Timetable Management Software

A Project Report

Submitted by

Aadesh G. Magare **111203032**

Sourabh G. Limbore **111203031**

in partial fulfillment for the award of the degree

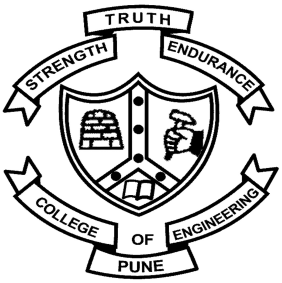
of

B.Tech Computer Engineering

Under the guidance of

Prof. Abhijit A. M.

College of Engineering, Pune



DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE-5

May, 2016

DEPARTMENT OF COMPUTER ENGINEERING

AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

Certified that this project, titled Timetable Management Software has been successfully completed by

Aadesh G. Magare	**111203032**
Sourabh G. Limbore	**111203031**

and is approved for the partial fulfillment of the requirements for the degree of "B.Tech. Computer Engineering".

SIGNATURE
Abhijit A. M.
Project Guide
Department of Computer Engineering and Information Technology, College of Engineering Pune, Shivajinagar, Pune - 5.

SIGNATURE
Vandana Inamdar
Head
Department of Computer Engineering and Information Technology, College of Engineering Pune, Shivajinagar, Pune - 5.

Abstract

Time table generation is a time consuming problem faced by many educational institutes. It belongs to the class of combinatorial optimization problems. We have implemented a semi-automated approach for solving this constraint heavy problem for educational institutes like College of Engineering Pune (COEP). It allows the user to make time table as per his/her choice while ensuring all constraints are satisfied and there are no conflicts. There are some existing solutions for this problem which are fully automated but difficult to use. We adopted a much simpler approach for solving it. We have developed a desktop application using object oriented programming paradigm with a user friendly interface.

It supports quantification of constraints, importing data from text files, mapping and filtering of data. Basic features such as cut, copy, paste are available. One can merge, unmerge no of cells as per requirements. Venue utilisation statistics and global warnings are readily available. Keyboard shortcuts are available for frequently used functions. Timetable can be exported in pdf, html and ods formats.

The code is available on GitHub under General Public License (GPLv3).
`https://github.com/Aadesh-Magare/Btech-Project-Timetable-Management`

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Every academic institution needs timetable for its functioning. Almost all college functions are computerized but timetable generation is still manually done in many institutes. Timetable scheduling is very complex problem involving many constraints. Scheduling timetable manually is time and effort consuming task. The problem is to manage the timetable such that it will satisfy all the constraints. The constraints include clashes of classrooms and timeslots, working hours of particular teacher or classrooms, number of lectures for particular subject, lunch breaks for class, etc.

We have implemented a semi-automated approach for solving this constraint heavy problem for institutions like COEP. It will allow users to make timetable in a semi-automated manner while ensuring that all the constraints are satisfied.

Limitations of fully automated approach:

- All constraints need to be ready at beginning.
- All data input is mandatory at beginning.
- No flexibility.
- Algorithm works mechanically, can't be modified.

- Personalization not possible.

Limitations of fully manual approach:

- All constraints need to be checked manually.
- Keeping track of data is very difficult.
- Mechanical tasks need to be done manually.
- Debugging timetable is really difficult.
- Takes a lot of time.

1.1 Problem Statement

To design and implement an application which can be used to manage the timetable of a department in an educational institute, ensuring that none of the constraints are violated and resources are used optimally while ensuring the flexibility of semi-automated approach.

1.2 Requirements

- The application should suggest create new or open project suggestion on startup.
- It should ask for titles (headers) when creating new project, which would be displayed at the top of timetable.
- Titles should be changeable dynamically.
- It should ask for basic constraints like working days per week, number of lectures per day, maximum workload for class and lecture start time for the institution at the time of creation of timetable.

- There should be three tabs, *Class, Venue and Teacher* which will show classwise, venuewise and teacherwise timetable respectively.
- Timetable should be displayed in a grid i.e. a table having cells.
- Cells in a grid should contain abbreviated names of teacher, venue, class and subjects. (and batches if applicable)
- It should show pop-up box for input in cells, having dropdown style choices for teacher, venue, class and subject.
- It should support typing in cell directly.
- There should be user-friendly cell formatting (colours and styling).
- It should support merge, unmerge of cells.
- It should have support for batches in class for e.g. *syce-b1, syit-b1* etc.
- There should be a panel for easy scrolling on left side of screen.
- Standard keyboard short-cuts like copy, paste, merge should work.
- Project file should be opened via command line argument.
- It should have an ability to manage data of *teachers, venues, classes, subjects* at runtime and import it from files.
- It should have ability to specify individual *teacher workload.*
- It should support *teacher-subject, venue-class and teacher-class* mapping and importing from file.
- *Subject* list in the pop-up box should filter as per *class-subject* mapping.
- While typing in pop-up box, suggestions should be shown.

- Venue/class capacity, number of hours for a subject, workload for a teacher should be changeable at runtime.
- Venue utilisation statistics should be shown.
- There should be a check constraints option in menu.
- It should export timetable in *ods, pdf, html* formats.
- It should have ability to define styling for *ods* document.
- It should show keyboard short-cuts in menu.

