Detailed Project Report

on

Automation Of Time-Table For IIIT Dharwad

Submitted by

Team: *BumbleBee*

Raskin Verma 24BCS117, Vishwa D 24BCS163

Sindhu Talari 24BCS153, Udit Dadhich 24BCS158

Under the guidance of

Vivekraj VK

Professor of SDTT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

12/08/2025

Contents

List of Figures

List of Tables

1 Introduction

Here the problem we are trying to solve is automating the time table making process. A time table is needed as it organizes and structures the academic schedule which benefits the students and teachers in multiple ways.

In order to solve this problem we will need the proper documentation which contains all the requirements such as course code, instructor timings, working days, lunch timings, class availability and many more. It's much like managing a sports league scheduled, where you have to balance venues, teams and refrees to ensure everything fits together perfectly.

1.1 Constraints on the system

When creating an automated timetable system, the following constraints must be considered to ensure that the generated schedule is practical and efficient:

- Instructor Availability: Each instructor has specific workdays and hours. The system shouldn't assign them when they're not available.
- Limitations on Classrooms and Resources: A limited number of classrooms and labs are available. The system must prevent reservations from overlapping.
- Course Requirements: A number of courses require specific equipment, like computer labs, which need to be allocated appropriately.

- Overlapping of Sessions: Neither a teacher nor a student may be enrolled in two classes at the same time.
- Working Hours and Breaks: Lectures should be planned around regular business hours, accounting for intermissions and lunch breaks.
- Balanced Workload: The system should distribute lectures evenly throughout the week to avoid overloading instructors or students on a single day.
- Priority Rules: Certain courses (like core subjects) may need to be given more weight than electives when allocating slots.
- Semester/Batch Dependencies: Students should not be enrolled in classes that conflict with required subjects during the same semester.

1.2 Available Infrastructure

The automated timetable system will operate within the existing institutional infrastructure:

- Classrooms should be equipped with enough seating and basic teaching facilities.
- Computer and electronic labs are required for certain courses.
- List of professors along with their preferred teaching slots.
- Google calendar to be integrated with the timetable for easy access.
- Institutional servers and computers that can host the timetable generation system.

2 Existing System

- What we currently have here is a timetable in which all the courses have a predefined slot.
- This time table here is manually created with the help of tools such as Google Sheets or Excel.
- The blocks are labeled with codes (e.g., A1, L1, D1-T, X, Z, U, etc.), each corresponding to a specific subject, course component, lab session, or tutorial.
- Blocks are color coded to easily differentiate between courses.
- The timetable also includes slots for breaks to allow students and faculty to rest or transition between classes.
- Early morning and evening slots are reserved for extra minor or major courses.

2.1 Present Time Table

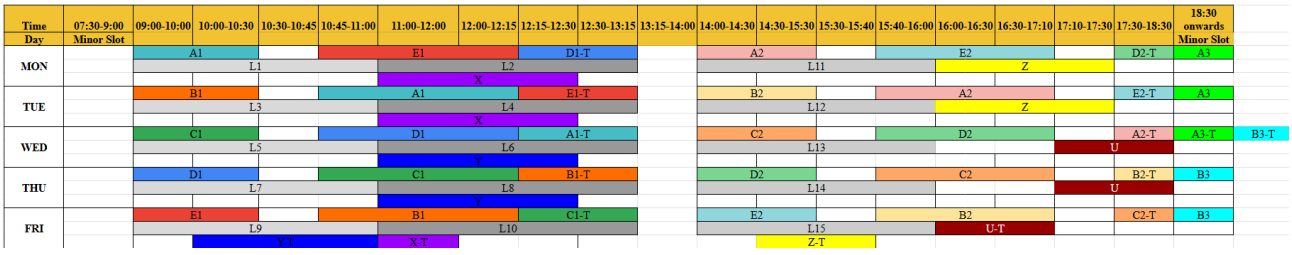

Figure 1. Present Academic Timetable

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 2. Present Semester Course Details

