**Project Title --Developing a Backend Admin for Learner’s Academy.**

**Git link**

[**https://github.com/jyothiarunkr/Phase2-Project-Learners-Academy.git**](https://github.com/jyothiarunkr/Phase2-Project-Learners-Academy.git)

**Project Report By Dr. Jyothi N.M.**

# .Abstract

A college -Learner’s Academy timetable is a temporal arrangement of a set of classes and classrooms in which all given constraints are satisfied.

Timetabling has long been known to belong to the class of problems called NP hard. This project introduces a practical timetabling algorithm capable

of taking care of both strong and weak constraints effectively, used in an automated timetabling system.

Our project takes various inputs from the user such as Teacher List, Course List, Semester List, Room List, Day List and Timeslot as well as various rules,

facts and constraints for generating the Time Table

**Project objective:**

As a Full Stack Developer, design and develop a backend administrative portal for the Learner’s Academy.

Use the GitHub repository to manage the project artifacts.

**Background of the problem statement:**

Learner’s Academy is a school that has an online management system. The system keeps track of its classes, subjects, students, and teachers. It has a back-office application with a single administrator login.

**Functional Requirements:**

**The administrator can:**

● Set up a master list of all the subjects for all the classes  
● Set up a master list of all the teachers  
● Set up a master list of all the classes  
● Assign classes for subjects from the master list  
● Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)  
● Get a master list of students (Each student must be assigned to a single class)

**System Design**

**The flow and features of the application:**

● Plan more than two sprints to complete the application  
● Document the flow of the application and prepare a flow chart   
● List the core concepts and algorithms being used to complete this application  
● Implement the appropriate concepts, such as exceptions, collections, and sorting techniques for source code optimization and increased performance

**Technologies used:**

● Eclipse/IntelliJ: An IDE to code for the application   
● Java: A programming language to develop the web pages, databases, and others  
● SQL: To create tables for admin, classes, students, and other specifics  
● Git: To connect and push files from the local system to GitHub   
● GitHub: To store the application code and track its versions   
● Scrum: An efficient agile framework to deliver the product incrementally   
● Search and Sort techniques: Data structures used for the project   
● Specification document: Any open-source document or Google Docs

# **2.Introduction**

Even though most college administrative work has been computerized, the lecture- timetable scheduling is still mostly done manually due to its inherent difficulties.

The manual lecture-timetable scheduling demands considerable time and efforts. The lecture-timetable scheduling is a constraint satisfaction

problem in which we find a solution that satisfies the given set of constraints.

The college lecture-timetabling problem asks us to find some time slots and classrooms which satisfy the constraints imposed on

offered courses, instructors, classrooms and so on. Therefore, the variables to be instantiated are time slots and classrooms of offered courses.

Since the problem is a combinatorial optimization problem belonging to NP-hard class, the computation time for timetabling tends

to grow exponentially as the number of variables increase.

There have been a number of approaches made in the past decades to the problem of constructing timetables for colleges and schools.

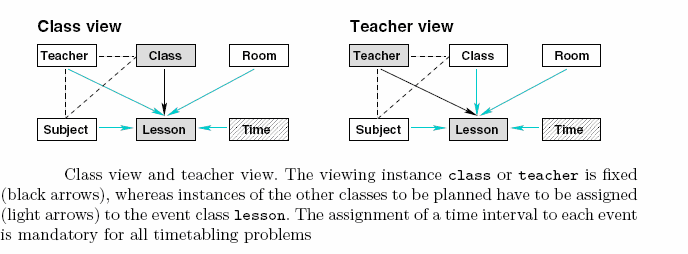
2.1Timetabling

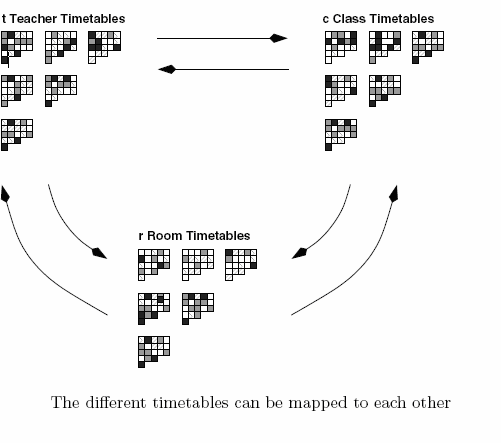
A timetable construction is an NP-complete scheduling problem. It is not a standard job–shop problem because of the additional classroom allocation.

