, ensuring consistency and reducing manual workload for the exam section.

- The system will also manage classroom allocation and seating arrangements, ensuring fairness by preventing two students taking the same exam from occupying the same desk.
- Automatic syncing with Google Calendar for reminders and easy mobile access.
- Special handling of half-semester courses so they do not overlap with regular full-semester ones.
- Ensuring elective and minor courses share the same slot across batches to prevent conflicts.
- Vision-based automation examples: detecting empty classrooms via CCTV to reallocate sessions instantly.
- QR code features: students can scan to view their updated personal timetable after any change.
- AI-powered suggestions: recommending optimal lab batches based on past attendance patterns.
- eather-based rescheduling: shifting physical classes to online mode during heavy rain alerts.

2 Existing System

Present Manual Approach to Timetable Scheduling at IIIT Dharwad Timetable scheduling at IIIT Dharwad is currently carried out using a manual, slot-based system with the help of spreadsheets and static timetable templates. While this method provides a structured framework, it is still largely dependent on human coordination and adjustments, making it less adaptive to dynamic changes. Steps in the Current Manual Scheduling Process

- Course Information Collection – At the start of each semester, details such as course code, course name, LTPSC structure (Lecture, Tutorial, Practical, Semester, Credits), course instructor, and the batches or programs for which the course is offered are collected.
- Slot-Based Allocation – Fixed time slots (A1, B1, C1, etc. for lectures; L1–L15 for labs; A1-T, B2-T, etc. for tutorials) are predefined. Courses are manually assigned to these slots based on instructor availability and department requirements.
- Resource Mapping – Classrooms, labs, and equipment are assigned manually based on expected student strength and type of class (lecture, tutorial, or practical).
- Clash Checking – Coordinators manually check for conflicts in faculty schedules, student electives, and lab usage.
- Timetable Finalization – A final timetable is created in spreadsheet or image format, which is then circulated to faculty and students via email or notice boards.

2.1 Present Semester Timetable

- Lecture Slots: A1–E2, U, Z, etc. are each 1.5 hours long.
- Tutorial Slots: (e.g., D1-T, B3-T) are 1-hour sessions used for problem-solving or discussion.
- Lab Slots: L1–L15 are 2-hour practical sessions.
- Minor Slots: Early morning (07:30–09:00) and late evening (18:30 onwards) are reserved for minor courses.
- Color-Coding: Each slot is color-coded for quick identification.
- Example: b On Monday, A1 (09:00–10:00) is followed by L1 (10:00–12:00), and later by D1-T (12:15–12:30), with E2 scheduled in the afternoon.

Figure 1. Present semester time table

2.2 Previous Semester Timetables

- Slot Reassignment: Courses were often shuffled across slots to balance faculty workloads.
- Lab Distribution: Earlier labs were spread across multiple days, while recent timetables try to keep labs on the same day as the lecture.
- Elective Scheduling: In previous semesters, electives were scattered across different time slots, causing clashes. Now they are grouped into common slots.
- Half-Semester Courses: In earlier timetables, these were sometimes placed without clear separation, whereas now they are systematically scheduled.

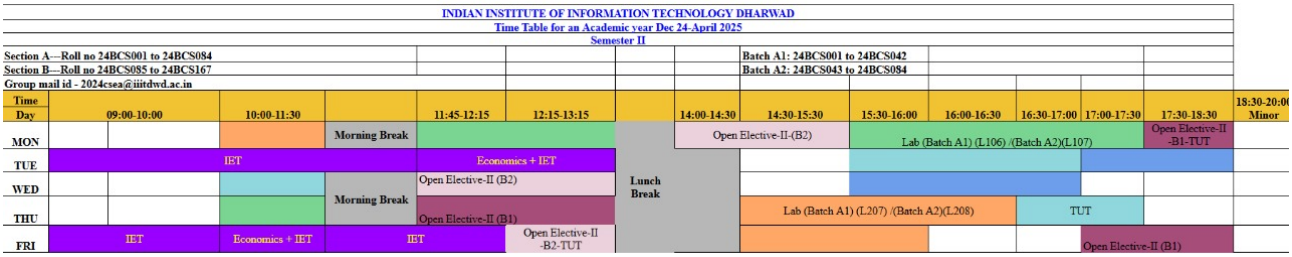

Figure 2. Previous semester time table

2.3 Limitations of the Manual Approach

- Time-Consuming: Preparing and finalizing the timetable requires multiple iterations.
- Error-Prone: Manual clash detection can overlook scheduling conflicts.
- Rigid Structure: Making mid-semester changes is difficult without disrupting other classes.
- Lack of Automation: No integration with personal calendars or automatic notifications.