2.2 Previous Academic Time Table

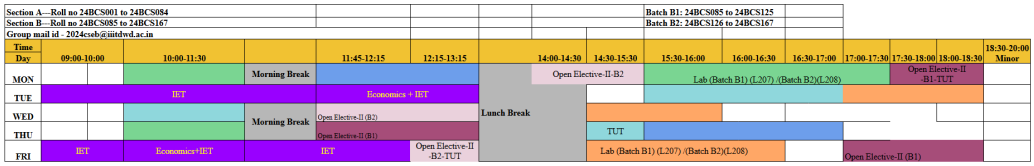

Figure 3. Previous Academic TimeTable

Sl No.	Course Code	Course Title	Credits (L-T-P-S-C)	Faculty	Lab assistance	Room No.
1	CS162	Optimization	3-1-0-0-2	Dr. Dibyajyothi	Ashwini Chikkenkoppa, Shrirang Pujari	C004
2	CS164	Computer architecture	3-0-2-0-2	Dr. C. B. Akki	Vinod Konnur Basavaraj Choukimath	C203
3	CS163	Data Structures & Algorithms	3-0-2-0-4	Dr. Suvadip Hazra	Sahana E Punagin Chaitra D	C203
4	HS204 / HS153	Economics/ Innovation and Entrepreneurship	2-1-0-0-3	Dr. Anushree Kini/ Dr.Deepak k T		C202/ C004
5	CS165	Mathematical foundations of Computing	3-0-0-0-3	Dr. Animesh Roy		C203
6	B1	Introduction to Design	1	Dr. Sandesh P		C204
		Introduction to Personal Finance	2	Dr. Siddharth		C202
		Introduction to Internet of Things	2	Dr Jagadeesha Bhat		C203
	B2	Concurrency and Computation	2	Dr Pramod Y		C202
		Computer Intensive Statistical Methods	2	Dr Ramesh Athe		C203
		Industry Insights Program Part 1	1	Mr. Ram Subramanian & Mr. Sasi Kumar Sundara Rajan		C204
Semester Credits				**18**		

Figure 4. Previous semester course details

Some issue and limitations with the present and previous timetable:

- The whole procedure is manual, which is time and effort consuming.
- There is a strong likelihood of scheduling conflicts such as faculty double booking.
- The use of color coding and alphabetical coding might make it a bit difficult for some students in understanding the schedule.
- Some days have too many back-to-back classes, which might lead to fatigue and less attentiveness.
- It is hard to update or modify the table as it is manually created using excel or google sheets.
- If there is any need to add more courses, it might cause problems.
- It does not follow the L-T-P-S-C structure (e.g. It shows tutorial for courses which do not have any tutorial).
- Changes in faculty schedules or room availability are not quickly reflected in the timetable.
- Students may end up with long gaps between classes, leading to wasted time during the day.
- Some instructors may have uneven teaching loads (e.g., many classes in a single day, none on others)

3 Requirements Modeling

A Use Case Diagram (UCD) is used to model the functional requirements of the system by identifying the stakeholders and the various interactions they have with the system. In the context of timetable automation, different stakeholders such as admin, faculty members, and students are directly impacted. The following diagram represents these stakeholders and their interactions with the automated time-table system.

