

# Teachers' Timetable Management System

Veena S Gadad

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[veenagadad@rvce.edu.in](mailto:veenagadad@rvce.edu.in)

Sindhu Rajendran

Department of Electronics  
R. V. College of Engineering  
Bangalore, India  
[sindhur@rvce.edu.in](mailto:sindhur@rvce.edu.in)

M Aswartha Reddy

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[maswarthar.cs21@rvce.edu.in](mailto:maswarthar.cs21@rvce.edu.in)

Shishira M Iyar

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[shishiramiyar.cs21@rvce.edu.in](mailto:shishiramiyar.cs21@rvce.edu.in)

Aditya Veer Singh

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[adityaveers.cs21@rvce.edu.in](mailto:adityaveers.cs21@rvce.edu.in)

Saksham Prasad

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[sakshamprasad.cs21@rvce.edu.in](mailto:sakshamprasad.cs21@rvce.edu.in)

Parakh Agrawal

Department of Computer Science  
R. V. College of Engineering  
Bangalore, India  
[parakhagrawal.cs21@rvce.edu.in](mailto:parakhagrawal.cs21@rvce.edu.in)

**Abstract**—This paper deals with the development of a website that displays teachers' timetables. It also deals with the development of an interface that allows efficient substitute teacher assignment. The workloads of all the teachers are also calculated and displayed.

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

## 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. We have developed a website where students can view the teachers' timetables sorted by department.

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, labour-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.

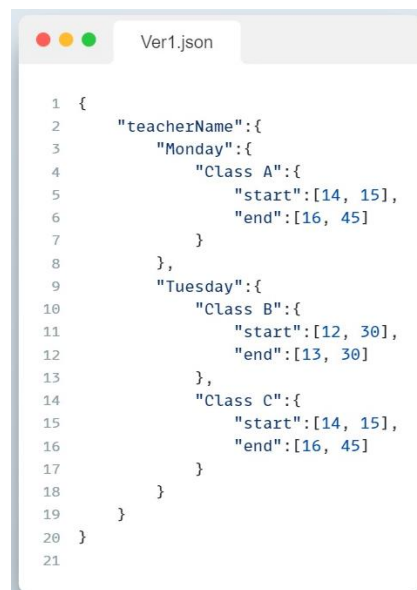
## II. METHODOLOGY

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

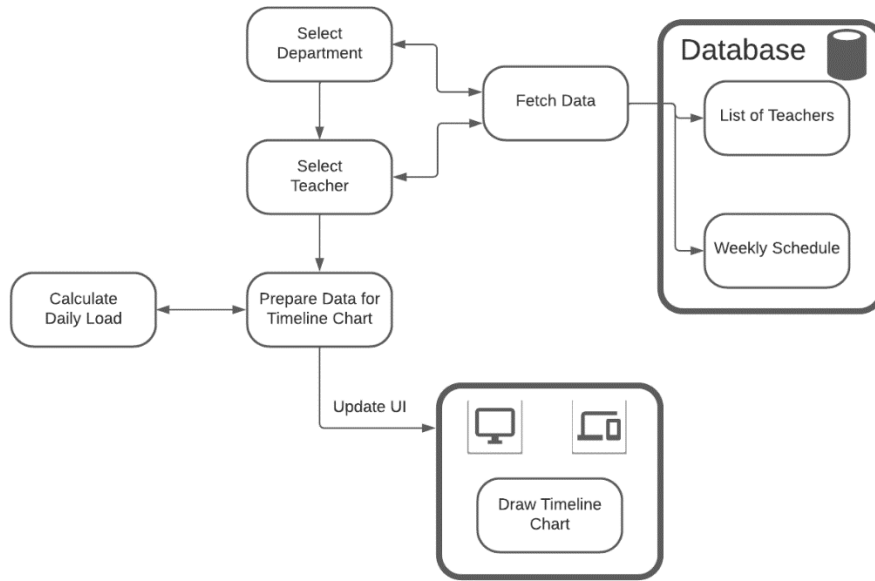
Well formatted data is important for any program to function. The timetables in excel format are not machine-readable. We convert it into a suitable JSON file for well-functioning of the website and ease of access.