Chapter 2

Literature review

2.1 Existing Solutions

There are some existing softwares which solve the problem of timetable scheduling. Given below is the list of some of the well known softwares.

2.1.1 FET

This is an open-source software for automatic timetable generation. FET has some features like automatic generation of timetable based on provided constraints data, exporting the generated timetable in pdf, csv, html formats. It runs on most of the systems including GNU/Linux, Windows, MacOS. FET is too complex for new users. Large amount of data is required initially to generate timetable. There's no flexibility in timetable generation.

2.1.2 aSc Timetable

aSc Timetable is shareware software for automatic timetable generation with user-friendly GUI. It supports automatic generation of timetable with manual adjustments. The main issue with aSc Timetable is, its neither freeware nor open-source. It is not available on GNU/Linux.

2.1.3 Mimosa

Mimosa scheduling software is user-friendly and can be used for any organization since it focuses on core challenges of scheduling. It runs on most of the systems including GNU/Linux, Windows, MacOS. Mimosa is neither freeware nor open-source.

Features	**FET**	**aSc Timetable**	**Mimosa**
Automatic/Manual	fully automatic semi-automatic	both	both
Platform	Windows GNU/Linux, Mac	Windows Mac	Windows GNU/Linux, Mac
Import data from files	yes	yes	yes
Export	html, xml, csv	html, xml, csv	html, xml, csv
Open Source	yes	no	no

Table 2.1: Comparison of existing solutions

2.2 Proposed Solution

The main aim is to simplify the process of timetable management. The software won't automatically generate timetable but help the user manage timetable. The solution considers both soft and hard constraints. Soft constraint violation generates warning whereas hard constraints can't be violated. It supports dynamic checking of constraints. Timetable can be easily exported in popular formats like pdf, html and ods.

Features	**Proposed Solution**
Automatic/Manual	semi-automatic
Platform	Windows, GNU/Linux
Import data from files	yes
Export	html, **ods**, pdf
Open Source	yes

Table 2.2: Proposed solution

Chapter 3

System design criteria

3.1 Requirements

The software is very lightweight and runs on a machine with bare minimum configuration. Following are the minimum software and hardware requirements for the program.

3.1.1 Hardware requirements (Same as for Python platform)

- Memory/RAM : 512MB.
- Hard Disk Space : 1GB.
- Processor : Intel Pentium 4 or later.

3.1.2 Software requirements

- Operating system: GNU/Linux or Windows
- Python (v2.7)
- Libraries: Wxpython, Wxwidgets
- Python modules: pickle, ezodf, pdfkit

3.2 Constraints

Constraints are the rules that must be satisfied by the timetable. Hard constraints are the ones which can not be violated. Soft constraints can be violated but generate warnings.

3.2.1 Hard constraints

1. No clashes in *teacher* / *venue* / *class* / *batch* timetable.
2. Allocated hours of a subject in timetable should not exceed the no of hours allocated to it by the institute.

3.2.2 Soft constraints

1. *Teacher's* workload should not exceed maximum(max) *teacher* workload.
2. *Venue* capacity should be greater than or equal to *class* capacity.
3. Compulsory lunch break for each *class* / *batch*.
4. Allocated hours for a *subject* in timetable should not be less than number of hours allocated to it by the institute.

3.3 Approach of design

Teacher, Venue and Classes are the three classes that are central to the working of application. Every individual teacher, venue or class is an instance of one of these classes. User needs to input data i.e. list of teachers, venues, classes and subjects. This data can also be imported from external files. Given below is the format of data that should be followed.

Teacher Name	Abbreviation	WeeklyMaxLoad	DailyMaxLoad
Abhijit A M	AM	25	5
Satish Kumbhar	SSK	28	5
Jibi Abraham	JA	22	5

Table 3.1: Teacher Table

Class Name	Abbreviation	Size
SYComp	SYC	75
TYComp	TYC	75

Table 3.2: Class Table

Subject Name	Abbreviation	No of hours
DataStructure	DSA	4
MathI	M1	4

Table 3.3: Subject Table

Venue Name	Abbreviation	Capacity
Academic Complex 201	AC201	120
Academic Complex 302	AC302	120

Table 3.4: Class Table

When user adds an entry in any of the timetable it is verified against all existing entries for violation of constraints. The change is allowed only after verification, failing which all changes are discarded. During this, constraints

like teacher's work load, number of hours allocated to the subject are also checked. Venue capacity is verified for every venue-class pair. After an entry is successfully added teacher work load and subject hour count is incremented. When a hard constraint is violated all changes are discarded and an error is shown immediately. When a soft constraint is violated the warning is added to global warning section. User can view or manage global warnings any time from the menubar.

3.3.1 List of Implemented Features

- Support for batches.
- Individual teacher work load.
- Venue-class capacity checking.
- Venue utilization statistics.
- Check all constraints function.
- Pop-up box for input with dropdown suggestions.
- Mapping and filter data in pop up box.
- Direct typing in cell for input.
- Mandatory lunch breaks.
- Cut-paste / copy-paste.
- Support for standard keyboard functions - enter, delete, backspace.
- Import data from files.
- Export *pdf*, *html* and *ods*.