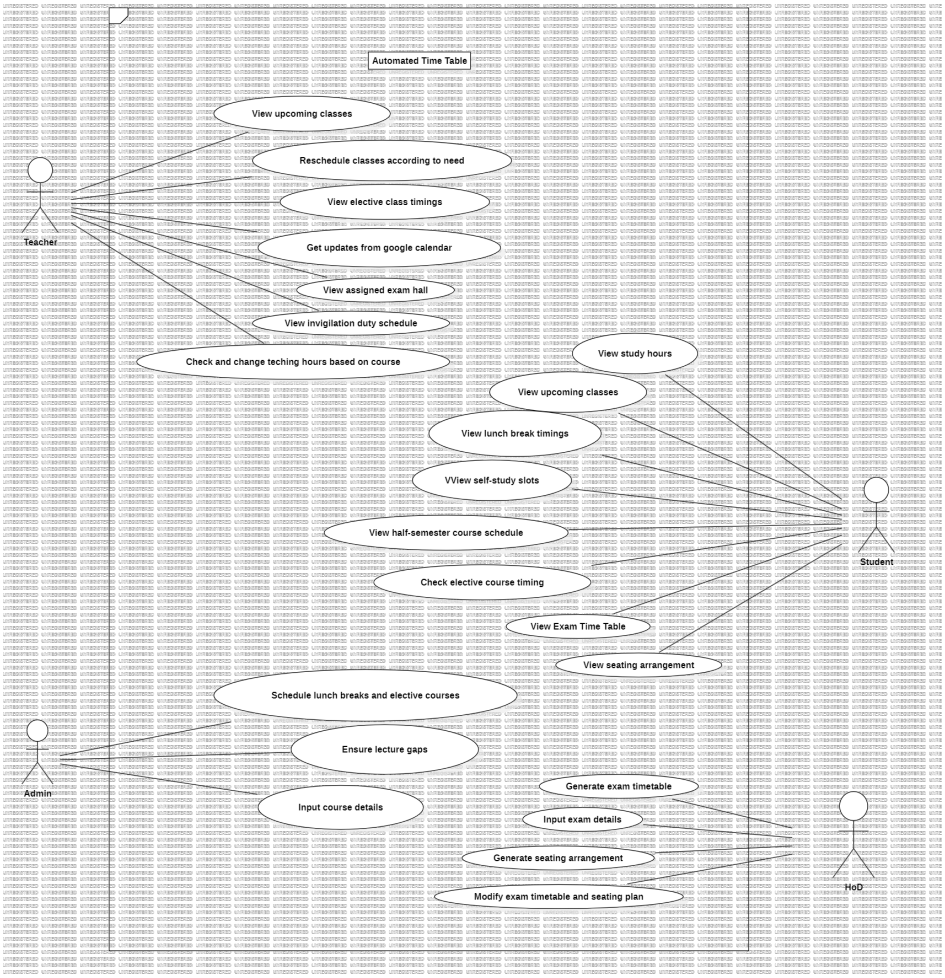

Figure 5. Use Case Diagram

3.1 List of Requirements for Classroom Time-Table

Functional Requirements

S. No.	Requirement Description
1	Follow LTPSC (L: Lectures, T: Tutorials, P: Practical, S: Self-Study, C: Credits).
2	Make sure that no classes are scheduled on Saturday and Sunday.
3	Working hours should only be between 9:00 A.M to 18:00 P.M each working day.
4	Avoid overlapping of classes.
5	Keep track of the availability of Professors, Classrooms, Labs, Students and Lab assistants.
6	Ensure that professors don't have continuous classes as it could become hectic for them. At most there should be only two consecutive classes for a professor.
7	Mandatory 10-minute breaks should be considered between consecutive lectures.
8	Lab class should be scheduled on the same day as its lecture.
9	Lunch break needs to be scheduled within the working hours of mess (12:30 P.M to 14:00 P.M).
10	Half semester courses should be appropriately scheduled.
11	All the electives should be scheduled in the same time slot.
12	It needs to be ensured that all branches are free for the elective slot as students from different branches opt for the electives.
13	Avoid different branches/years having the same classroom/lab at a time.
14	Automatically update dedicated self-study hours for each course.
15	Cultural Calendar should be studied so no event takes place on working days.

Table 1
Functional requirements for timetable automation

Non-Functional Requirements

S. No.	Requirement Description
1	Software should be integrated with Google Calendar so students get notified for every upcoming class.
2	Visual display of the Time Table should be color-coded for easy understanding.
3	The system should support 100+ concurrent users without slowing down.
4	First-time users should be able to use the system with minimal training.
5	The system must support multiple departments and campuses.
6	It should handle growth in students, courses, and classrooms.
7	Role-based access control: only authorized users can edit or view certain data.
8	Changes in policies or academic rules should be easily configurable.
9	All timetable changes must be logged and traceable.
10	System must be accessible 24/7, especially during peak scheduling periods.

Table 2
Non-functional requirements

3.2 List of Requirements for Exam Time-Table

S. No.	Requirement Description
1.	The system shall generate an automated exam timetable along with the academic timetable.
2.	The system shall automatically assign classrooms for exams based on availability and student strength.

3.	The system shall generate a seat arrangement ensuring that no two students writing the same exam are seated next to each other.
4.	The seating paln shall ensure optimal utilization of class capacity.
5.	Exams should be scheduled only when invigilators are available.
6.	Automatically notify students and faculty via portal/Google Calendar when exam schedule is published or changed.
7	Avoid conflicts for students enrolled in multiple overlapping subjects.
8	Allow for constraints like no exams on weekends, max 2 exams per day, etc.
9	Notify students and faculty of the final timetable via email/SMS.
10	Import or manage student details (name, ID, course, etc.).