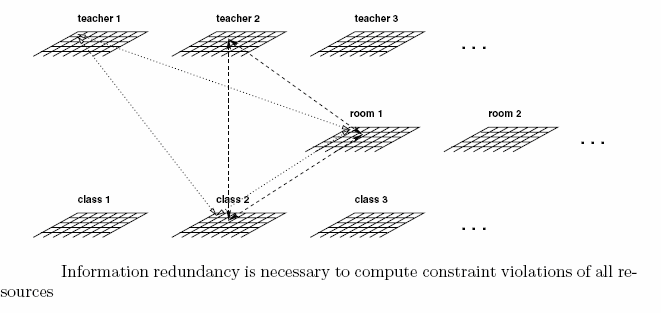
It is large and highly constrained, but above all the problem differs greatly for different schools and educational institutions.

It is difficult to write a universal program, suitable for all imaginable timetabling problems. Although manual construction of timetables

is time-consuming, it is still widespread, because of the lack of appropriate computer programs.







* 1. Constraints

Assignments usually cannot be done arbitrarily, but many constraints have to be considered. We distinguish two different types,

namely hard and soft constraints. A solution is feasible if no hard constraints are violated.

A feasible solution is better than another if fewer soft constraints are violated.

A timetabling algorithm can use different strategies to get a solution without violations of hard constraints.

Violations can either be avoided from the outset or penalized to lead the algorithm towards better solutions and introduce repair mechanisms.

The time slots are generally assigned from 11 AM to 5.30 PM for weekdays and from

7.30 AM to 11 AM on Saturday. The time time slots are labeled as *Ti* (*i* = 1 *. . .* 55)..

There are various constraints to be satisfied at the time to instantiate variables about time slots and classrooms.

The constraints can be categorized into strong and weak constraints as follows:

2.3Strong Constraints

*C*1: A classroom is not assigned to more than one lecture at the same time.

*C*2: An instructor cannot teach more than one class at the same time.

*C*3: Courses for the same year-session students of a department cannot take place at the same time.

*C*4: The classroom for a course should have enough capacity to take students registered in the course.

*C*5: The classroom should be well equipped with required facilities for the classes.

## 2.4Weak Constraints

*C*6: The lectures are not assigned to time slots which are in the instructor’s forbidden time zones.

*C*7: Instructors’ daily lecture hours should be restricted to be within the allowed maximum hours.

*C*8: As far as possible, classes are scheduled in the instructor’s preferred time zones.

*C*9: A lunch/dinner break must be scheduled.

*C*10: If possible, the lectures should not be scheduled on too late night time slots.

*C*11: The theory courses are scheduled on Monday and Tuesday, and the practical courses are scheduled on Wednesday, Thursday, and Friday.

*C*12: For daytime students, the cultural subjects courses are scheduled in the morning time slots (1st to 4th time slots on weekdays),

and major courses are scheduled in the afternoon time slots (5th to 8th time slots).

*C*13: For nighttime students, the cultural-subjects courses are scheduled on the 11th to 12th slots, and the major courses are scheduled on the 13th to 16th time slots on weekdays.

*C*14: If possible, the lecture hours for a course should be scheduled consecutively.

*C*15: As far as possible, classes should be scheduled in their corresponding department’s exclusive-use classrooms.

*C*16: The classrooms should be allocated in a manner to minimize the distances between adjacent classes’s classrooms.

It is desirable for timetables to satisfy all strong and weak constraints. However, it is usually not easy to meet all these constraints.

The strong constraints must be satisfied all the times, but weak constraints can be somewhat sacrificed to find feasible timetables.

Among the weak constraints, constraints from *C*6 to *C*14 are imposed on the allocation of time slots.

Constraints *C*15 and *C*16 are imposed on the allocation of classrooms. The constraints are arranged in the order of importance in the scheduling.

For example, if it is impossible to find schedules satisfying both *C*6 and *C*7 simultaneously,

it is preferred to choose a schedule that satisfies *C*6 but *C*7 rather than a schedule satisfying *C*7 but *C*6.

# 

# Requirements and Analysis

In this section we formulate the various requirements of our project.