- Open project with cmd argument.
- Save / open project.
- Merge / unmerge selected cells.
- Delete entry.
- Warning list for soft constraint violation.
- Number of hours for subject, teacher workload and venue capacity / class size changeable at runtime.

Chapter 4

Implementation

4.1 Technologies used

4.1.1 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Python supports multiple programming paradigms, including object oriented, imperative and functional programming or procedural styles. Entire application is written in python including both front end and back end.

4.1.2 wxPython

wxPython is a GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface with ease.

wxPython is used for all the UI components. *wx.grid*, *wx.StaticText*, *wx.Panel*, *wx.ComboBox* are some of the important classes used in front end.

4.1.3 Wxwidgets

Wxwidgets gives you a single, easy-to-use API for writing GUI applications on multiple platforms that still utilize the native platform's controls and

utilities. *WxPython* is a python wrapper over *Wxwidgets*. It makes extensive use of *Wxwidgets* under the hood.

4.1.4 Python modules

- **Pickle :** *pickle* module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.
- **Ezodf :** *ezodf* is a Python package to create new or open existing OpenDocumentFormat files to extract, add, modify or delete document data.
- **Pdfkit :** *wkhtmltopdf* python wrapper to convert html to pdf using the webkit rendering engine

4.2 Front-end

GUI consist of a grid i.e. a table which is used in typical timetables. Basic constraints and header information can be provided through a window. WxPython provides classes *wx.grid* class for the table UI and *wx.Dialog* class for windowing purposes.

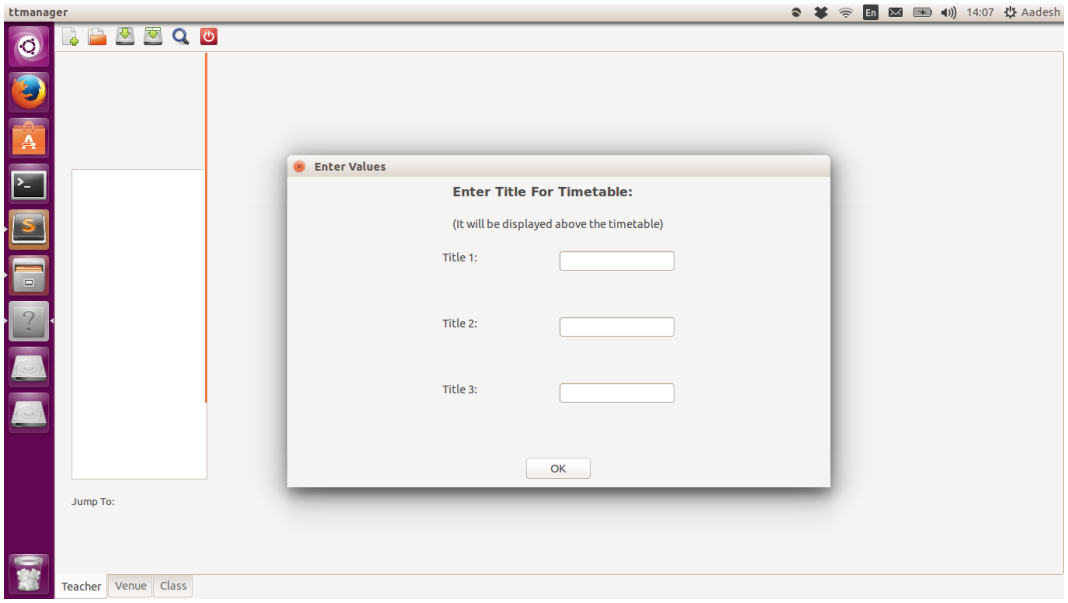


Figure 4.1: Header Information

Separate windows are available to enter teacher, venue, class and subject data. A window to enter teacher-subject, teacher-class, venue-class mappings is available. The class *wx.ListCtrl* is used for all list related UI.

An example of teacher data entry.