Table 3
List of Requirements for Exam Time Table

Indian Institute of Information Technology Dharwad

End Sem Time table for Jan-Apr 2025 (for B.Tech. only)

Session	FN: 10:00 AM to 1:00 PM								
Date	12-Apr-2025	13-Apr-2025	15-Apr-2025	16-Apr-2025	17-Apr-2025	19-Apr-2025	20-Apr-2025	21-Apr-2025	22-Apr-2025
Day	Saturday	Sunday	Tuesday	Wednesday	Thursday	Saturday	Sunday	Monday	Tuesday
Course Code	CS204	CS206	HS205	MA202	CS307	CS310	DS355	DS358	CS472
	CS365	CS458	CS464	CS469	CS301	DS356	DS306		CS272
	CS455	DS308	EC363	EC361	HS159	EC310			
	EC272	DS204	HS206	CS462	DS309	DS351			
	HS152	EC204	CS163	MA203	DS205	HS161			
	DS164	CS164		CS165	HS204	MA163			
	EC162			CS162	DS151				
					CS152				
					EC154				

Session	AN: 3:00 PM to 6:00 PM								
Date	12-Apr-2025	13-Apr-2025	15-Apr-2025	16-Apr-2025	17-Apr-2025	19-Apr-2025	20-Apr-2025	21-Apr-2025	22-Apr-2025
Day	Saturday	Sunday	Tuesday	Wednesday	Thursday	Saturday	Sunday	Monday	Tuesday
Course Code	EC205		DS357			EC307		PH352	
								IC279	

