

# Teachers' Timetable Management System

M Aswartha Reddy  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
maswarthar.cs21@rvce.edu.in

Shishira M Iyar  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
shishiramiyar.cs21@rvce.edu.in

Saksham Prasad  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
sakshamprasad.cs21@rvce.edu.in

Parakh Agrawal  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
parakhagrawal.cs21@rvce.edu.in

Prof. Veena Gadad  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
veenagadad@rvce.edu.in

Prof. Sindhu Rajendran  
Department of Electronics  
Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
sindhur@rvce.edu.in

Aditya Veer Singh  
Department of Computer Science  
and Engineering  
Rashtreeya Vidyalaya College of  
Engineering  
Bangalore, India  
adityaveers.cs21@rvce.edu.in

**Abstract**—This paper deals with the development of a website that displays teachers' timetables using JAM stack. Through this we aim to improve the efficiency of timetable management in educational institutions. It also deals with the development of an interface that allows efficient substitute teacher assignment based on multiple appropriate factors like daily work-load. The workloads of all the teachers are also calculated and displayed.

**Keywords**—Timetable, Teachers, Education, Substitute teacher

## I. INTRODUCTION

This conference paper discusses the development and implementation of a website to display teachers' timetables in a user-friendly format. In many educational institutions, students are not aware of their teachers' timetables. This makes it difficult for them to know when to meet specific teachers. Moreover, it is convenient for teachers to be able to see the timetables of their colleagues. We have developed a website where students and teachers can view the teachers' timetables sorted by department. The website is user-friendly and easily accessible.

As the second part of our problem statement, we aim to solve the problem of substitute teacher assignment digitally. Prior to this, substitute teacher assignment was a time-consuming, labor-intensive process. A timetable officer would have to go through the timetables of each teacher to find if that teacher is free at that given time. It would be ideal if a teacher who is most free in a day is given substitute teacher duty. But for this, the officer would have to manually calculate the working hours of each teacher in each day. We aim to automate this process as well.

The final product is a single well-designed website for timetable viewing and finding potential substitute teachers.

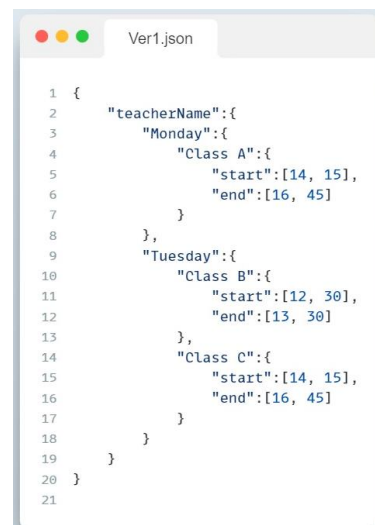
## II. METHODOLOGY

We have used JAM stack (JavaScript, API, Markup) to design the website. Google charts API is used to draw the timetables. First the data is converted into a suitable format.

### A. Data

Well formatted data is important for any program to function. The timetables in excel format are not machine-readable. We convert it into a suitable JSON file for well-functioning of the website and ease of access. The JSON can be easily read by the JavaScript code of the backend of the website.

Our initial approach was to store the data in this format:



```
1 {
2   "teacherName":{
3     "Monday":{
4       "Class A":{
5         "start":[14, 15],
6         "end":[16, 45]
7       }
8     },
9     "Tuesday":{
10      "Class B":{
11        "start":[12, 30],
12        "end":[13, 30]
13      },
14      "Class C":{
15        "start":[14, 15],
16        "end":[16, 45]
17      }
18    }
19  }
20 }
21
```

Figure 1-Initial data format

But this approach had a couple of drawbacks

1. The times are stored using two variables: hour and minute. This makes calculations difficult and is a wastage of memory. Since time is unidimensional, it is more appropriate to use a single variable to store time. Calculations to find the duration of a class becomes much simpler with one variable. Just subtracting the end and start times of a class gives the duration. This can also be done backwards to find the end time of a class. To check if an event(class) occurs before or after another event, we just have to compare the time variables of these two events.
2. JSON data structure is such that keys must be unique. But class names are not always unique. A teacher might take the same class twice in a day. This results in a conflict in the JSON file. When that occurs, the last uploaded data is stored and the initial data is lost. This leads to data loss and makes the website unviable.