We wanted that timetable generator should be able to consider following different resources available

1. Teachers
2. Courses
3. Semesters
4. Rooms

It should provide mechanism for specifying various constraints such as

* 1. Teacher-Room Constraints
  2. Year-Room Constraints
  3. Teacher-Timeslot Constraints
  4. Teacher-Day Constraints
  5. Teacher-Year Constraints

**3.Data Base Design**

**Following Tables are created**

**1.**Teacher List

2.Course List

3.Day List

4.Semester List

5.Room List

6.Formatting Options

7.Constraints, Rules, Policies

**Table creation**

**CREATE TABLE `day\_list` (**

**`hr` varchar(100) DEFAULT NULL,**

**`mon` varchar(100) DEFAULT NULL,**

**`tue` varchar(100) DEFAULT NULL,**

**`wed` varchar(100) DEFAULT NULL,**

**`thu` varchar(100) DEFAULT NULL,**

**`fri` varchar(100) DEFAULT NULL,**

**`sat` varchar(100) DEFAULT NULL,**

**`yr` varchar(100) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**CREATE TABLE `faculty\_list` (**

**`facname` varchar(100) DEFAULT NULL,**

**`email` varchar(100) DEFAULT NULL,**

**`address` varchar(100) DEFAULT NULL,**

**`mobile` varchar(100) DEFAULT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

Similary table created for course, room and semester

**4.Conclusion**

**The process of Time Table generation has been fully automated with this software.**

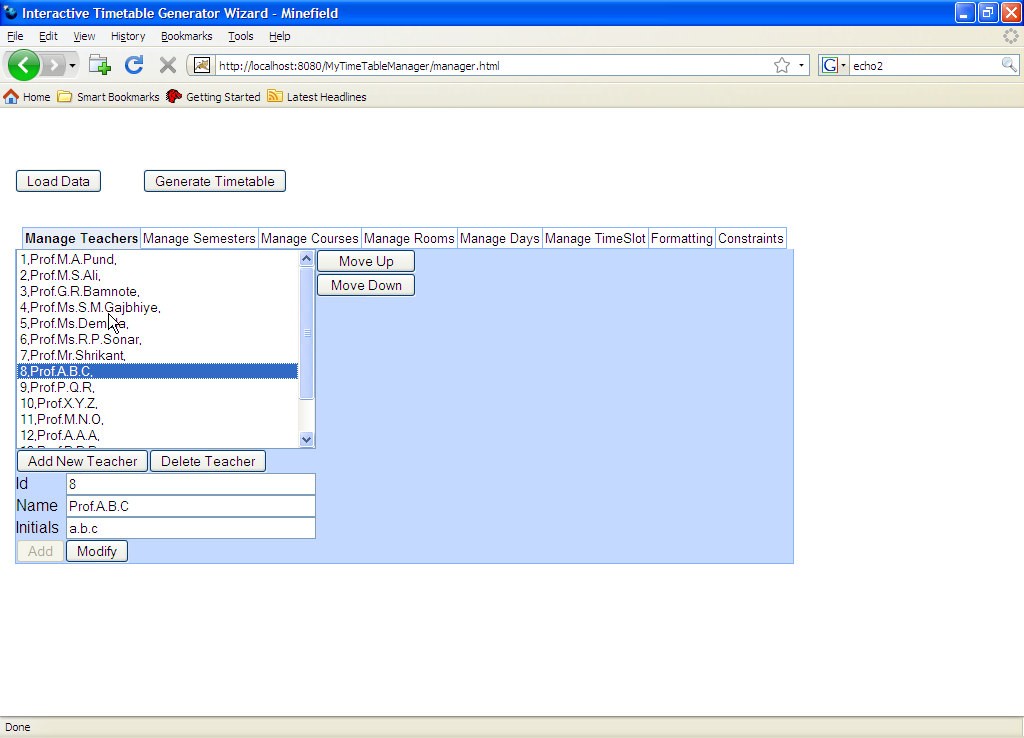
**This web app can now cater to multiple colleges, universities and schools which can**