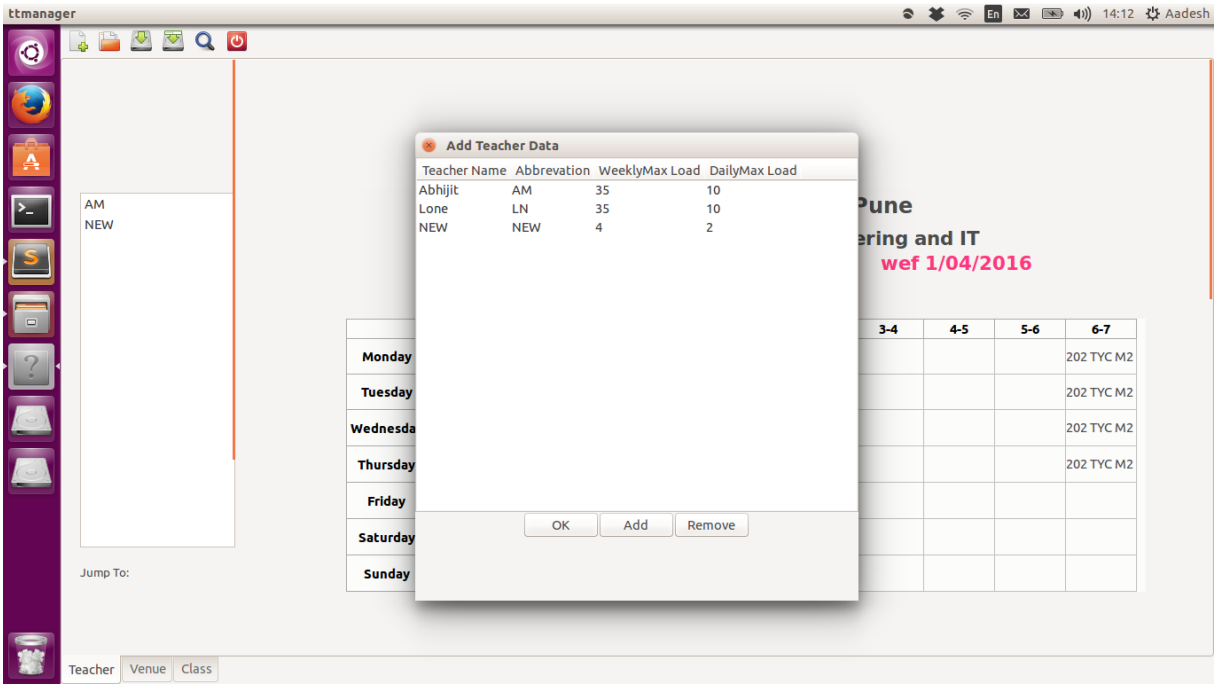


Figure 4.2: Teacher data entry

An example of venue data entry.

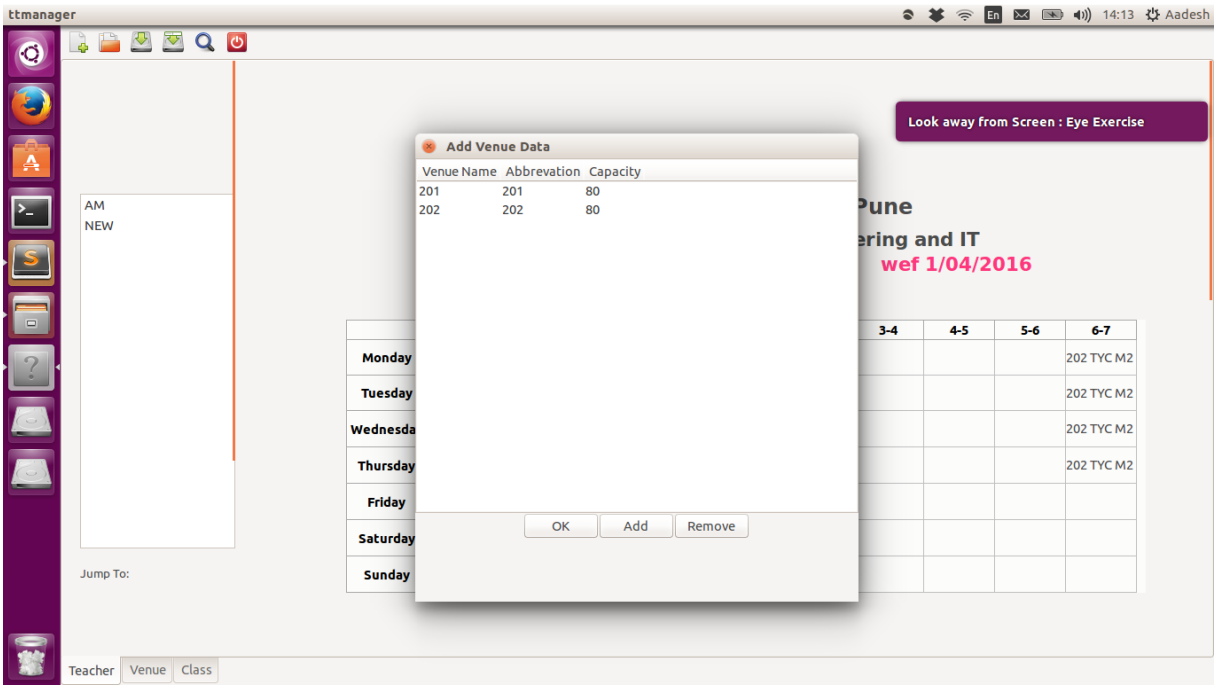


Figure 4.3: Venue data entry

An example of venue-class mapping entry.