To solve the former, we have come up with a standard time format. The time format we use is minutes since 9am. All the times are stored in their “minutes since 9am” equivalents. Instead of storing start and end times, we store the start time and duration of classes in terms of minutes since 9am. This makes calculations and comparisons easy. In this method, the time 10:30 am is stored as 90, since it is 90 minutes since 9 o clock. The total work-hours of a teacher are just the sum of all the durations. One might assume that converting this to normal time format might be a difficult task. But it is rather simple. It can be converted back to normal time using the following equations:

$$startHour = 9 + \lfloor \frac{start}{60} \rfloor$$

$$startMin = start \% 60$$

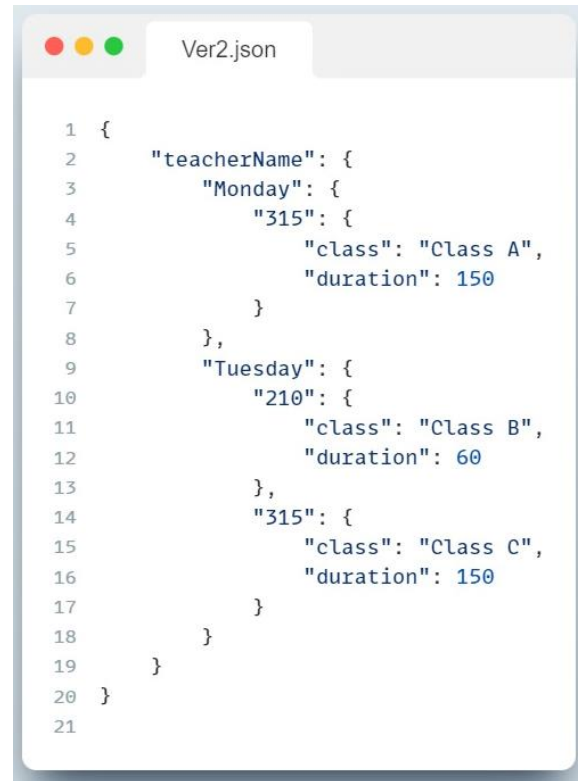
$$endHour = 9 + \lfloor \frac{start + duration}{60} \rfloor$$

$$endMin = (start + duration) \% 60$$

These equations are computationally inexpensive and thus the results are obtained quite fast. These times can then be showed on the website for humans to read. These calculations are done in real-time and displayed without any latency to improve the user experience.

To solve the latter problem, we store the start times as the key for JSON. The class name and duration are saved as values in the JSON under this key. Start times are unique since no two classes can start at the same time for a given teacher. The times have a maximum precision of 1 minute. Times less than this are not considered. That should not be an issue since timetables never have more than minute precision.

Thus, we arrive at this data structure.



```
1 {
2   "teacherName": {
3     "Monday": {
4       "315": {
5         "class": "Class A",
6         "duration": 150
7       }
8     },
9     "Tuesday": {
10      "210": {
11        "class": "Class B",
12        "duration": 60
13      },
14      "315": {
15        "class": "Class C",
16        "duration": 150
17      }
18    }
19  }
20 }
21
```

Figure 2-Improved data format

The first key is teacher name, inside which are keys for the different days of the week. Inside this are the start times of all the classes.

This contains two values namely class name and duration. Class name and duration is accessed by indexing the JSON as follows:

JSON[teacher name][day][start time][class]

JSON[teacher name][day][start time][duration]

This data is stored locally on the server that hosts the website for easy access.

