USER MANUAL

Time-Table Assist Tool

Developed by Namrata Malkani, Aashima, Akhil Rajput

Sections ----------

Getting Started ----------

1.1 Introduction

Time-Table assist Tool is a software built to help you in making the institute timetable. It does not create the timetable but helps you in avoiding the various clashes which may skip the human eye. The user interface enables you to make the timetable and each time you make an addition, the various constraints are checked by querying the database. If the entry does not cause any clash, it is added into the time slot specified, else an error message is displayed. You can also modify the main database tables according to your needs.

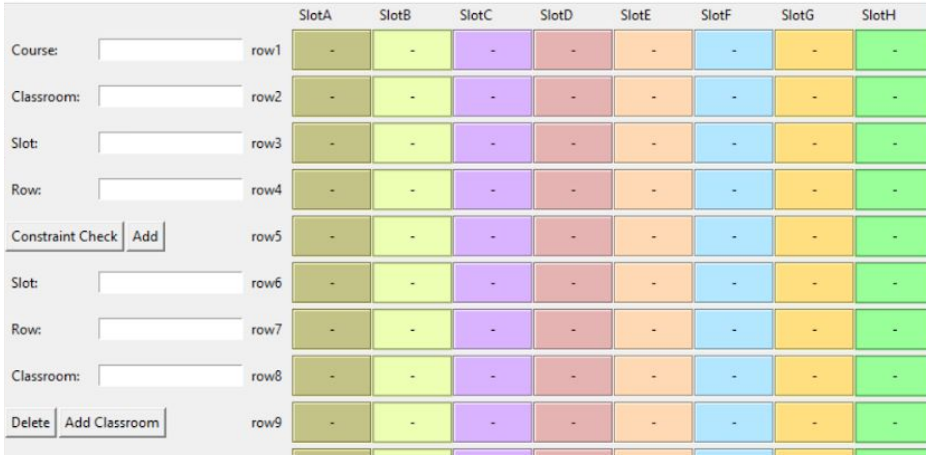


Figure- Partial Snapshot of gui

There are 8 time slots named A-H and 20 class rooms. Therefore, a slot can have maximum 20 courses and no two courses in a given slot can have the same classroom. A faculty can teach only one course in a slot. The user interface enables you to allocate time-slots and classroom to a course. Besides these obvious constraints, there are core courses and basket clashes. For a given discipline (Example CS/ME/EE) in a given semester, no two core courses can be put in the same slot. No two courses belonging to the same basket can be put in the same slot. All the data is stored in form of tables in the database. You need not worry about the database creation but you need to make sure that the data and the files provided are correct. The details about database, raw data and features provided by the user interface are mentioned in following sections.

1.2 System Requirements and Installation

You system must have:

1. Python (mostly preinstalled, preferably version 3.6 and above)
2. Tkinter python module
3. Numpy python module
4. SQLite3

1.2.1 Ubuntu

Pre-installation checks:

Go to your etc/apt/ folder. You will find the apt.conf file. This file needs to be edited in order to download or install software on proxy. If you do not find that file, create it and edit it accordingly.

- Open terminal and type:
- `cd /etc/apt`
- `sudo nano apt.conf`
- Append the lines

For examples:
Proxy gateway: https://gateway.iitmandi.ac.in port: 8080
Lines appended:

```
Acquire::http::Proxy "http://gateway.iitmandi.ac.in:8080";
Acquire::https::Proxy "https://gateway.iitmandi.ac.in:8080";
Acquire::ftp::Proxy "ftp://gateway.iitmandi.ac.in:8080";
```

Now that you are done with the checks, you can proceed to the installations.

Installations:

- **Python**
  Python comes pre-installed with ubuntu. That being said, our users of ubuntu have it pre-installed.
  Open the terminal using `ctrl+alt+t`
  Next, check the version of python using the command:
  ```
  python3 --version
  python --version
  ```
  If on the first command you get "No such package found", it means you do not have python3 installed. Also if the second command shows some python2 version, it again means you have outdated python, which means it needs updation.

Updating python on ubuntu:
The good thing with ubuntu is that package manager provides with version 3 which gets installed on running a single command.

```
sudo apt-get update
sudo apt-get install python3
```

One must just remember to launch their scripts with python3.
Example:

```
python3 code.py
```

- **Tkinter**
  Almost all the linux versions come with Tkinter installed already. And if you have correctly followed the steps above then the installation of python3 must definitely put Tkinter in your system.