5. Comparison of Previous (Manual) and Proposed (Automated) System

The following table highlights the differences between the previously followed manual scheduling system and the proposed automated timetable scheduling system.

Aspect	Manual System	Automated System
Class Timetable	• Created manually by academic section. • Prone to human errors and conflicts. • Multiple revisions and approvals required.	• Generated automatically by system. • Faculty and classroom availability checked. • Updates quickly without redoing full schedule.

Exam Timetable	• Planned manually based on course registrations. • Students often faced back-to-back or clashing exams. • Heavy administrative workload.	• Generated automatically using academic calendar. • Avoids back-to-back clashes. • Balances exam load across different days.
Seating Arrangement	• Assigned manually by invigilators/staff. • No automated check to separate same-exam students. • Relied on printed charts and manual cross-checking.	• Plans generated automatically. • Ensures no two students of same exam share a desk. • Provides printable seating charts and attendance sheets.

Overall	• Time-consuming and inefficient. • High risk of errors or bias. • Poor adaptability when changes occur.	• Saves time and reduces workload. • Minimizes human error and ensures fairness. • Scalable, flexible, and efficient for long-term use.

Room C101

WINDOW

Date 19/04/2025

session FN

COL1		COL2		COL3	
23BCS001	23BDS001	23BCS009	23BDS009	23BCS018	23BDS017
23BCS002	23BDS002	23BCS010	23BDS010	23BCS019	23BDS018
23BCS003	23BDS003	23BCS011	23BDS011	23BCS020	23BDS019
23BCS004	23BDS004	23BCS012	23BDS012	23BCS021	23BDS021
23BCS005	23BDS005	23BCS013	23BDS013	23BCS022	23BDS022
23BCS006	23BDS006	23BCS015	23BDS014	23BCS023	23BDS023
23BCS007	23BDS007	23BCS016	23BDS015	23BCS024	23BDS024
23BCS008	23BDS008	23BCS017	23BDS016	23BCS025	23BDS025

Door

Room C102

WINDOW

Date 19/04/2025

session FN

COL1		COL2		COL3	
23BCS026	23BDS026	23BCS034	23BDS034	23BCS042	23BDS042
23BCS027	23BDS027	23BCS035	23BDS035	23BCS043	23BDS043
23BCS028	23BDS028	23BCS036	23BDS036	23BCS044	23BDS044
23BCS029	23BDS029	23BCS037	23BDS037	23BCS045	23BDS045
23BCS030	23BDS030	23BCS038	23BDS038	23BCS046	23BDS046
23BCS031	23BDS031	23BCS039	23BDS039	23BCS047	23BDS047
23BCS032	23BDS032	23BCS040	23BDS040	23BCS048	23BDS048
23BCS033	23BDS033	23BCS041	23BDS041	23BCS049	23BDS049

Door

Figure 3. Previous semester seating arrangement

Indian Institute of Information Technology Dharwad
Supplementary Exams May 2025

Session	FN: 10:00 AM to 01:00 PM	
Date	19-May-2025	20-May-2025
Day	Monday	Tuesday
Course Code	PH105	CS105
		CS161
		CS163
		CS165
		CS201
		CS202
		CS204
		CS207
		CS307
		CS310
		CS352
		DS151
		DS161
		DS301
		EC105
		EC204
		EC205
		EC272
		HS102
		HS153
		HS156
		MA163