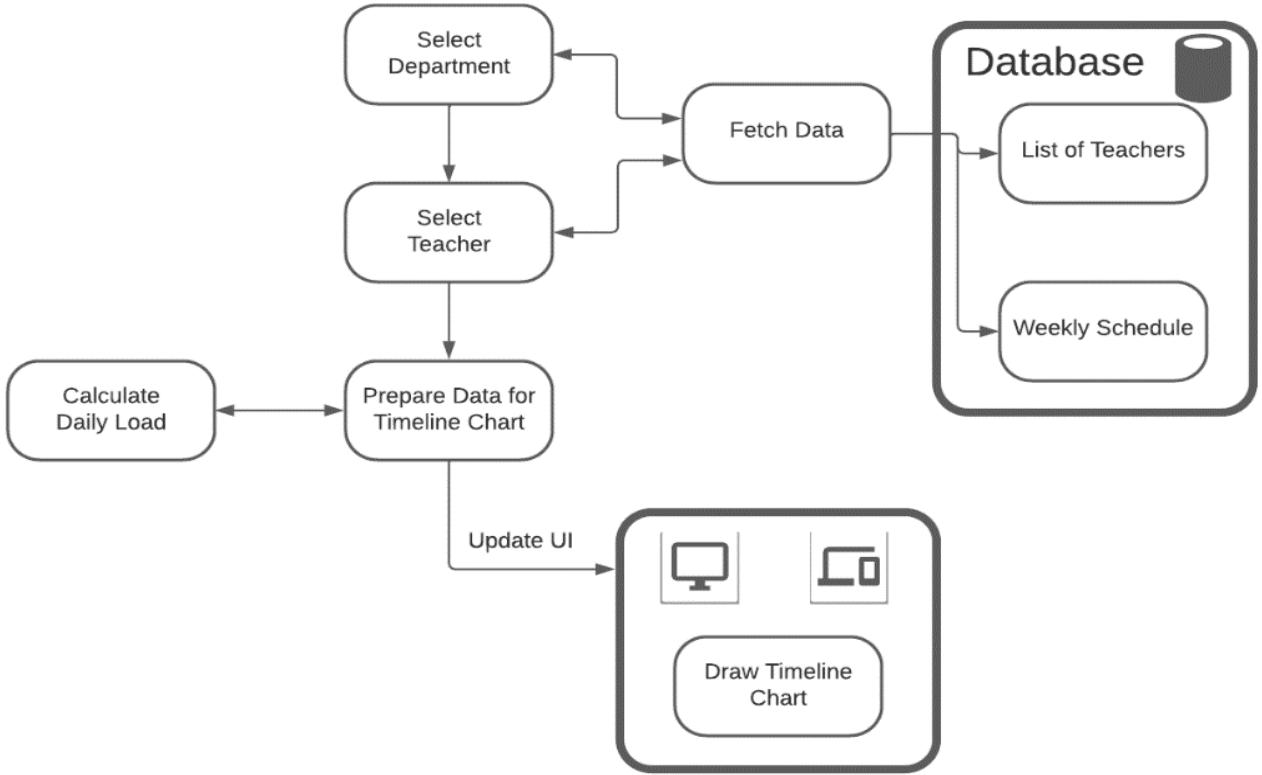


Figure 3-Flowchart

The website consists of two webpages, one for viewing the timetables and another for finding substitute teachers.

### B. Timetable Page

The timetable page takes department and teacher names as inputs and displays the timetables of the teachers in a visually pleasing manner. It also shows the daily workload of the teachers. The timetable is a timeline chart drawn using Google Charts API. The API takes inputs such as start and end times, and the text to be written in the timetable. Then it produces a timeline chart in the HTML element assigned to it. The details of a teacher like their name, qualifications etc. can also be shown on this page.

The timeline chart takes data in a specific format. It takes a 2D array, with the rows being different elements and columns being the attributes of each element. The first column determines which row in the chart the element falls in. The next one is the text displayed in the element. The next two are the start and the end times.

When hovered over the chart elements, vital information such as duration is displayed in a tool-tip. The tool-tip can be customized by modifying the source code. The daily workload of all the teachers is calculated as follows:

$$Workload = \sum_0^n duration$$

We iterate through all the classes a teacher takes in each day and sum up their durations to obtain the daily workloads. Daily workload is then displayed on the webpage. Sum of all the daily workloads is the weekly workload of the teacher.

### C. Substitute teachers page

The find teachers page takes department, date, start time, and end time and calculates the teachers who are free during the entered timeslot in that day. Any of those teachers can be chosen to be the substitute teacher. Since it seems more appropriate to assign a teacher who has relatively less work in that day, the daily workloads of all the teachers in the list are

calculated. The information is then displayed in a table on the website.