- **Numpy**
  Numpy is preferably installed with pip but it can cause unnecessary hassle for the otherwise technically inept user and may require some debugging and googling. Hence it is best to proceed without going through pip.

```
sudo apt-get install python-numpy
```

- **SQLite**
  It is our database management system. Inbuilt in the linux package manager, hence easily installed by the command:

```
sudo apt-get install sqlite
```

1.2.2 Windows

Pre- installation checks:

Proxy configurations according to the network is necessary. Hence check your proxy settings in the system and see if they are in accord with the network proxy. If they are not set the proxy and port number by going to the ‘System proxy settings’.

Installations:

- **Python**

Python doesn't come pre-installed with windows systems.
Installing and using Python on Windows 10 is very simple. The installation procedure involves just three steps:

1. Download the binaries

   Open the [official Python website]() in your web browser. Navigate to the Downloads tab for Windows.

   Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version.

   Click on the link to download **Windows x86 executable installer** if you are using a 32-bit installer. In case your Windows installation is a 64-bit system, then download **Windows x86-64 executable installer**.

2. Run the Executable installer

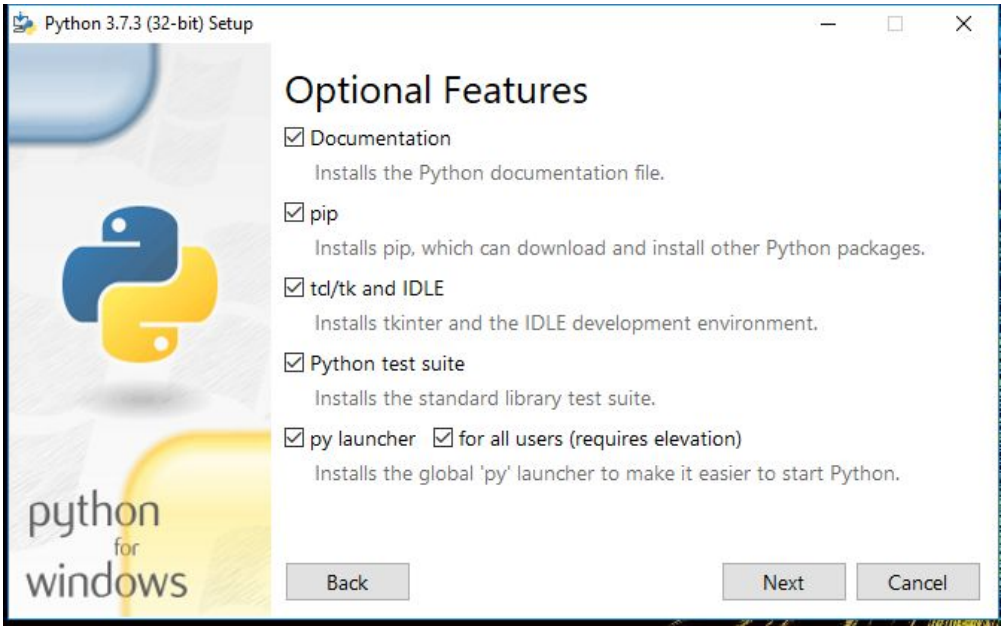
   Once the installer is downloaded, run the Python installer.

   Check the **Install launcher for all users** check box. Further, you may check the **Add Python 3.7 to path** check box to include the interpreter in the execution path.

Select **Customize installation**.

Choose the optional features by checking the following check boxes:

1. Documentation
2. pip
3. tcl/tk and IDLE (to install tkinter and IDLE)
4. Python test suite (to install the standard library test suite of Python)
5. Install the global launcher for `.py` files. This makes it easier to start Python
6. Install for all users.



Click **Next**.This takes you to **Advanced Options** available while installing Python. Here, select the **Install for all users** and **Add Python to environment variables** check boxes.
Optionally, you can select the **Associate files with Python**, **Create shortcuts for installed applications** and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step.

After selecting the Advanced options, click **Install** to start installation.

3. Add Python to PATH environment variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the

command line. Since you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step.