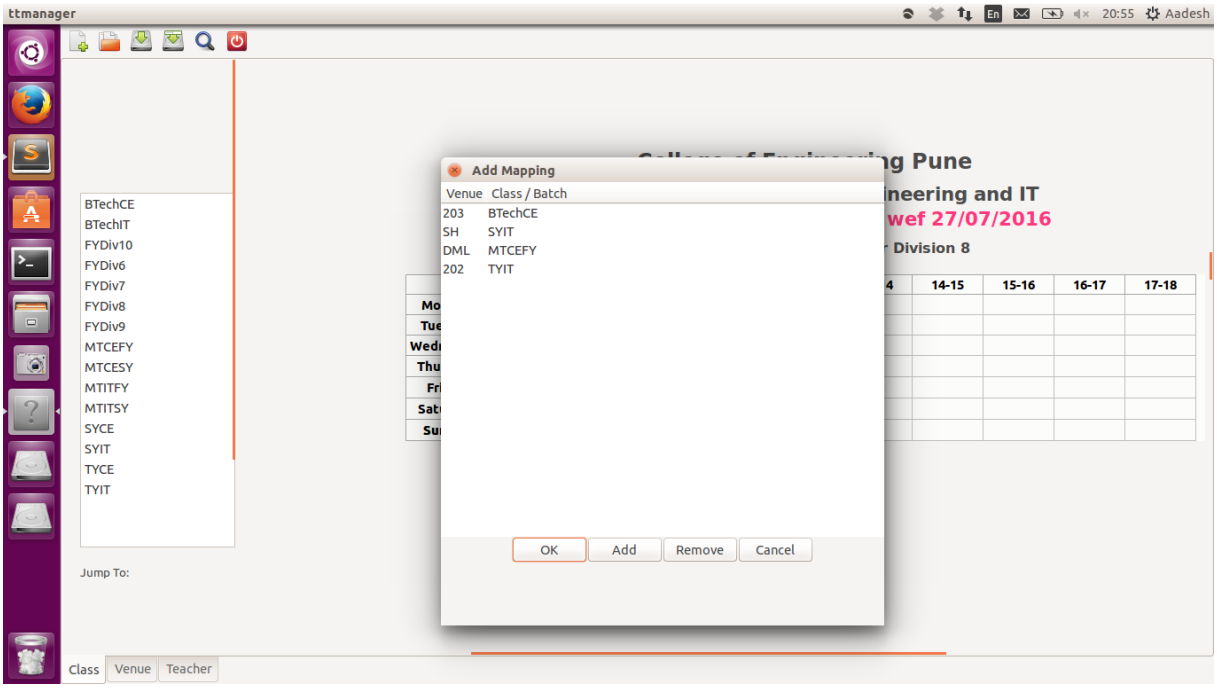


Figure 4.4: Venue-class mapping

An example of importing data from file.

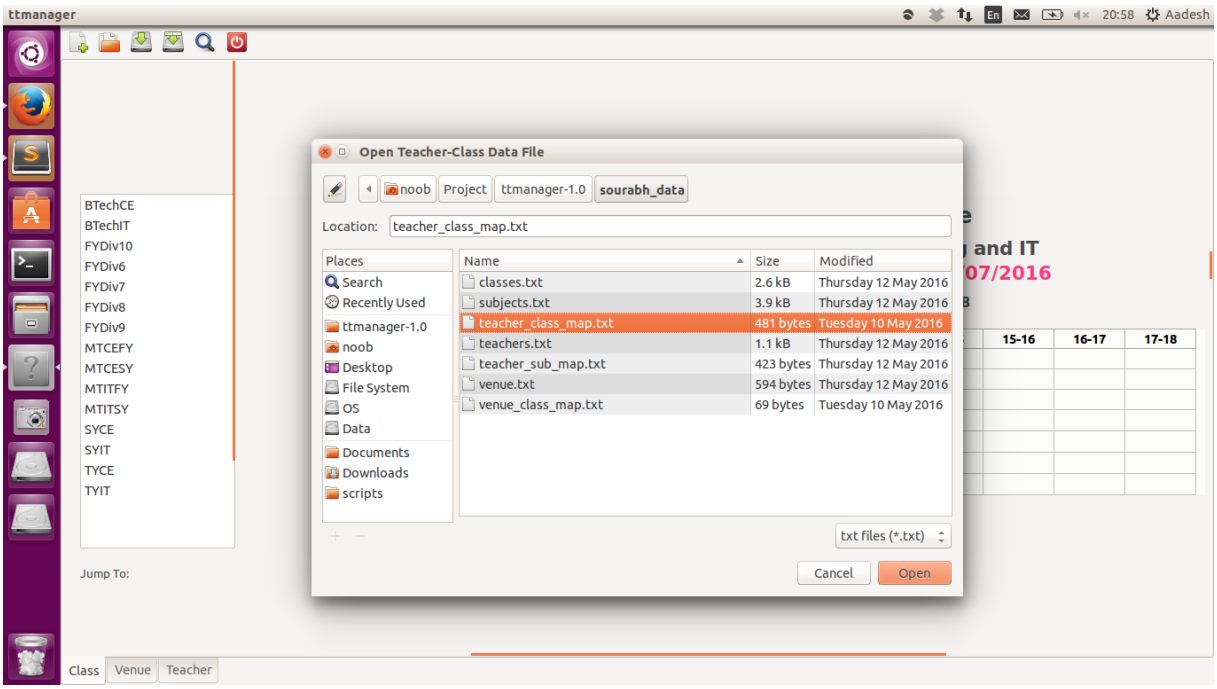
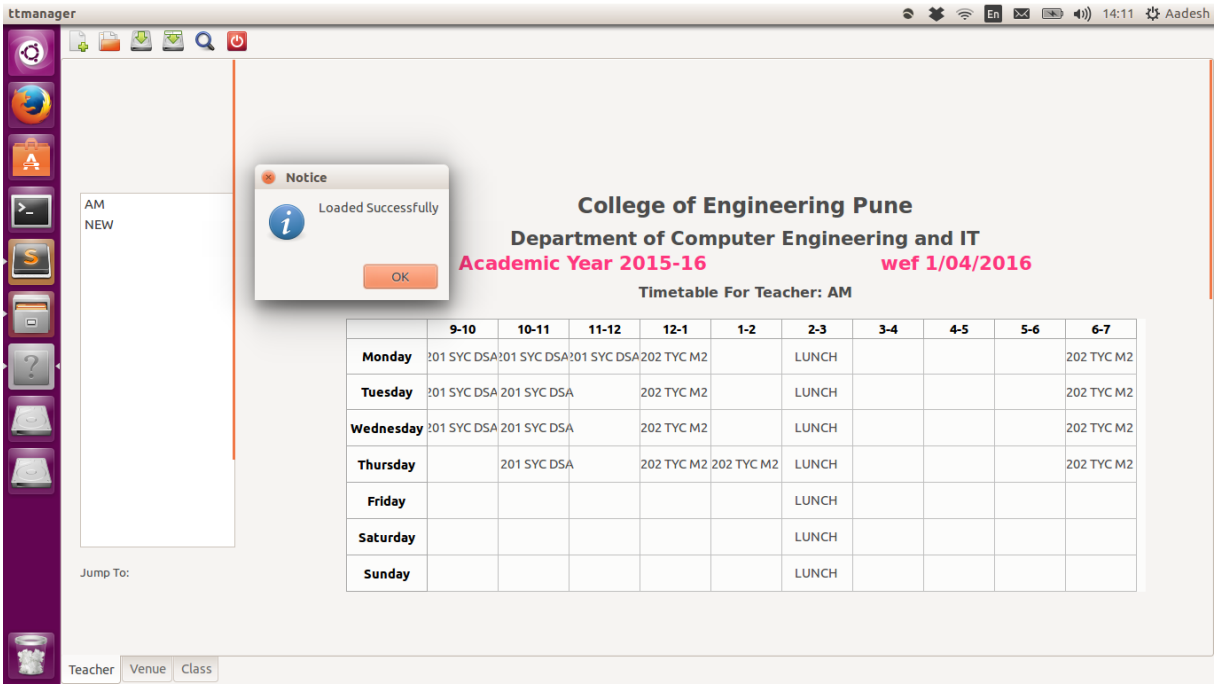