S.no.	code	course
1	CS105	Problem Solving through Programming
2	CS161	Problem Solving through Programming
3	CS163	Data Structures and Algorithms
4	CS164	Computer Architecture
5	CS165	Mathematical Foundations of Computi
6	CS201	Discrete Mathematics
7	CS202	Design & Analysis of Algorithms
8	CS204	Operating Systems
9	CS206	Theory Of Computing
10	CS207	Object Oriented Programming
11	CS208	Computer Architecture
12	CS301	Software Engineering
13	CS307	Machine Learning
14	CS309	Statistics for Computer Science
15	CS310	Database Management System
16	CS352	Cryptography & Information Security
17	DS151	Linux for Engineers
18	DS161	Introduction to DS and AI
19	DS164	Data Curation Techniques
20	DS301	Graphs and Social Networks
21	EC105	Computer Architecture
22	EC154	Introduction to Digital VLSI Design
23	EC204	Analog & Digital
24	EC205	Control Systems
25	EC272	VLSI Physical Design
26	EC301	Introduction to VLSI Design
27	EG102	Basic Circuit Theory
28	HS102	Professional Communication

Figure 4. Previous semester supplementary seating arrangement

3 Requirements Modeling

3.1 List of Requirements

Requirement	Description
Functional Requirements	
Resource Availability Tracking	• Maintain database of professors, classrooms, laboratories, lab assistants, and students. • Update availability in real-time based on schedules or events. • Allow manual overrides by administrators.

Course Information Input	• Accept course details: Code, Name, LTPSC structure(explained above), Instructor, Target Audience. • Validate input before saving. • Class fuctional time is 9AM to 5PM and from Monday to Saturday. • In case of holidays , manual updation.
Class Allocation Logic	• Ensure professors have at least a 3-hour gap between lectures. • Avoid back-to-back classes for the same professor. • Allow only one lecture/tutorial per course per day. • Schedule electives and minors in overlapping slots.

Lab Scheduling Rules	• Labs should be on the same day as the related lecture. • Prevent overlapping labs needing the same resources.
Student & Professor Conflict Resolution	• Detect and resolve timetable clashes automatically. • Allow rescheduling based on availability.
Lunch Break Management	• Schedule lunch breaks within mess operating hours. • Avoid any kind of lectures/lab during lunch periods.

Half-Semester & Special Courses	• Separate scheduling logic for half-semester courses. • Allocate project/workshop sessions without affecting regular classes.
Automated Exam Timetable & Seating Arrangement	• Generate exam timetables automatically along with class timetables. • Assign classrooms dynamically based on student strength and course requirements. • Implement seating arrangement logic to ensure that no two students taking the same exam are seated on the same desk. • Allow administrators to override classroom or seating allocations if needed.

Exam Management Quality Constraints	• Ensure fairness in seating allocation to prevent malpractice. • Guarantee quick timetable generation within seconds, even for large batches. • Provide error-free assignments with conflict detection for overlapping exams. • Maintain data privacy and security of exam-related information.
Color-Coded Visual Timetable	• Assign distinct colors for each course. • Include legend for clarity.

Integration with External Platforms	• Export timetable to Google Calendar. • Send automated email/app notifications for changes.
User Roles & Permissions	• Admin: Create/edit timetable. • Professor: View timetable, request changes. • Student: View timetable for their batch/electives.
Non-Functional Requirements	
Performance	• Generate a complete timetable within 2–5 seconds.
Scalability	• Handle 200+ courses, 100+ professors, and 2000+ students without performance issues.

Usability	• Provide intuitive drag-and-drop UI for admins. • Support weekly and daily views.
Reliability	• Daily backup of timetable data. • Ensure 99.5% uptime.
Security	• Role-based access control. • Log all changes for auditing.
Maintainability	• Modular code structure for adding new rules easily.

Compatibility	• Support major browsers (Chrome, Firefox, Edge). • Mobile-friendly responsive design.