Figure 6. Previous Exam Time-Table

4 Software Design

4.1 Data Flow Diagram

4.1.1 Level - 0

These are the Data flow Diagrams of our model/software

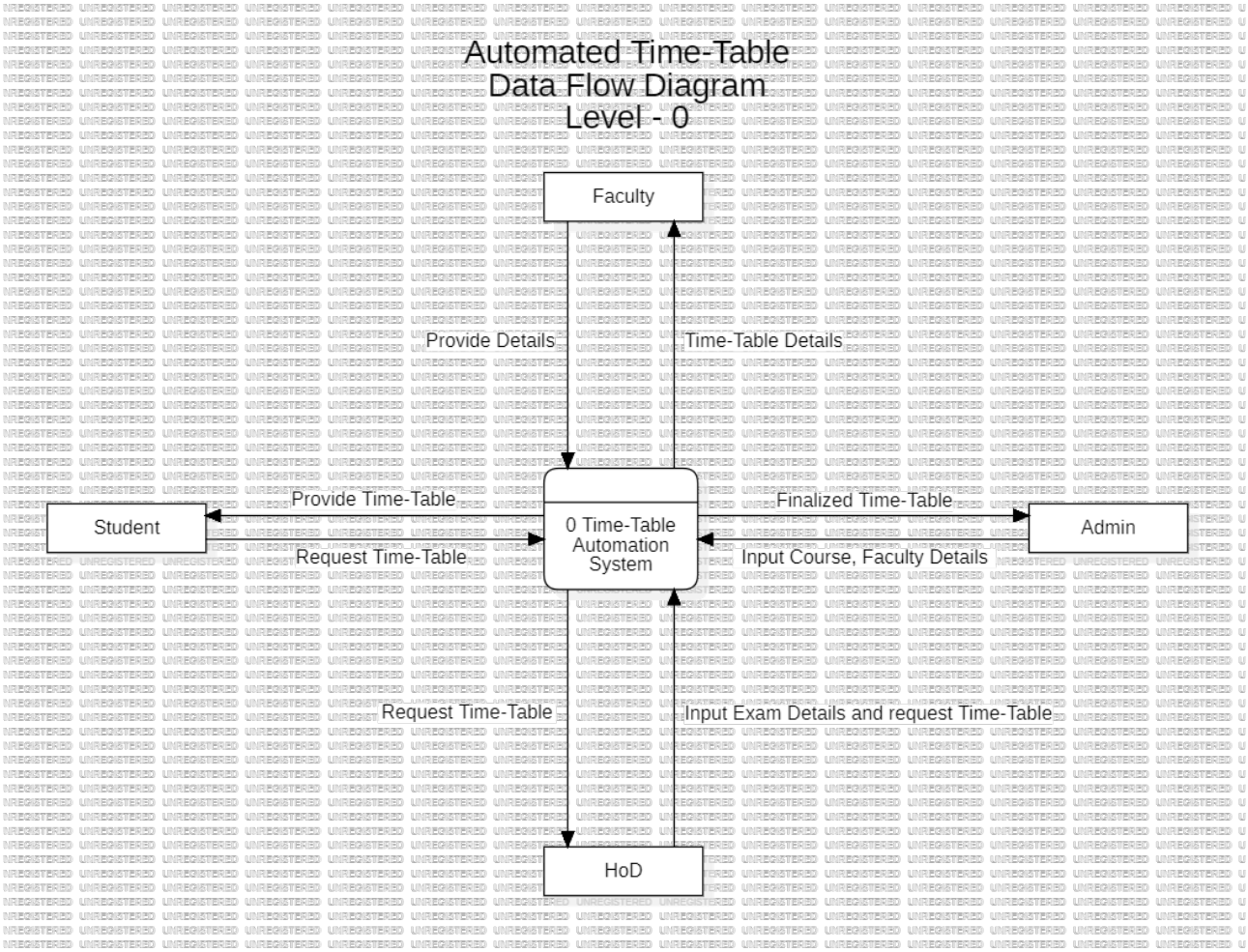

Figure 7. Level-0 DFD

4.1.2 Level - 1

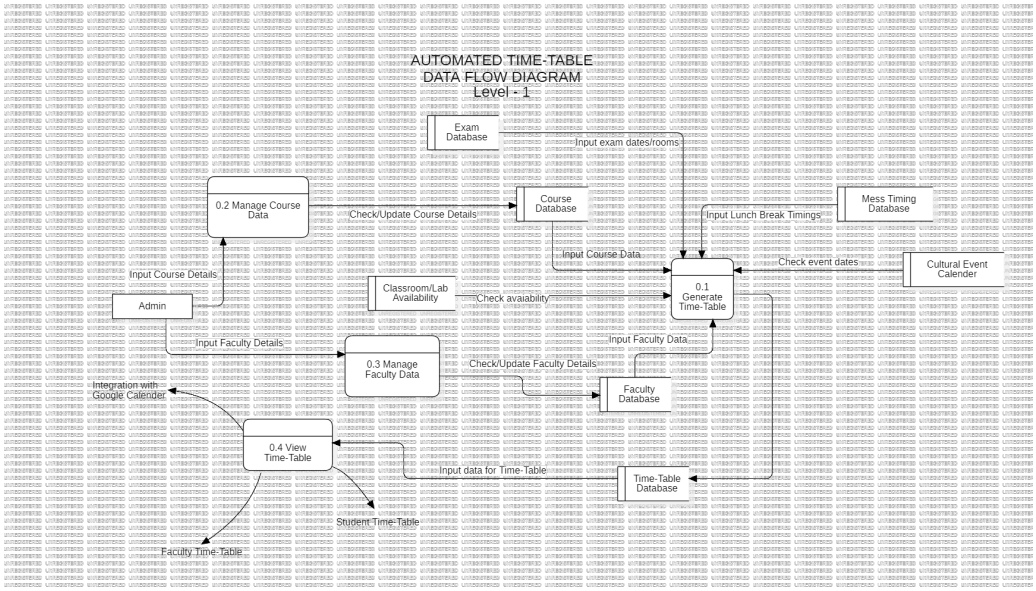

Figure 8. Level-1 DFD

4.1.3 Level - 2

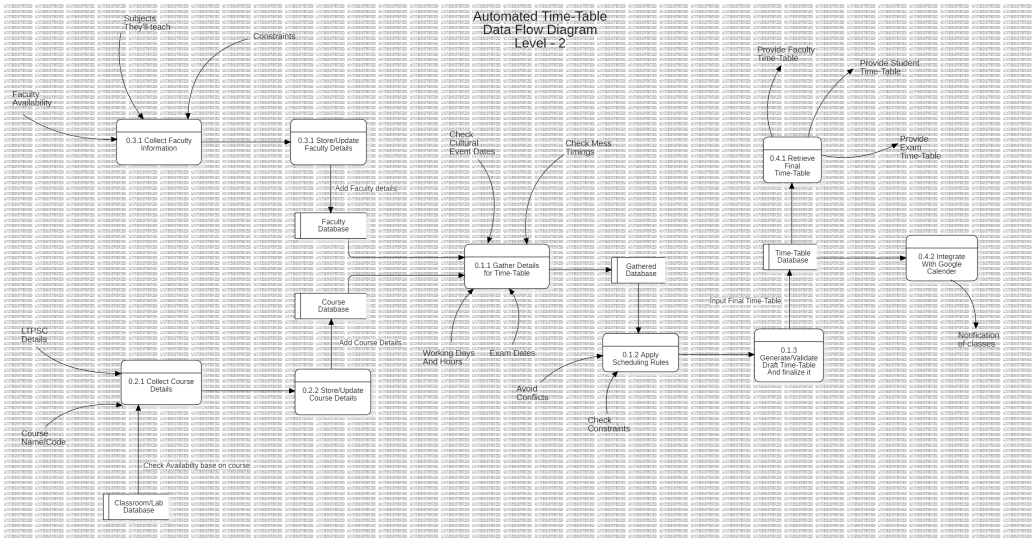

Figure 9. Level-2 DFD

4.2 Data Dictionary

```
Data dictionary for Automated Time-Table System (DFD Level 0 / Level 1 / Level 2)

student-id: integer
faculty-id: integer
admin-id: integer
course-id: string
course-name: string
section: string
semester: integer
room-no: string
capacity: integer
year: integer
month: integer
day: integer
hour: integer
minute: integer
start-time: hour + minute
end-time: hour + minute
time-slot: start-time + end-time
day-of-week: [Monday,Tuesday,Wednesday,Thursday,Friday,Saturday]

/* LTPSC = Lecture / Tutorial / Practical / Seminar / Course-Type */
LTPSC: [Lecture,Tutorial,Practical,Seminar,Course-Type]
Course-Type: [Core,Elective,Lab-based,Theory]

/* primitives introduced in Level 2 */
subject-code: string
subject-name: string
subject: subject-code + subject-name + LTPSC
subjects: {subject}*
constraint-id: string
constraint: constraint-id + description + scope + type
```

```
description: string
scope: [Faculty,Room,Course,Section]
type: string
working-days: {day-of-week}*
working-hours: start-time + end-time
working-days-and-hours: working-days + working-hours
faculty-availability: faculty-id + available-slots
available-slots: {time-slot}*
gathered-database: /* composite collected by Gather Details */
scheduling-rule-id: string
rule: scheduling-rule-id + rule-text + priority
scheduling-rules: {rule}*
conflict-id: string
conflict: conflict-id + time-slot + reason + involved-entries
conflict-list: {conflict}*
conflicts-resolved-flag: [Yes,No]

/* Course and faculty composites */
course-details: course-id + course-name + semester + section + subjects + LTPSC
faculty-details: faculty-id + faculty-name + department + faculty-availability
faculty-name: string
department: string

/* Classroom / availability */
classroom-availability: room-no + date + time-slot + status + (exam-flag)
status: [Available,Not-Available]
exam-flag: [Yes,No]
date: year + month + day

/* Mess / breaks / events */
mess-timing-database: {mess-timing}*
mess-timing: start-time + end-time + description
cultural-event-calendar: {event}*
event: event-id + event-name + date + time-slot + description