**rely on it for their Time Table scheduling which earlier had to be done by hand.**

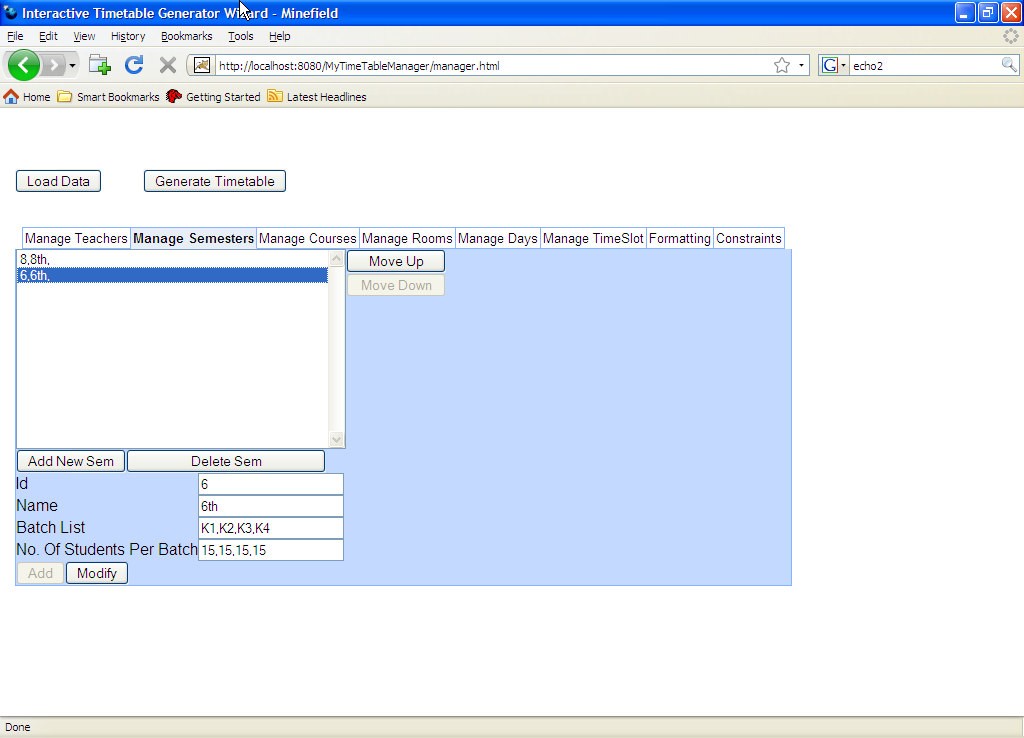
**5.Screen Shots**

**Step by step Screen shots of the Project**

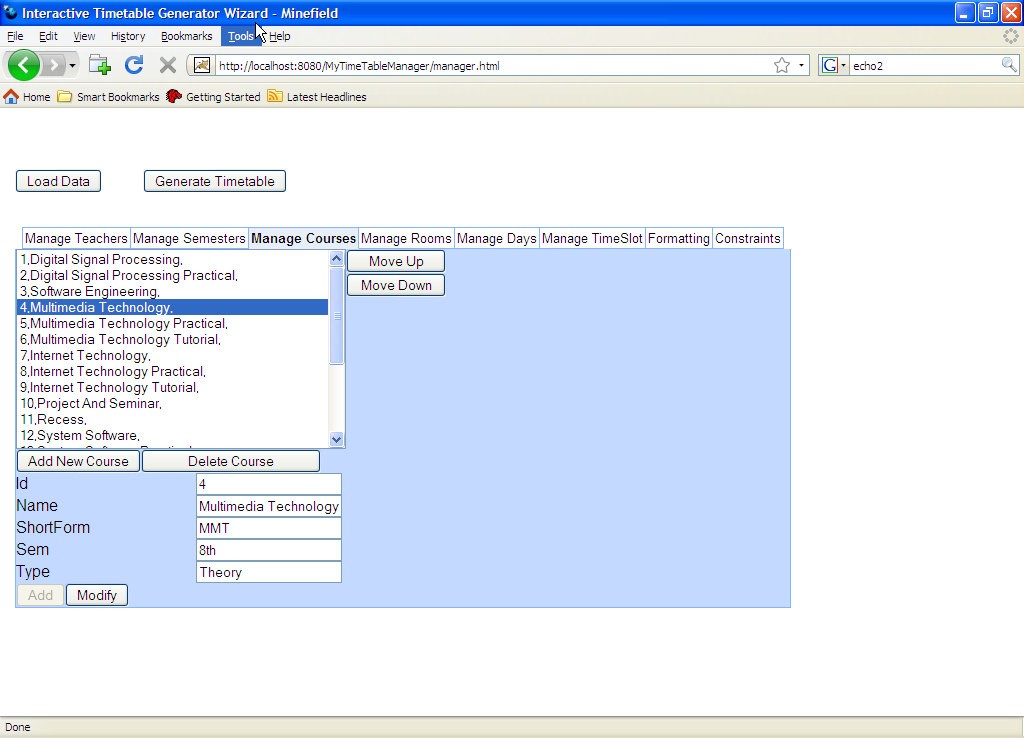