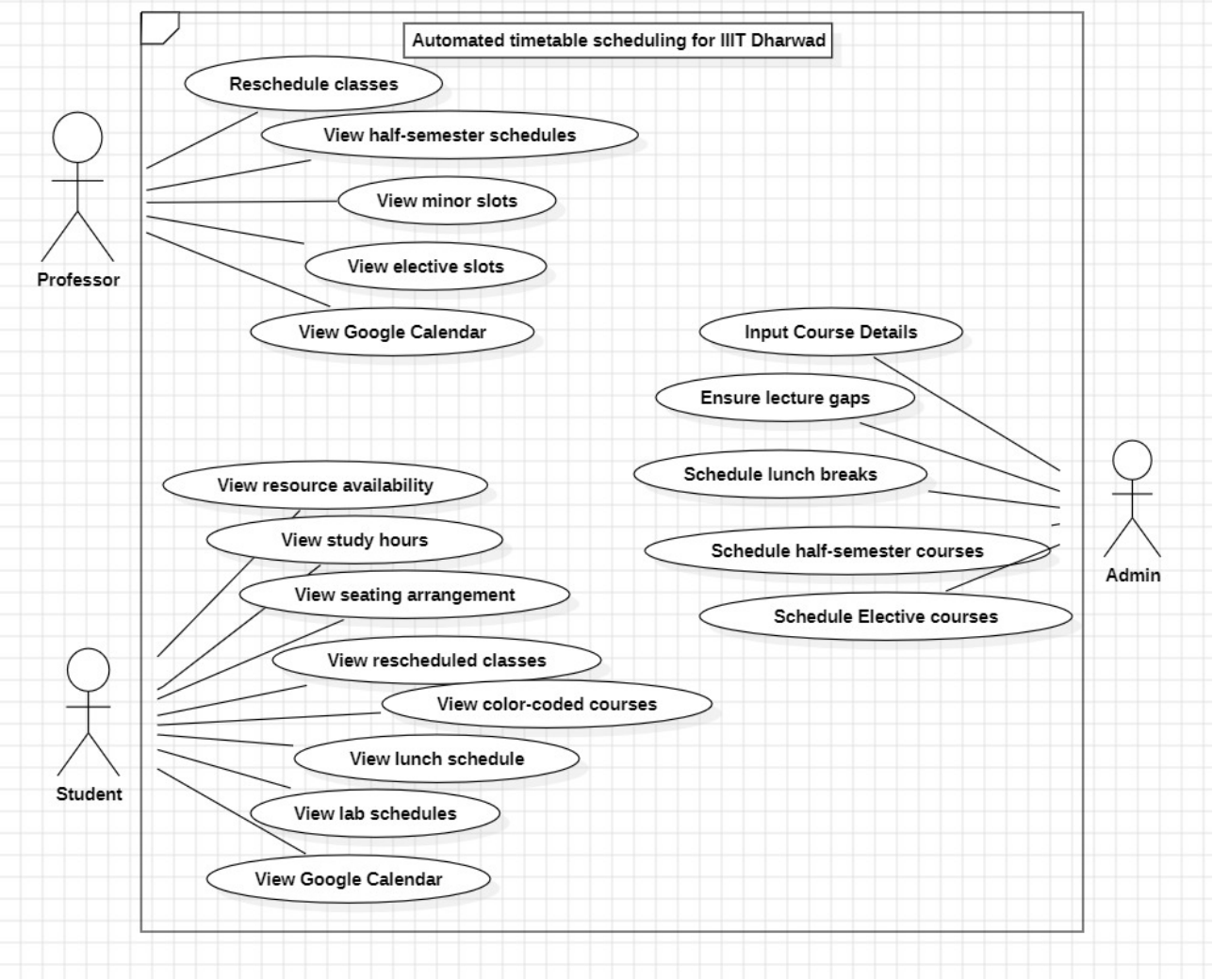

Figure 5. Use case diagram.

4 Software Design

In this section, we present the software design aspects of the system. One of the most effective tools for representing the flow of information and processes is the **Data Flow Diagram (DFD)**. A DFD visually depicts how data moves through the system, showing inputs, processes, storage, and outputs. It helps in understanding both the logical flow and functional decomposition of the system. Typically, a DFD is created at multiple levels: starting with a high-level context diagram (Level 0) and progressively breaking down into more detailed diagrams (Level 1, Level 2, etc.).