```

```
event-id: string
event-name: string

/* Exam schedule */
exam-database: {exam}*
exam: exam-id + course-id + exam-date + exam-room
exam-id: string
exam-date: date
exam-room: room-no

/* Time-table constructs */
time-slot-entry: course-id + faculty-id + room-no + day-of-week + time-slot + LTPSC
time-table: {time-slot-entry}*

/* timetable for a semester+section or faculty */
student-time-table: student-id + semester + section + time-table
faculty-time-table: faculty-id + time-table
time-table-database: {time-table}*

/* Level-2 gathered and draft structures */
gathered-database: Course-Details + Faculty-Details + Classroom-Availability +
                   Mess-Timing-Database + Cultural-Event-Calendar + Exam-Database +
                   working-days-and-hours + subjects + constraints + faculty-availability
draft-time-table: time-table + conflict-list + draft-status
draft-status: [Draft,Under-Review]
scheduled-time-table: time-table + conflicts-resolved-flag

/* Processes / requests / responses (data flows) */
request-time-table: student-id + semester + section
provide-time-table: student-time-table
finalized-time-table: time-table + approval-status
approval-status: [Approved,Pending,Rejected]

/* Generate timetable inputs/outputs (Level 1 / Level 2) */
```

```
generate-time-table-inputs: course-details + faculty-details + classroom-availability +
                            mess-timing-database + cultural-event-calendar + exam-database
                            + working-days-and-hours + constraints
generate-time-table-output: finalized-time-table + time-table-database + draft-time-table

/* Level-2 scheduling specifics */
gathered-database-output: gathered-database

apply-scheduling-rules: scheduling-rules + gathered-database
apply-scheduling-output: draft-time-table + conflict-list

draft-validate-output: finalized-time-table + time-table-database

/* Google Calendar integration (notifications only) */
google-calendar-notification: student-id + notification-entry
notification-entry: course-id + faculty-name + room-no + date + time-slot + reminder-type
reminder-type: [Class-Start,Class-Update,Cancellation]

/* Databases (stores) */
course-database: {course-details}*
faculty-database: {faculty-details}*
classroom/lab-database: {room-no + capacity + type}*
time-table-database: {time-table}*
mess-timing-database: {mess-timing}*
cultural-event-calendar-db: {event}*
exam-database: {exam}*
```