**1.Management of Teachers List**

****

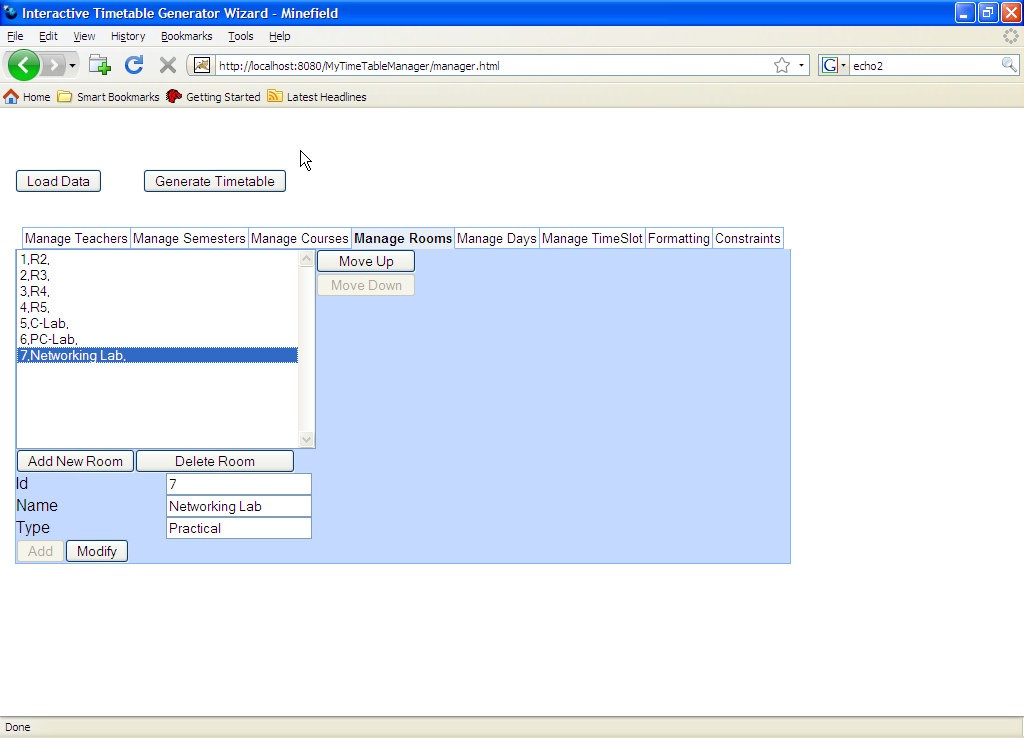
**2.Management of semester list**

****

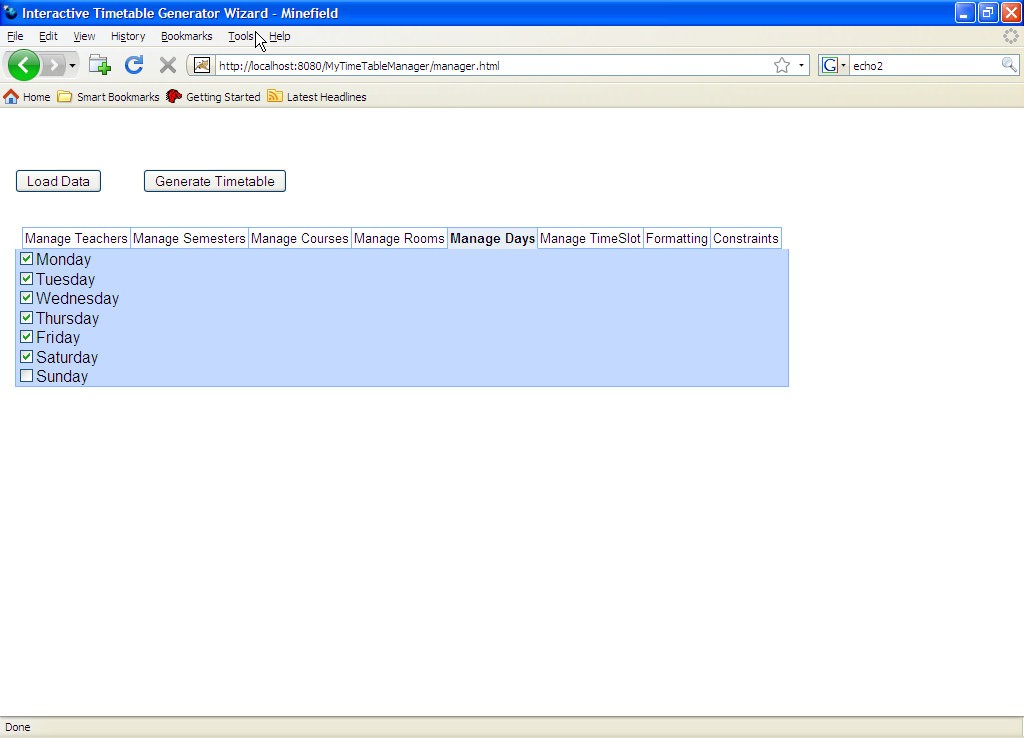
**3.Management of Course List**

****

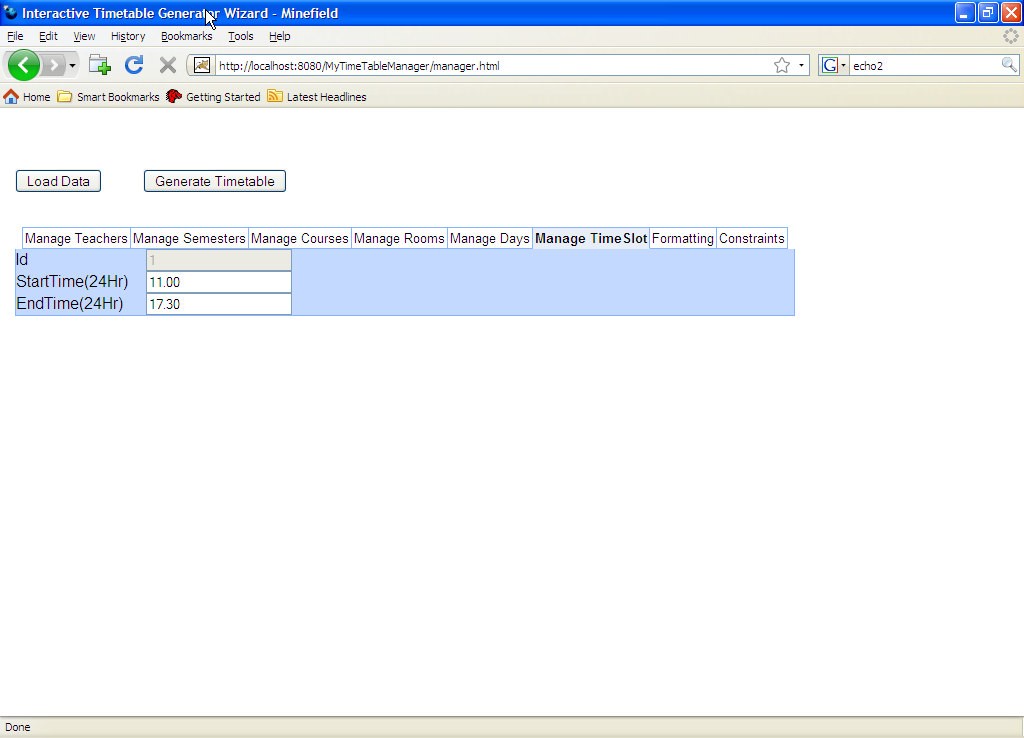