- **Tkinter**

  If you have followed the above correctly, then you have Tkinter installed as it comes pre installed with most python3

- **Numpy**

  Try the following command to install:

  ```
  python -m pip install numpy --user
  ```

  There can be various reasons why a command wouldn't work and most of them being related to incorrect proxy settings or being system dependent in some way. This one has worked well for most systems after installation of python3.

- **SQLite**

  Follow the following:

  1. Go to www.sqlite.org
  2. Go to Download tab. Click.
  3. Scroll down to look for pre compiled binaries for Windows.
  4. You need to download the one that says sqlite-tools-win32-x86-3190300.zip
  5. Save the zip file in any folder. Right click and extract the files. The extracted folder contains 3 .exe files.
  6. We want to be able to open any directory and run sqlite on the command prompt on any dataset in the directory. In order for this to happen, one needs to include the file path of SQLite3 in the environment variable setting of windows.
  7. Create a folder within windows (C:) called "sqlite" and copy the 3 executables we had in the extracted folder, to this folder.
  8. Now this folder called sqlite needs to be included in the list of environment variables. Copy the folder path.
  9. In the windows search, find "edit the system environment variables"
  10. Settings -> Path -> Edit -> New
  11. Paste the path. Click OK.

12. Now sqlite has been installed and can be accessed in any folder on the system.

Setup ----------

2.1 Raw Data

2.1.1 Description

The raw data is user dependent and it is your responsibility to make sure that it is correct. The following files contain raw data that is imported to build the database:

1. *courses.csv*
   This files contains the entire course list for the particular semester provided by the institute.
2. *teachers.csv*
   This file contains the instructors short name (as used in the course list) and their full names.
3. *class.csv*
   This file contains the class names and the maximum strength they can accommodate.
4. *basket.csv*
   This file contains the different baskets as provided by various faculty. Each column corresponds to a particular basket. A basket cannot have more than 8 courses.

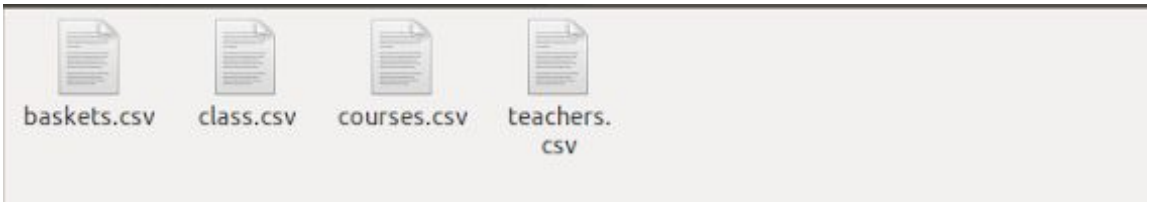


Figure- .csv files in directory

If you wish to make changes to any of these *.csv* files:

1. Enter correct data
2. Follow the same format

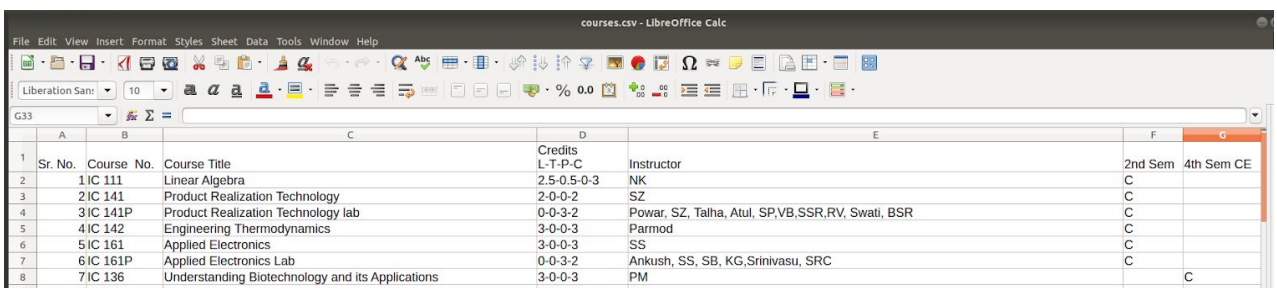
3. Save the changes
4. Button **Reload Database**