Figure 4.5: Importing data from file

User can open/save partial timetable.



	9-10	10-11	11-12	12-1	1-2	2-3	3-4	4-5	5-6	6-7
Monday	201 SYC DSA	201 SYC DSA	201 SYC DSA	202 TYC M2		LUNCH				202 TYC M2
Tuesday	201 SYC DSA	201 SYC DSA		202 TYC M2		LUNCH				202 TYC M2
Wednesday	201 SYC DSA	201 SYC DSA		202 TYC M2		LUNCH				202 TYC M2
Thursday		201 SYC DSA		202 TYC M2	202 TYC M2	LUNCH				202 TYC M2
Friday						LUNCH				
Saturday						LUNCH				
Sunday						LUNCH				

Figure 4.6: Opening of saved timetable

Pop-up boxes are used to take input from the grid. Data gathered from all these UI components is fed to back-end to initialise the timetable. Left-Panel provides easy to navigate mechanism. Double-clicking on an item scrolls to that timetable on current screen. Wxpython provides *wx.lib.scrolledpanel* class for scrollable panels.

4.3 Back-end

All data is manipulated by python scripts. *Teacher, Venue, and Classes* are the User-defined class. Every teacher, venue and class involved in the timetable is an instance of these classes. These instances are stored in global lists *all_teachers*, *all_venues* and *all_classes* respectively. These classes have methods such as can add, *add_entry*, *remove_entry* which are slightly different for each class.

```
if teacher slot available then
    if teacher workload less than maximum workload then
        increment teacher workload;
    else
        add warning;
    end
    if venue slot available then
        if venue capacity less than class size then
            show warning;
        else

        end
        if class slot available then
            add entry to timetable;
            return;
        else
            remove teacher entry;
            remove venue entry;
            throw error and return;
        end
    else
        remove teacher entry;
        throw error and return;
    end
else
    throw error and return;
end
```

Algorithm 1: Algorithm to insert a entry:

Algorithm to add new teacher / venue / class:

if *object corresponding to name exists* **then**

    return it;

else

    create instance of Teacher/Venue/Class depending upon type of Data;

    push the instace into the global list of objects;

    return the created object;

end

Algorithm to remove a entry:

Get the objects corresponding to the entry;

if *object type is teacher or venue* **then**

    empty that slot;

else

    if *entry is related to whole class* **then**

        empty that slot;

    else

        remove entry corresponding to that batch from the slot;

    end

end

Decrease the workload count;

Algorithm to find venue utilization:

Get all venue objects from *globaldata.all_venues*;

while *there is next object* **do**

> get next object;
>
> check all lecture slots on each working day;
>
> for each non-empty entry increment count corresponding to that venue;
>
> find utilization relative to total available time slots;

end

Algorithm for ods export:

if *styling_reference.ods file exists* **then**

> Use the *styling_reference.ods* file for styling;
>
> Get save file path from user through file select box.;
>
> Create three sheets named teacher, venue and class using ezodf and add to the notebook.;
>
> Populate the spreadsheet for each teacher, venue and class by traversing the matrix associated with it.;
>
> Use styling classes from the reference file;
>
> Save the file in user specified path;

else

> show error and return;

end

Algorithm to verify constraints:

for *each class in globaldata.all_classes* **do**

  if *every batch as lunch break on each working day* **then**

  else

    add warning to global warning list;

  end

  if *workload of class is within limits* **then**

  else

    add warning to global warning list;

  end

  if *no of hours for each subject satisfied* **then**

  else

    add warning to global warning list;

  end

end

for *each teacher in globaldata.all_teachers* **do**

  if *lunch break on each working day* **then**

  else

    add warning to global warning list;

  end

end

Teacher and Classes class have additional methods to add / remove lunch. A lunch entry is treated differently than normal lecture entry. The Classes class has another special method valid_lunch_break, it is used to verify that all classes and batches have valid lunch breaks.

Algorithm to insert lunch entry:

if *class slot or teacher slot available* **then**

    add entry to class or teacher;

else

    throw error;

    return;

end

A wrong entry causes Exception. In that case, its handled and its affects are undo-ed. There are 4 types of Exceptions ExistingEntry Exception, ExtraWorkLoad Exception, LimitForSubject Exception and DailyWorkLoad Exception.

1. **ExistingEntry Exception :** When the current entry clashes with existing timetable.

2. **ExtraWorkLoad Exception :** When the current entry exceeds the weekly workload of teacher.

3. **LimitForSubject Exception :** When the current entry exceeds the weekly no of lectures for a subject.

4. **DailyWorkLoad Exception :** When the current entry exceeds the daily workload of teacher.

Hard constraint violation is not permitted where as soft-constraint violation causes warning which is saved in global warnings section. These global warnings are stored in file on filesystem. User can remove warnings from the file (which he/she chose to ignore). The warnings can be refreshed which causes the system to check all constraints and build the warning list again.

pickle module is used to save the timetable on filesystem. It allows to save / open the timetable by dumping the in-memory contents on the disk. Exporting the timetable in ods, pdf and html formats is possible. Python modules *ezodf* and *pdfkit* are used to export the timetable. *styling_reference.ods* is used as a reference style for the ods export.

Chapter 5

Conclusion

We have implemented a software for timetable management for educational institutes. This software is helpful to manage the timetable of a department in an educational institutes while ensuring no violation of constraints and optimal use of resources. It provides

- User friendly GUI.
- Ability to quantify constraints.
- Import data from external files.
- Dynamic checking of constraints.
- Partial saving and opening of timetable.
- Warning's window - manage warnings at one place.
- Export timetable in ods, pdf, html format.

Chapter 6

Future Scope

Further work can be done in following areas:

- Port to web-platform - to avoid installation issues and support collaborative management.
- Automatic timetable generation.
- Improvement in GUI.
- Specification of constraints.