4.3 Modular Structure

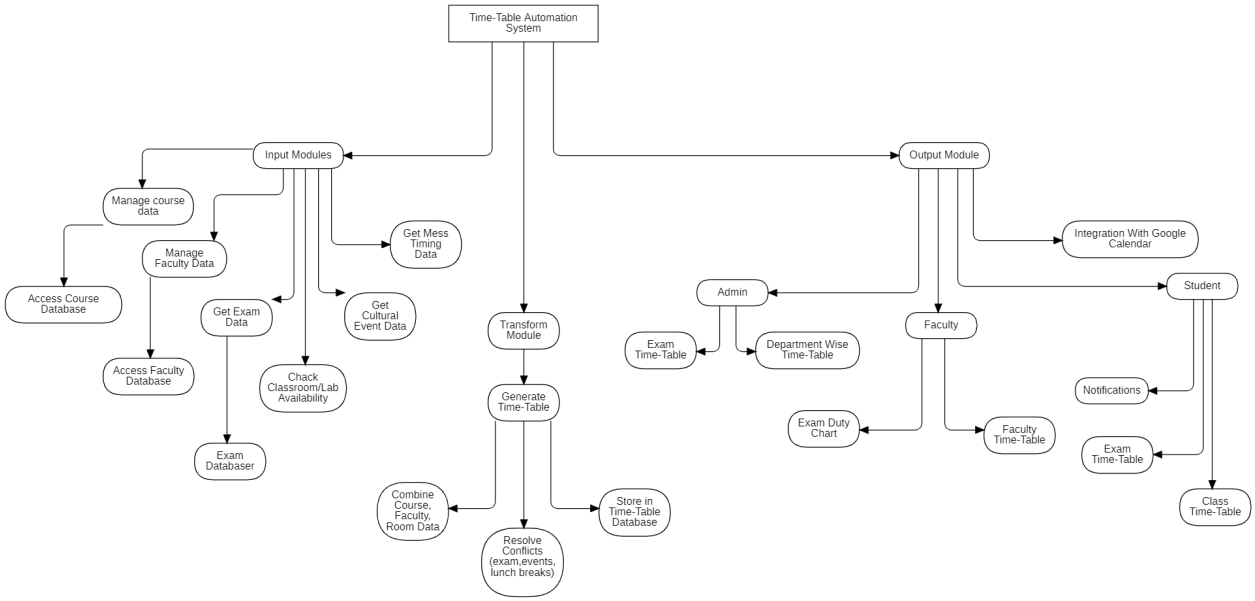

Figure 10. Modular Structure

4.4 Low-Level Design

4.4.1 Data Structures

The Following are the data structures which will be used to represent the key entities. **Course**

```
// Holds details of each course
struct Course {
    char course_id[10];
    char course_name[50];
    int semester;
    char section[5];
    char type[15];
    int credits;
};
```