2.1.2 Format

1. *courses.csv*

S.no.	Course No	Course Title	Credits L-T-P-C	Instructor	Something	--

   a. The first row must contain headers. The name of headings can be anything.
   b. Total number of columns = 5 + any number
   c. The file must follow the format as shown:
      1st column = S.no.
      2nd column = Course Code
      3rd column = Course Title
      4th column = Credits distribution
      5th column = Instructor Short Name
      Remaining columns = contain anything
      Remaining number of columns = Calculated Dynamically



Sr. No.	Course No.	Course Title	Credits L-T-P-C	Instructor	2nd Sem	4th Sem CE
1	IC 111	Linear Algebra	2.5-0.5-0-3	NK	C	
2	IC 141	Product Realization Technology	2-0-0-2	SZ	C	
3	IC 141P	Product Realization Technology lab	0-0-3-2	Powar, SZ, Talha, Atul, SP,VB,SSR,RV, Swati, BSR	C	
4	IC 142	Engineering Thermodynamics	3-0-0-3	Parmod	C	
5	IC 161	Applied Electronics	3-0-0-3	SS	C	
6	IC 161P	Applied Electronics Lab	0-0-3-2	Ankush, SS, SB, KG,Srinivasu, SRC	C	
7	IC 136	Understanding Biotechnology and its Applications	3-0-0-3	PM		C

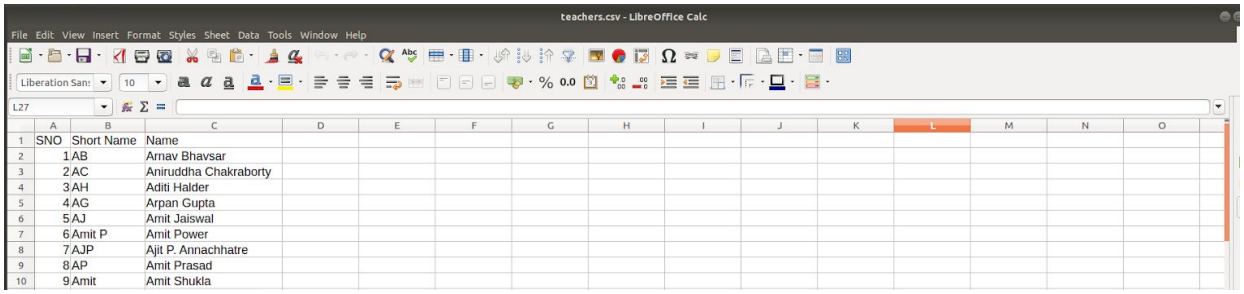
Figure- Part of courses.csv

2. *teachers.csv*

S.no.	Short Name	Full Name

   a. The first row must contain headers. The name of headings can be anything

b. Total number of columns = 3
c. The file must follow the format as shown:
   1st column = S.no.
   2nd column = Short Name
   3rd column = Full Name



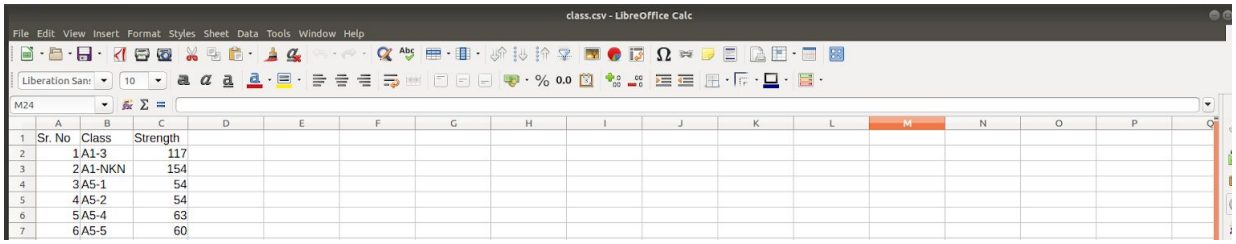
	A	B	C
1	SNO	Short Name	Name
2	1	AB	Arnav Bhavsar
3	2	AC	Aniruddha Chakraborty
4	3	AH	Aditi Halder
5	4	AG	Arpan Gupta
6	5	AJ	Amit Jaiswal
7	6	Amit P	Amit Power
8	7	AJP	Ajit P. Annachhatre
9	8	AP	Amit Prasad
10	9	Amit	Amit Shukla