To obtain the list of teachers who are free in a particular day between two times (start and start + duration), we need a function findTeachers (day, start, duration) which checks if each teacher from the selected department is free between those times.

The function checks each class of the teacher to see if it overlaps with the entered timeslot. If there is an overlap, a flag is flagged and that teacher is determined to be not free in the timeslot. If the flag is not triggered at all for a given teacher, then that teacher is free in the entered timeslot.

The logic for checking overlap is as follows:

For each class of that teacher in that day,

Let st : start time of the class

dur : duration of the class

start : start time of the entered timeslot

duration : duration of the entered timeslot

```
if(st+dur > start && st<start+duration){
    /*ensures all classes that start before
    the given time-slot ends before the given time-slot
    AND
    ensures no classes start in the timeslot
    */
    flag = 1
}
```

The if statement checks for two conditions. If the end time of the class is after the start time of the timeslot **AND** the start time of the class is before the end time of the timeslot, the class and the timeslot overlap and the teacher is determined to be busy. If this condition is not true for all the classes of a teacher, the teacher is said to be free in the timeslot.

A table is generated with the teacher names and their daily workloads. Teachers from this table can be selected for assignment. When selected, the timetable of that teacher is displayed under the table. This way, the timetable officer can compare the timetables of different teachers before assigning a substitute teacher. This ensures that other factors can also be considered before assigning a substitute teacher.

### III. TECH STACK

- JAM stack: The core of the website is designed using JAM stack. JavaScript handles all the logic, whereas Markup defines the front end. Java script is an excellent tool to build simple, yet elegant websites.
- Google charts API: Google charts API is a tool for data visualization made by google. In this project, we have used the timeline chart from this API to display the

timetables in a visually appealing manner. Timeline charts are appropriate for timetables.

- Bootstrap: Bootstrap is a free, open-source front-end development framework for the creation of websites and web apps. We have used Bootstrap to stylise the website. Bootstrap ensures a visually pleasing and comfortable user experience.
- Git and GitHub: Git is an open-source version control system. We used git to keep track of the changes made to our source code and other files. Git keeps versions of the source code so that any revisions to the source code can be reverted easily. GitHub is a free hosting service for git. We used GitHub as a remote server to collaborate on the project.
- Python is used for data entry. Python is excellent at reading and modifying JSON files. A python script is created which allows data entry on the terminal. It uses sentinel control and loops to enter teacher's timetables. The timetable officer, or anyone else with access, can upload the data manually in a simple manner using this script.
- The website is hosted on Heroku app, which is a free hosting service. It is also hosted on GitHub pages for redundancy.

### IV. RESULTS AND CONCLUSION

This project resulted in the implementation of a single well-designed website for teachers' timetable viewing and finding potential substitute teachers. It is found that using the website is much simpler and more convenient to view teacher's timetables. It is also found that substitute teacher assignment is significantly easier and more accurate using the website.

The timetables (Fig. 1) are well designed to be visually appealing.

The table for finding substitute teachers (Fig 2) is designed keeping usability in mind.

The website we have developed solves many of the current problems in the field of timetables.

In the future we wish to develop a system that can generate these timetables given a set of input constraints using Machine-Learning algorithms.