Constraint // Faculty, Room, Course, Section

```
struct Constraint {
    int constraint_id;
    char description[100];
    char scope[20];
    int priority;
};
```

Student

```
// Stores student details
struct Student {
    int student_id;
    char name[50];
    char department[30];
};
```

Faculty

```
// Stores faculty details
```

```
struct Faculty {
    int faculty_id;
    char name[50];
    char department[30];
};
```

Room

```
// Information about classrooms/labs
struct Room {
    char room_no[10];
    int capacity;
    char type[10]; //
};
```

TimeSlot

```
// Represents a single period of time
struct TimeSlot {
    char day[10];
    char start_time[6];
    char end_time[6];
};
```

Timetable Entry

```
// One scheduled class (course + faculty + room + time)
struct TimetableEntry {
    char course_id[10];
    int faculty_id;
    char room_no[10];
    struct TimeSlot slot;
    int is_finalized;
};
```

Timetable

```
// Complete timetable (list of entries)
struct Timetable {
    struct TimetableEntry entries[100];
    int entry_count;
};
```

Conflict

```
// Represents an issue (e.g., room clash, faculty clash)
struct Conflict {
    char message[100];
    struct TimetableEntry entry;
};
```

Availability

```
struct Availability {
    int faculty_id;
    struct TimeSlot slots[50];
    int slot_count;
};
```

4.4.2 Function Declarations (with Description)

Data Input

```
void loadCourses(struct Course courses[], int *count);

void loadFaculty(struct Faculty faculty[], int *count);

void loadRooms(struct Room rooms[], int *count);

void loadConstraints(struct Constraint constraints[], int *count);

void loadAvailability(struct Availability avail[], int *count);
```

Timetable Generation

```
struct Timetable generateInitialTimetable(
    struct Course courses[], int cCount,
    struct Faculty faculty[], int fCount,
    struct Room rooms[], int rCount,
    struct Constraint constraints[], int conCount
)
```

Validation

```
int checkConflicts(struct Timetable t, struct Conflict conflicts[], int *confCount);
```

Conflict Resolution

```
int detectConflicts(struct Timetable t, struct Conflict conflicts[], int *count);
void autoResolveConflicts(struct Timetable *t, struct Conflict conflicts[], int
    count);
```

Exam Scheduling

```
struct Timetable generateExamSchedule(
    struct Course courses[], int cCount,
    struct Room rooms[], int rCount
);
int assignSeatingPlan(struct Timetable examTT, int seatMatrix[][100]);
```

Notifications

```
void exportToGoogleCalendar(struct Timetable t);
void saveTimetable(struct Timetable t, char filename[]);
void notifyUsers(struct Timetable t);
```

5 Coding/ Implementation

Tech Stack

- **Programming Language:** `Python`
- **Frontend :** `React.js`
- **Backend :** `Python with Django`
- **Data Handling:** `CSV` files, `Pandas`, `NumPy`
- **Version Control:** `Git`, `GitHub`
- **IDE / Editor:** `Visual Studio Code`

6 Conclusion

The above project acts as a documentation required in the creating of Automated Time-Table for IIIT Dharwad. It discusses and elaborates the issues with the previous time table which was manually created with lots of complexitites. And hence it further provides the requirements for new Automated Software for creating Time-Table for IIIT Dharwad.

Automating the scheduling function will improve the utilization of classrooms and resources, reduce errors in allocating slots, and significantly reduce administrative burden.

This project was supervised under the guidance of Professor Vivekraj VK.

References

[1] Burke, Edmund K., and Sanja Petrovic. *Recent research directions in automated timetabling*. European Journal of Operational Research, 140(2), 266–280, 2002.

[2] Wren, Anthony. *Scheduling, timetabling and rostering — A special relationship?* In International Conference on the Practice and Theory of Automated Timetabling, Springer, Berlin, Heidelberg, 1996, pp. 46–75.

[3] Abdullahi, Mohammed, and Nor Azman Ismail. *A survey on the use of metaheuristics in course timetabling problems.* International Journal of Advanced Computer Science and Applications (IJACSA), 8(2), 2017, 152–162.

[4] Petrovic, Sanja, and Edmund Burke. *Case-based reasoning in course timetabling: An overview.* In International Conference on the Practice and Theory of Automated Timetabling.