Figure- Part of teacher.csv

3. *class.csv*

S.no.	Class Name	Class Strength

a. The first row must contain headers. The name of headings can be anything
b. Total number of columns = 3
c. The file must follow the format as shown:
   1st column = S.no.
   2nd column = Class Name
   3rd column = Class Strength



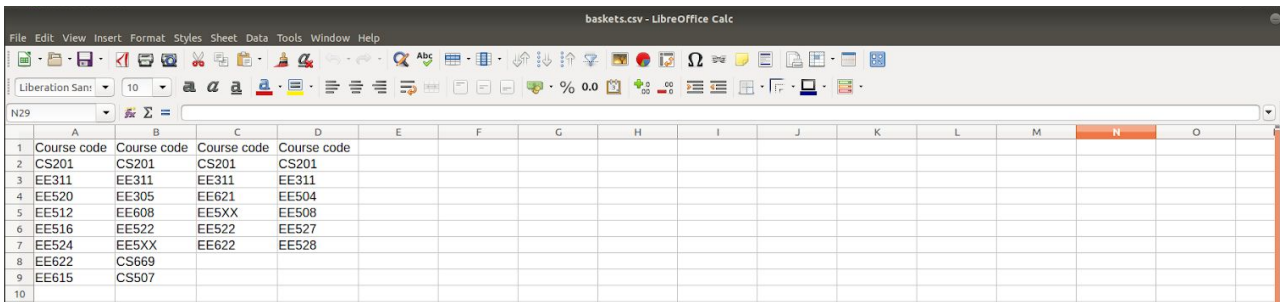
	A	B	C
1	Sr. No	Class	Strength
2	1	A1-3	117
3	2	A1-NKN	154
4	3	A5-1	54
5	4	A5-2	54
6	5	A5-4	63
7	6	A5-5	60

Figure- Part of class.csv

4. *basket.csv*

Basket 1	Basket 2	---

a. The first row must contain headers. The name of headings can be anything
b. Total number of columns = any number
c. Maximum number of rows = 8
d. Each column corresponds to one separate basket



	A	B	C	D
1	Course code	Course code	Course code	Course code
2	CS201	CS201	CS201	CS201
3	EE311	EE311	EE311	EE311
4	EE520	EE305	EE621	EE504
5	EE512	EE608	EE5XX	EE508
6	EE516	EE522	EE522	EE527
7	EE524	EE5XX	EE622	EE528
8	EE622	CS669		
9	EE615	CS507		
10				

Figure- Part of basket.csv

2.2 Database generation

The database has two parts:

1. Main Tables <- *test.py*
   These tables contain the raw data imported from the three .csv files.
2. Runtime Tables <- *allocate.py*
   These tables contain the information about the time-table.

There are 5 fixed tables:

1. Course_Data <- *courses.csv*
2. Professor_Data <- *teacher.csv*
3. Classroom_Data <- *class.csv*
4. Allocated_Subjs
5. Instructor_Slots

There are N number of tables where N is the number of baskets:

1. Basket_n <- *basket.csv*
   n = 1 to N

The details about the database are explained in section 4.

Running these python scripts generates a generic database file (*test.db*) which stores the database in a structured format. It can be opened through terminal:

1. Go to the directory where all the files are stored
2. Right click and choose `Open in Terminal`
3. Command
   `sqlite3`
4. Command
   `.open test.db`
5. Command
   .schema
6. Command
   `select * from Course_Data`
7. Command
   `.exit`