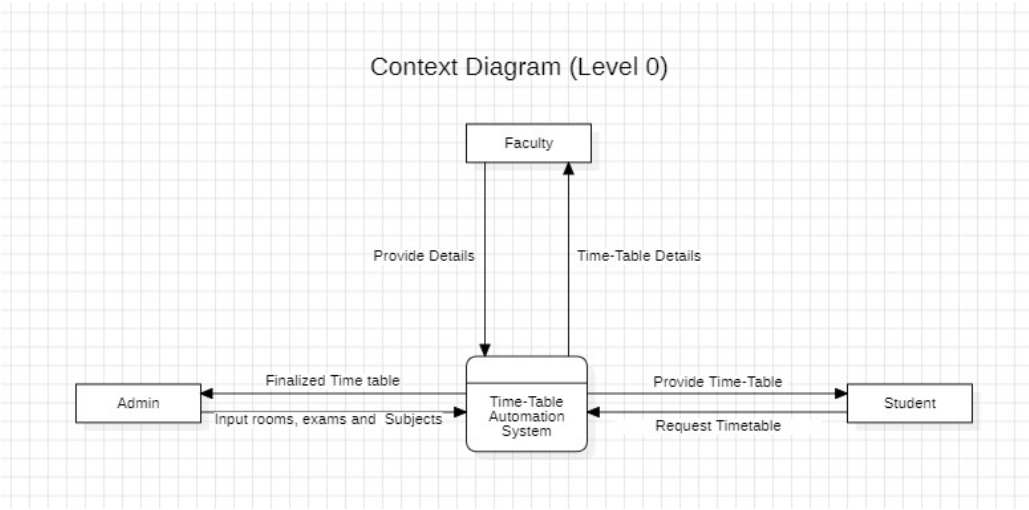

Figure 6. DFD Level 0 (Context Diagram)