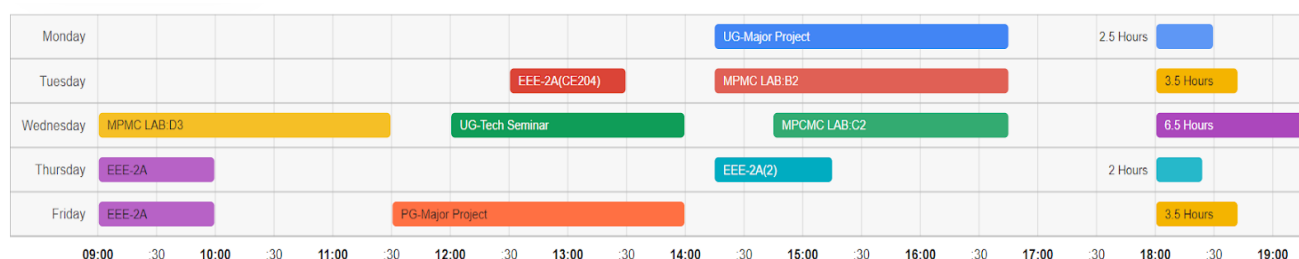


Figure 4-Timetable displayed using timeline chart

1	Sindhu Rajendran	2
2	M Uttara Kumari	4.5
3	Ravishankar S	0
4	B S Kariyappa	1
5	K R Usha Rani	2
6	Shylashree N	4

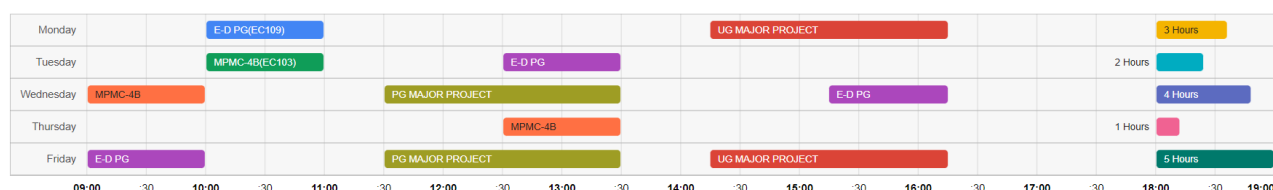


Figure 5-Substitute teacher assignment

## REFERENCES

- [1] A, Parkavi. (2018). A STUDY ON AUTOMATIC TIMETABLE GENERATOR. International Journal of Innovative Research & Growth. May.
- [2] Srinivasan, D. & Seow, Tian & Xu, Jian-Xin. (2002). Automated time table generation using multiple context reasoning for university modules. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002. 2. 1751 - 1756. 10.1109/CEC.2002.1004507.
- [3] AUTOMATIC TIME TABLE GENERATION USING GENETIC ALGORITHM - JETIR2107265
- [4] Anirudha Nanda "An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach". International Journal of Machine Learning and Computing.
- [5] Birbas, Theodore & Daskalaki, Sophia & Housos, Efthymios. (2009). School timetabling for quality student and teacher schedules. J. Scheduling. 12. 177-197. 10.1007/s10951-008-0088-2.
- [6] Cangalovic, M. & Schreuder, J.A.M. (1991). Exact Coloring Algorithm for Weighted Graph Applied to Timetabling Problems with Lectures of Different Length. European Journal of Operational Research, 51(2), 248-258.
- [7] Daskalaki, S., & Birbas, T. (2005). Efficient Solutions for a University Timetabling Problem through Integer Programming. European Journal of Operational Research, 160, 106-120.
- [8] Deris, S.B., Omatu, S., Ohta, H., Samat, P. (1997). University timetabling by constraint-based reasoning: A case study. Journal of the Operational Research Society, 48, 1178-1190.
- [9] Eikelder, H.M.M. ten, & Willemsen, R.J. (2001). Some Complexity Aspects of Secondary School Timetabling Problems. In E.K. Burke and W. Erben (Eds.), Practice & Theory of Automated Timetabling III, LNCS, No.2079 (pp. 3-17). Springer-Verlag.
- [10] Papoutsis, K., Valouxis, C., Housos, E. (2003). A column generation approach for the timetabling problem of Greek high schools. Journal of Operational Research Society, 54, 230-238.
- [11] Shu-Chuan Chu, Yi-Tin Chen ; Jiun-Huei Ho, Timetable Scheduling Using Particle Swarm Optimization, IEEE(2006)
- [12] Ioannis X. Tassopoulos, Grigorios N. Beligiannis, "A hybrid particle swarm optimization based algorithm for high school timetabling problems", in applied soft computing 12(2012), 3472-3489 [2] Johor Bahru, Safaai, D. ; Zaiton, M., A Combination of PSO and Local Search in University Course Timetabling Problem, IEEE(2009),492-495 [3] Liu Su-hua, Li Lin, Study of course scheduling based on particle swarm optimization , IEEE(2011), 1692-1695
- [13] [Bootstrap Documentaion](#)
- [14] [MDN Web Docs JavaScript Reference](#)
- [15] [Google Charts Timeline API Reference](#)
- [16] [bookCompanion Repository on GitHub](#)