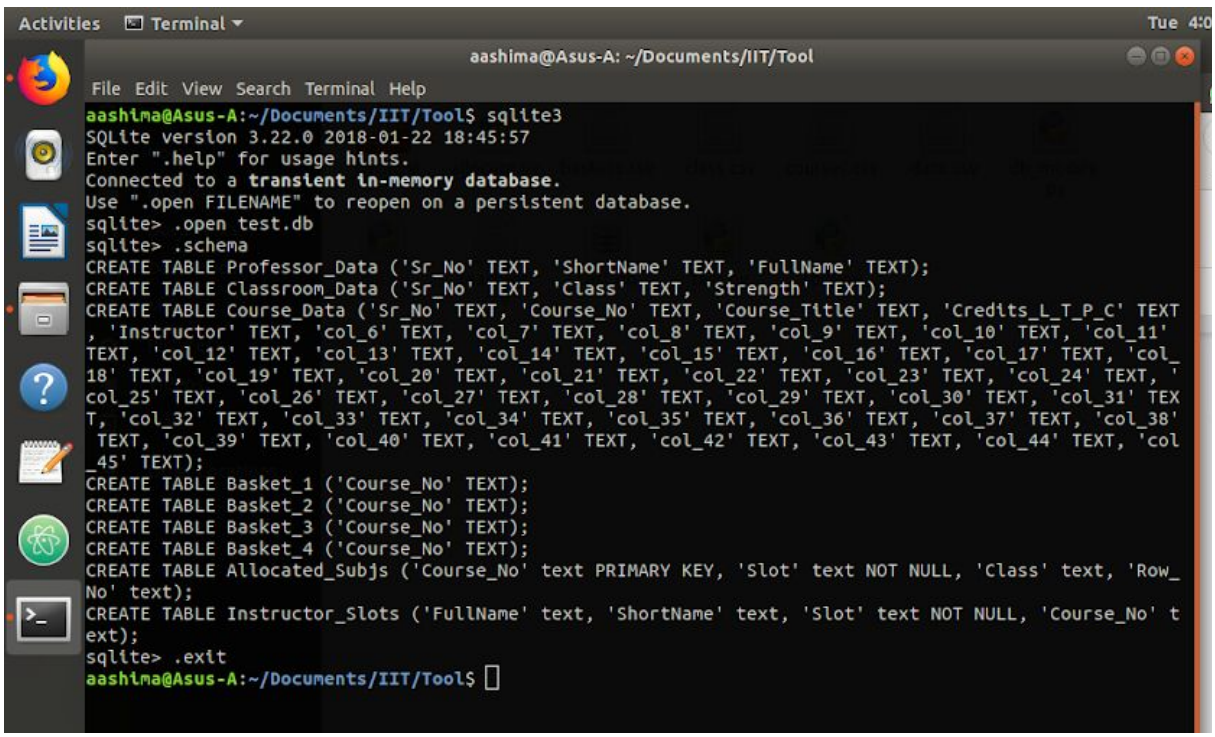


Figure- using sqlite3 on terminal

2.3 User Interface

The python script *gui.py* integrates the user interface and the database. You just have to run *gui.py* and begin your work. It itself executes *test.py* and *allocate.py* before creating the user interface for you. The user interface has been created using python tkinter. You get various features as explained in section 3.

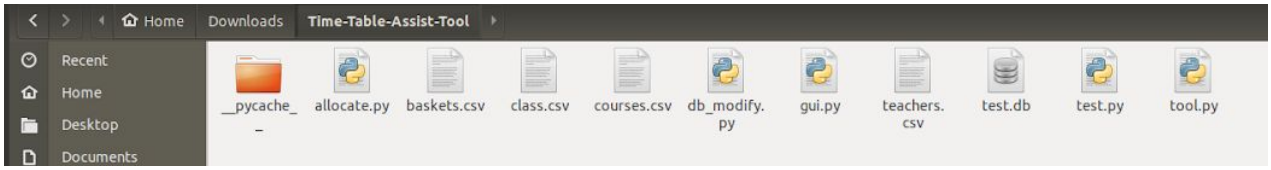
All the changes you make in the timetable are dynamically displayed on the right hand side in the form of a table with 8 columns and 20 rows. All the constraint checking is done using python script integrated with SQLite. If you are interested to understand the working of code, refer to the provided Documentation.

Figure- Opening Screen of GUI

How to use ----------

3.1 Running the Software

3.1.1 Ubuntu



1. Go to the directory where all the files are stored
2. Right click and choose **`Open in Terminal`**
3. Command

   ```
   python3 gui.py
   ```
4. Do your work using the various features as explained in section 3
5. Button **Quit** to exit
6. If the program hangs, press `Ctrl+C` on the terminal. Do not press `Ctrl+Z` as it does not terminate the program, instead puts it to sleep in the background. The program goes to sleep holding access to the database. Subsequently running the program will give an error message.