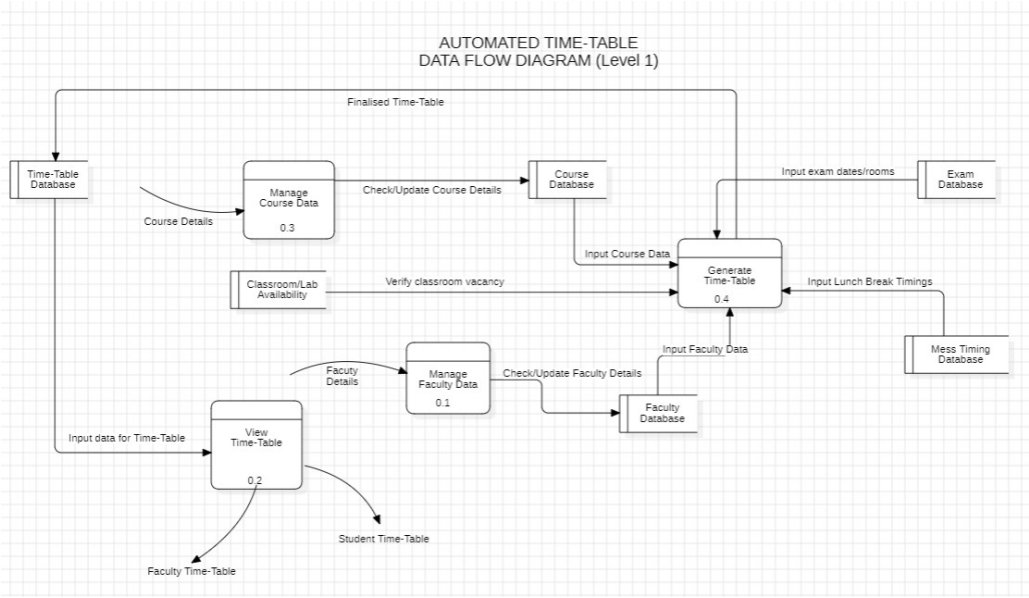

Figure 7. DFD Level 1

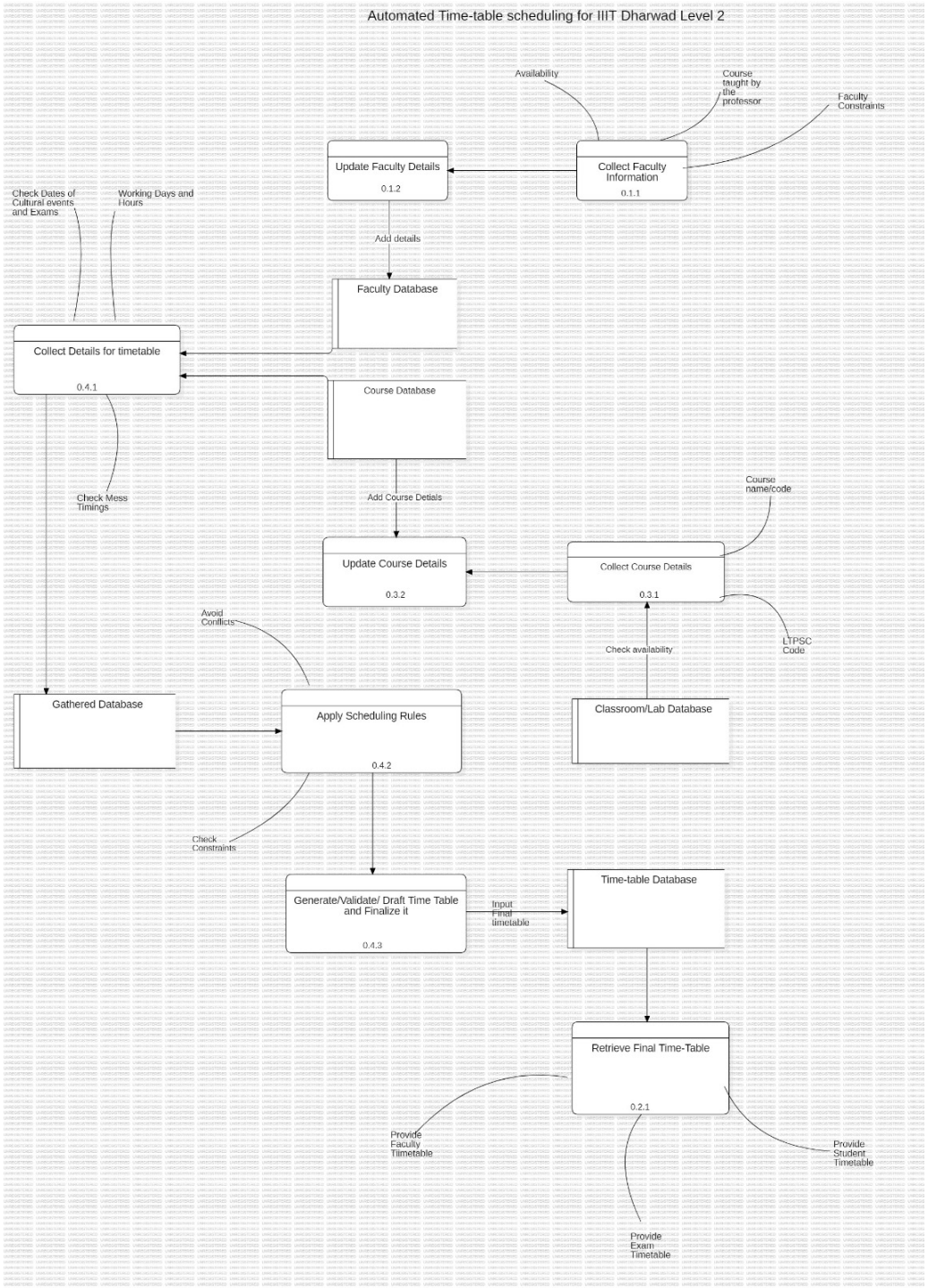

Figure 8. DFD Level 2

5 Low Level Design

Data Structures

```
struct Faculty {
    int facultyID;
    string name;
    vector<string> courses;      // Courses taught
    vector<string> availableSlots; // Time slots available
};
```

Listing 1: Faculty Information

```
struct Course {
    string courseCode;
    string courseName;
    int credits;
    string facultyID;
    vector<string> preferredSlots;
};
```

Listing 2: Course Information

```
struct Room {
    string roomID;
    int capacity;
    bool isLab;
    vector<string> availableSlots;
};
```

Listing 3: Classroom/Lab Information

```
struct Exam {
    string courseCode;
    string examDate;
    string roomID;
};
```

Listing 4: Exam Information

```
struct TimeTableEntry {
    string day;
    string timeSlot;
    string courseCode;
    string facultyID;
    string roomID;
};
```

Listing 5: Timetable Entry

```
struct TimeTable {
    vector<TimeTableEntry> entries;
};
```

Listing 6: Timetable

Function Declarations and Descriptions

1. Admin Module

```
void inputSubjects(vector<Course>& courseList);
void inputExams(vector<Exam>& examList);
void inputRooms(vector<Room>& roomList);
void finalizeTimeTable(TimeTable& tt);
```

2. Faculty Module

```
void provideAvailability(Faculty& f, vector<string> slots);
void viewFacultyTimeTable(Faculty f, TimeTable tt);
```