**4.Management of different rooms**

****

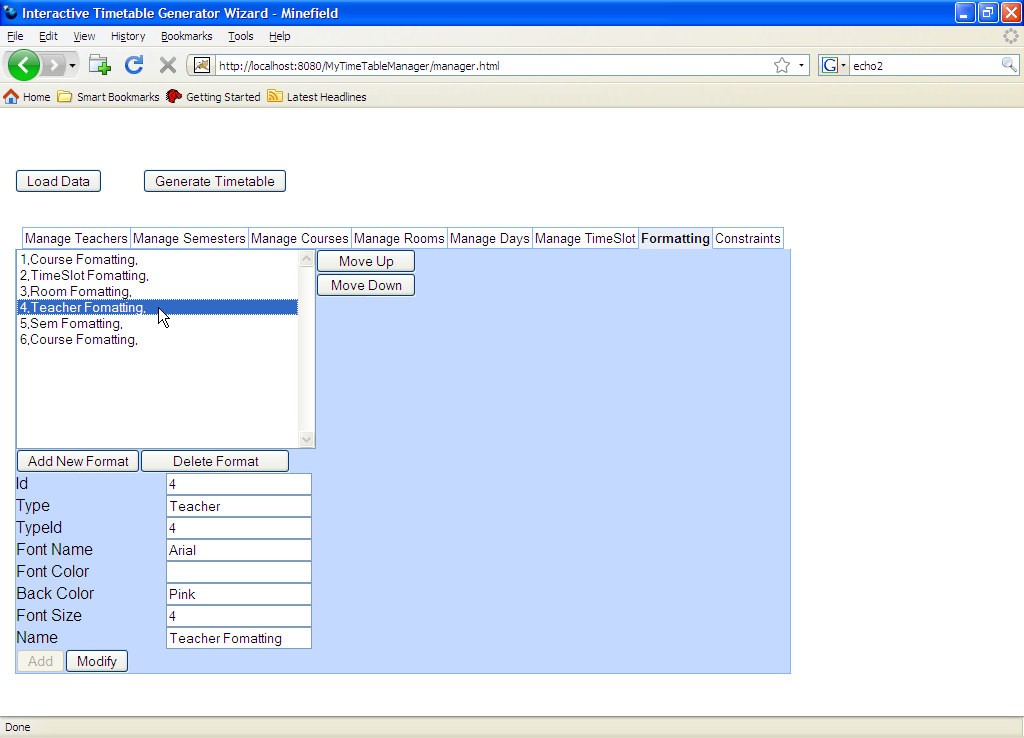
**5.Management of days**

****

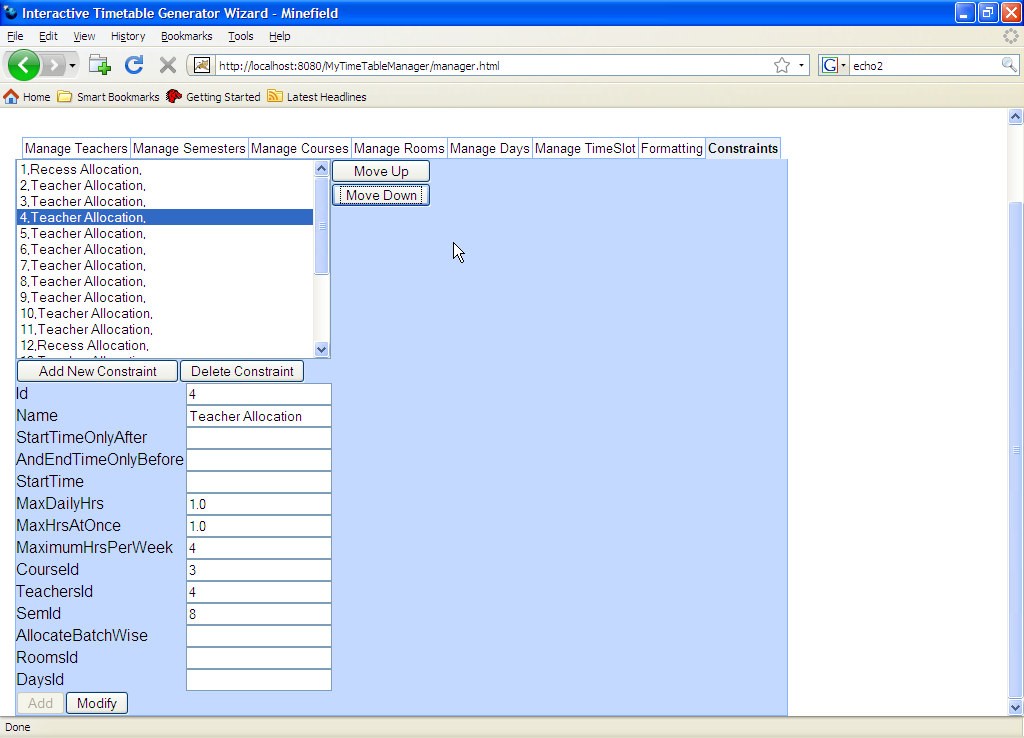
**6.Management of Time Slot**

****