Chapter 7

Diagrams

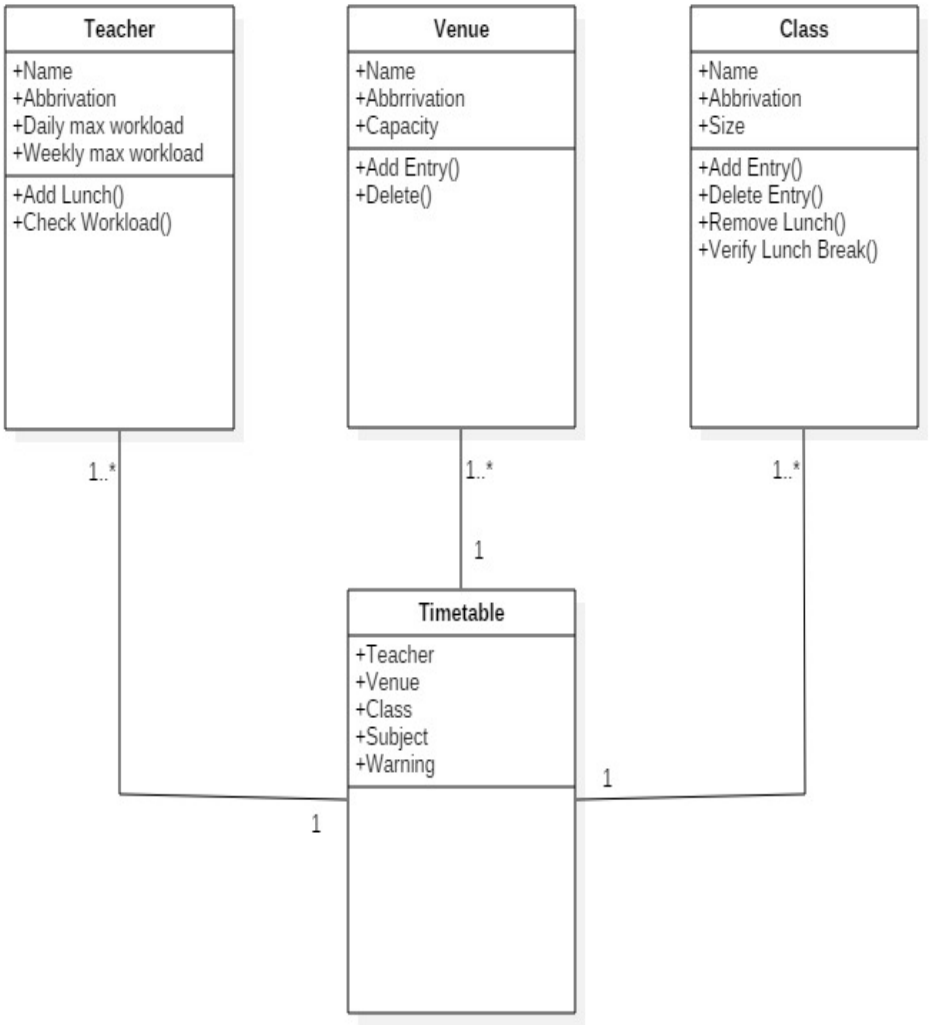


Figure 7.1: Class Diagram

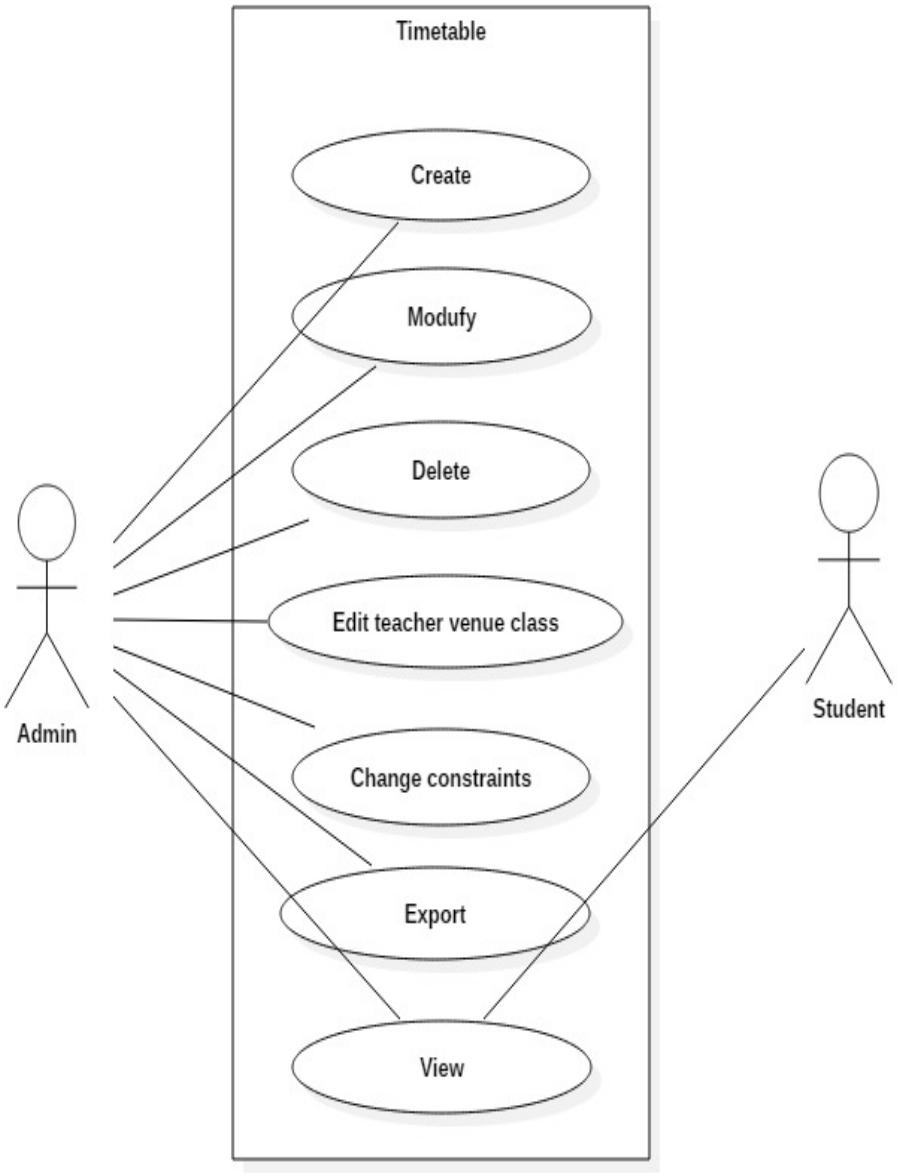


Figure 7.2: Use Case Diagram

Bibliography

[1] Python documentation,

https://docs.python.org/

[2] wxPython,

http://www.wxpython.org/

[3] wxwidgets documentation,

https://www.wxwidgets.org/docs/tutorials/

[4] ezodf: Python Package,

https://pypi.python.org/pypi/ezodf

[5] pickle: Python module,

https://docs.python.org/2/library/pickle.html

[6] pdfkit: Python module,

https://pypi.python.org/pypi/pdfkit

[7] GitHub link of project,

https://github.com/Aadesh-Magare/Btech-Project-Timetable-Management