3. Student Module

```
void requestTimeTable(string studentID);
void viewStudentTimeTable(string studentID, TimeTable tt);
```

4. Course Management Module

```
void collectCourseDetails(Course& c);
void updateCourseDetails(Course& c, string newName, int newCredits);
```

5. Faculty Management Module

```
void collectFacultyInfo(Faculty& f);
void updateFacultyDetails(Faculty& f, vector<string> newCourses);
```

6. Time-Table Generation Module

```
void collectDetails(vector<Course> courses, vector<Faculty> faculties, vector<Room> rooms);
void applySchedulingRules(TimeTable& tt, vector<Course> courses, vector<Faculty> faculties,
    vector<Room> rooms);
void generateTimeTable(TimeTable& tt, vector<Course> courses, vector<Faculty> faculties,
    vector<Room> rooms, vector<Exam> exams);
bool validateTimeTable(TimeTable tt);
```

7. Database Management Module

```
void saveCourseDB(vector<Course> courses);
vector<Course> loadCourseDB();

void saveFacultyDB(vector<Faculty> faculties);
```

```
vector<Faculty> loadFacultyDB();

void saveExamDB(vector<Exam> exams);
vector<Exam> loadExamDB();

void saveRoomDB(vector<Room> rooms);
vector<Room> loadRoomDB();

void saveTimeTableDB(TimeTable tt);
TimeTable loadTimeTableDB();
```

8. View/Retrieve Timetable Module

```
TimeTable retrieveFinalTimeTable();
void viewTimeTable(TimeTable tt, string userType, string userID);
```

6 Module

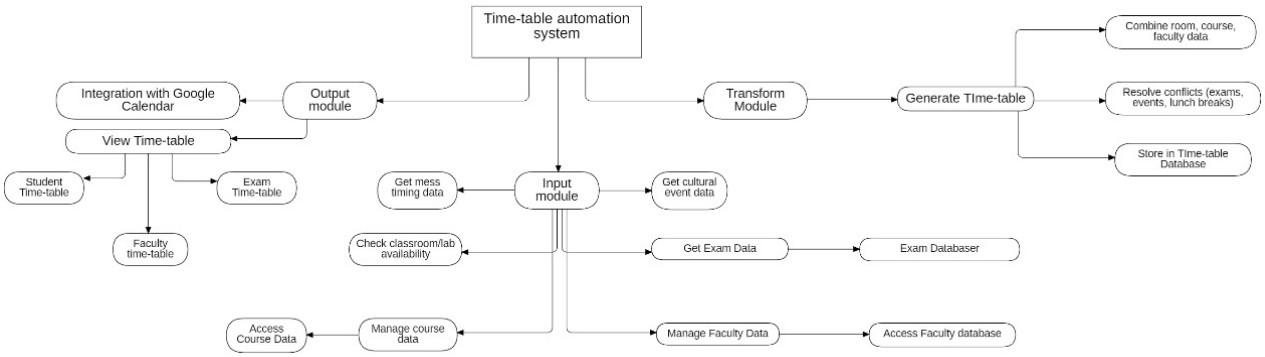

Figure 9. Module

7 Introduction to Data Dictionary

A *data dictionary* is a centralized repository that stores metadata—that is, information about the data used by a system. It describes elements such as names, definitions, data types, lengths, constraints, formats, and relationships, and may also document usage, ownership, and update frequency. A well-constructed data dictionary enhances consistency, clarity, and effective data governance across software systems.

8 Standard Notations in Data Dictionaries

In structured analysis and data modeling, several notations are commonly used to define data constructs and their relationships:

- **Composition:** $X = a + b$ indicates that X is composed of data elements a and b.

- **Selection (OR):** $X = [a \mid b]$ means X consists of either a or b.
- **Sequence (AND):** $X = a\ b$ or use of $+$ indicates both a and b are required—sequentially.
- **Repetition:**
  - $X = y[a]$ indicates y or more occurrences of a.
  - $X = [a]z$ indicates up to z occurrences of a.
  - $X = y[a]z$ indicates between y and z occurrences.