**7.Management of formatting Elements**

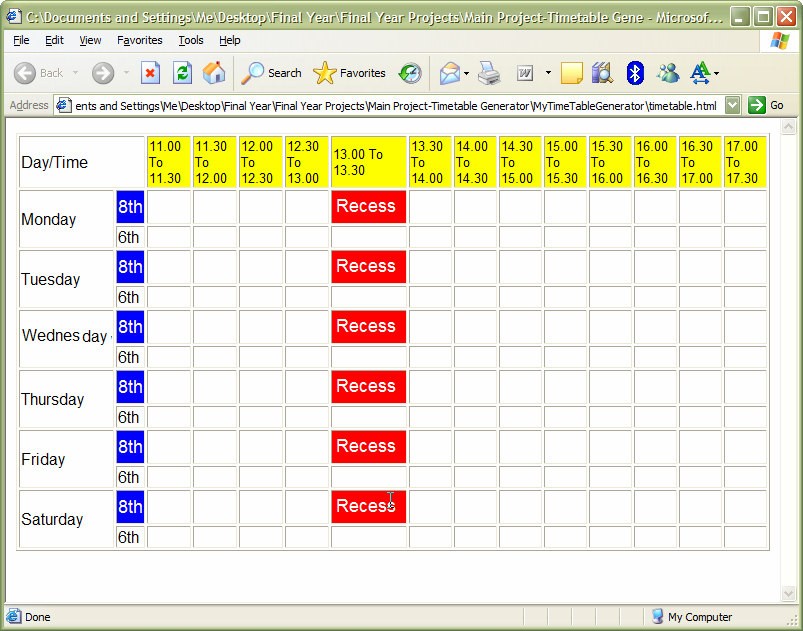
****

**8.Management of different rules, constraints while generation of Timetable**

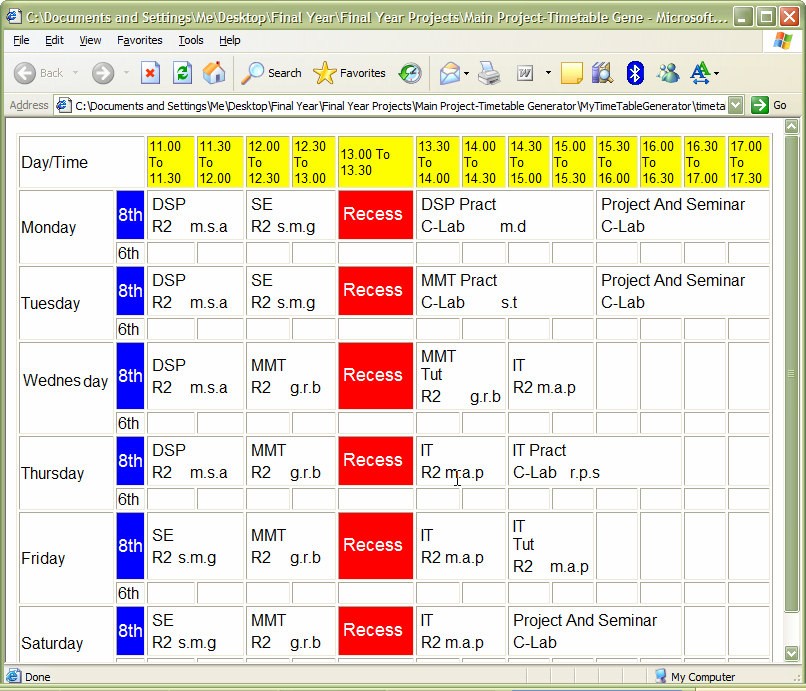
****

**Click on load data and then on Generate TimTable**

**9. Template of the Empty TimeTable before Generation**

****

**10.Final time table generated for the above selected elements and downloaded in destination directory**

****