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
```

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



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.
2. JSON keys must be unique, but class names are not always unique. This results in a conflict in the JSON file. This leads to data loss and makes the whole 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. This makes calculations and comparisons easy. For example, the total work-hours of a teacher is just the sum of all the durations. This can be converted back to normal time using the following equations:

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

$$startMin = start \% 60$$

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

To solve the latter, we store start time as the key for JSON. The class name and duration are saved as values in the JSON. Start times are unique since no two classes start at the same time for a given teacher. Thus, we arrive at this format.

```

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
  
```

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 their timetables. It also shows the daily

workload of the teachers. The timetable is a timeline chart      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
}
```

drawn using Google Charts API. The details of a teacher like their name, qualifications etc. can also be shown.

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 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.

### 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.

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

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. This way, the timetable officer can compare the timetables of different teachers 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.
- Google charts API: Google charts API is an excellent tool for data visualisation. In this project, we have used the timeline chart from this API to display the timetables in a visually appealing manner.
- 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.
- Git and GitHub: Git is an excellent open-source version control system. We used git to keep track of the changes made to our source code and other files. GitHub is a hosting service for git. We used GitHub as a remote server to collaborate on the project.
- Python is used for data entry
- The website is hosted on Heroku app, which is a 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.

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.

#### 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] [Bootstrap Documentaion](#) 
- [5] [MDN Web Docs JavaScript Reference](#) 
- [6] [Google Charts Timeline API Reference](#) 
- [7] [bookCompanion Repository on GitHub](#) 

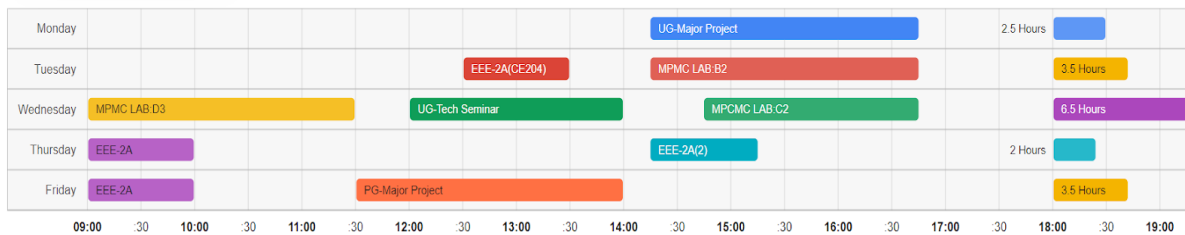


Figure 1-Example Timetable

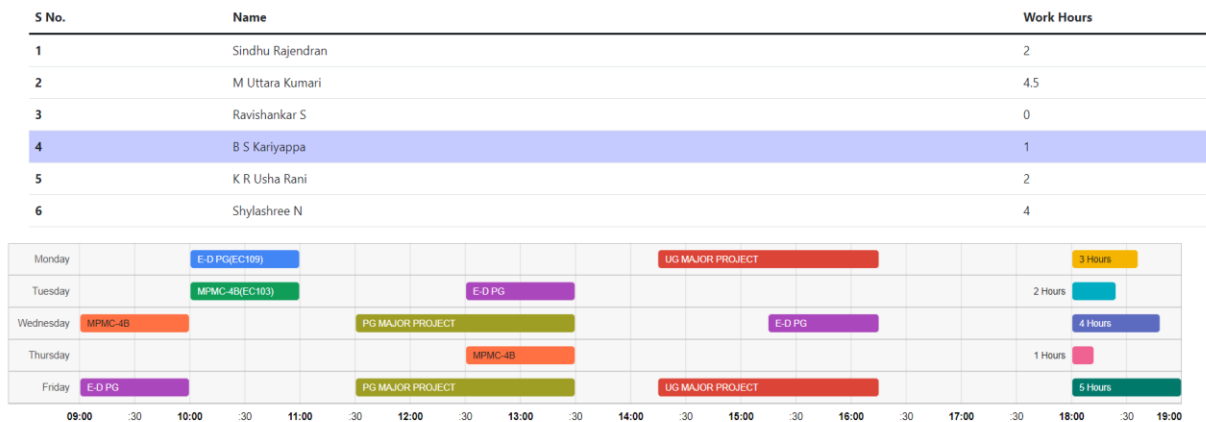


Figure 2-List of free teachers and timetable of the selected teacher