  These notations help precisely define data structures encountered in Diagrams and Data Dictionaries. :contentReference[oaicite:1]index=1

9 Application to the Time-Table Diagrams

In the context of the Time-Table Automation System diagrams (such as DFD-like diagrams you provided), the standard notations map naturally onto the graphical symbols as follows:

- **Rectangles (External Entities):** Entities like `Student`, `Faculty`, and `Admin` represent sources or sinks of data—e.g., *Student requests timetable*, *Admin inputs subjects*.
- **Rounded Rectangles (Processes):** Actions such as *Generate Timetable* or *Collect Input Data* correspond to operations on one or more data elements—often combining (composition, $+$), selecting options (e.g., choose either class or exam timetable), or sequencing tasks.

- **Cylinders (Databases / Data Stores):** Examples include `Class and Subject DB`, `Exam Schedule DB`, and `Room and Resource DB`. These components serve as repositories for structured data elements, potentially containing repetitions (e.g., multiple subjects, multiple time slots).
- **Arrows (Data Flow):** Indicate the direction and movement of data—such as "faculty availability" flowing into the data dictionary or "class timetable" flowing out to the student. If multiple items (e.g., subjects + rooms + capacity) are sent together, you might interpret that as composition; if selection or conditional routing is implied (e.g., exam or class data), that maps to the selection notation.

By relating each graphical symbol back to the underlying data constructs defined via these notations, the Data Dictionary can ensure consistency and clarity in documentation, design, and implementation of your Time-Table Automation System.

10 Coding / Implementation

The implementation of the **Automated Timetable Scheduling System** will be carried out using a modern, modular, and scalable technology stack. The key components of the tech stack are as follows:

10.1 Programming Language

- **Python 3.12+** — Chosen for its simplicity, wide community support, and availability of libraries for scheduling, optimization, and data management.

10.2 Frameworks and Libraries

- **Flask / FastAPI** — To build REST APIs for timetable generation and exam scheduling modules.
- **Pandas & NumPy** — For efficient handling and manipulation of course, faculty, and timetable datasets.
- **SQLAlchemy** — ORM for managing persistent data (faculty, courses, rooms).
- **Matplotlib / Plotly** — For generating color-coded visual timetables.
- **Google API Client** — For integration with Google Calendar and notifications.

10.3 Database

- **PostgreSQL / SQLite** — Relational database to store courses, faculty, rooms, exams, and generated timetables.

- Provides reliability, scalability, and ACID-compliant transactions.

10.4 Front-End (Optional Extension)

- **React.js with TailwindCSS** — For a responsive, intuitive UI enabling admin, faculty, and student interactions.
- **FullCalendar.js** — For displaying weekly/daily timetable views with drag-and-drop features for admins.

10.5 Version Control and Collaboration

- **Git & GitHub** — For version control, collaborative development, and CI/CD.

10.6 Testing and Quality Assurance

- **PyTest / Unittest** — For writing automated unit tests and integration tests.
- **Coverage.py** — To ensure code coverage and maintain high reliability.

10.7 Deployment

- **Docker** — Containerization for consistent deployment across environments.
- **Heroku / AWS EC2** — Potential platforms for hosting the system.

10.8 Summary

The chosen technology stack ensures that the system is:

- Scalable to handle 200+ courses, 100+ faculty, and 2000+ students.
- Maintainable due to modular code design.
- Secure with role-based access and database protection.
- User-friendly with both command-line and web-based access points.

11 Conclusion

11.1 Final Thoughts

The Automated Timetable Scheduling System is designed to address the persistent challenges faced by academic institutions in planning and managing schedules. By automating complex allocation processes, it not only saves time but also ensures fairness, accuracy, and compliance with institutional policies. The system's adaptability allows it to accommodate various academic structures, course patterns, and resource constraints, making it suitable for long-term use. With its focus on efficiency, user experience, and scalability, the project has the potential to become an essential tool for modern academic administration.

References

[1] Goktug, A. N., Chai, S. C., & Chen, T. (2013). A timetable organizer for the planning and implementation of screenings in manual or semi-automation mode. *Journal of Biomolecular Screening*, 18(8), 938–942. doi:10.1177/1087057113493720

[2] Shah, M., Patel, K., & Bhatt, C. (2018). Automated timetable generation using genetic algorithm. *International Journal of Computer Applications*, 182(18), 1–5. doi:10.5120/ijca2018917461