3.1.2 Windows

1. After downloading the Time-Table-Assist-Tool.zip, extract it in Desktop or anywhere you like.
2. Now go in the newly created folder Time-Table-Assist-Tool and copy the path of the folder (<PATH>)
3. Search for the command prompt by typing so in Windows Search.
4. Once the program is found, open it and type

   ```
   cd <PATH>
   ```
5. Now you are in the software folder.
6. Finally run

   ```
   python gui.py
   ```
7. Do your work using the various features as explained in section 3
8. Button **Quit** to exit.

9. If the program hangs, press `Ctrl+C` on the terminal. Do not press `Ctrl+Z` as it does not terminate the program, instead puts it to sleep in the background. The program goes to sleep holding access to the database. Subsequently running the program will give an error message.

3.2 Features

3.2.1 Time Table

1. **Check Constraints**
   You can allocate time slot and classroom to a new course by filling the input fields. An additional field 'Row' is there to give you complete control over the display of time table.
   This feature checks the validity of inputs and various clashes as explained in section 3.4. All possible clashes are displayed as explanatory warning messages. If there us no possible clash, no message is displayed. It is your responsibility to use these warning messages to correct your input. If you still want to make the allocation, do it as per risk and use the **Add** button.

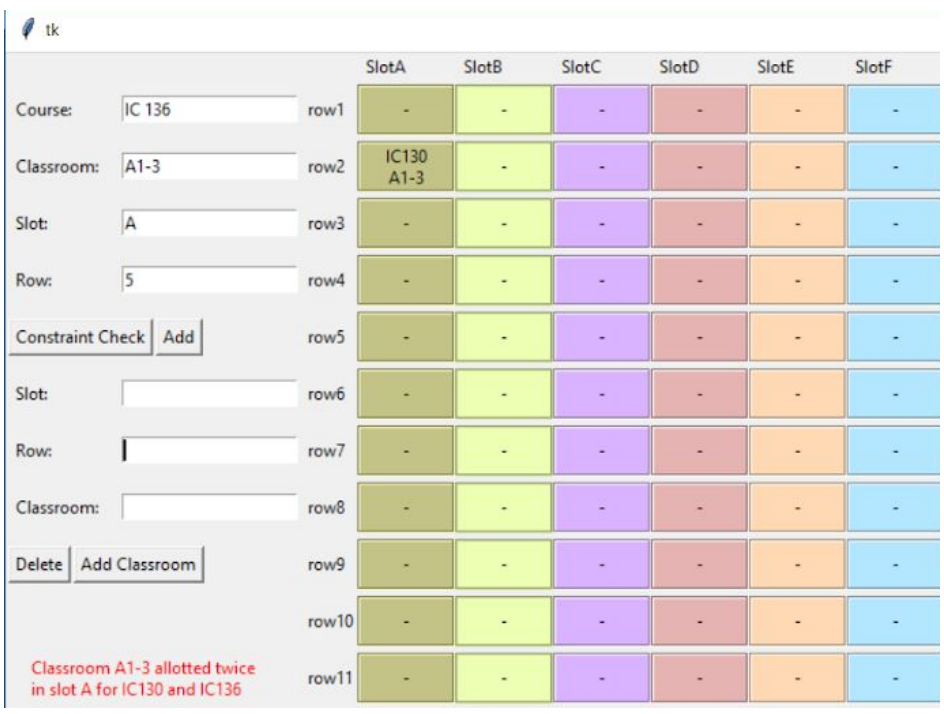


Figure- Feature Check Constraints

2. **Add**
   You can allocate time slot and classroom to a new course by filling the input fields. An additional field ‘Row’ is there to give you complete control over the display of time table.
   This feature checks the validity of inputs and if the course is being added for the first time, makes the desired allocation.
   No clashes are checked for making the allocation and it is your responsibility to use feature **Check Constraints** prior using **Add**.

Figure- Feature Add

3. **Delete**
   You can delete a course from the time table by filling the corresponding slot and row number. You will get an error message when the entries are invalid and an acknowledgement message when the desired course is deleted.

4. **New Time Table**
   Each time you quit the application, your current progress is saved. The same is reloaded when you run the program the next time. This feature erases your progress and you can make a new time table from the beginning.

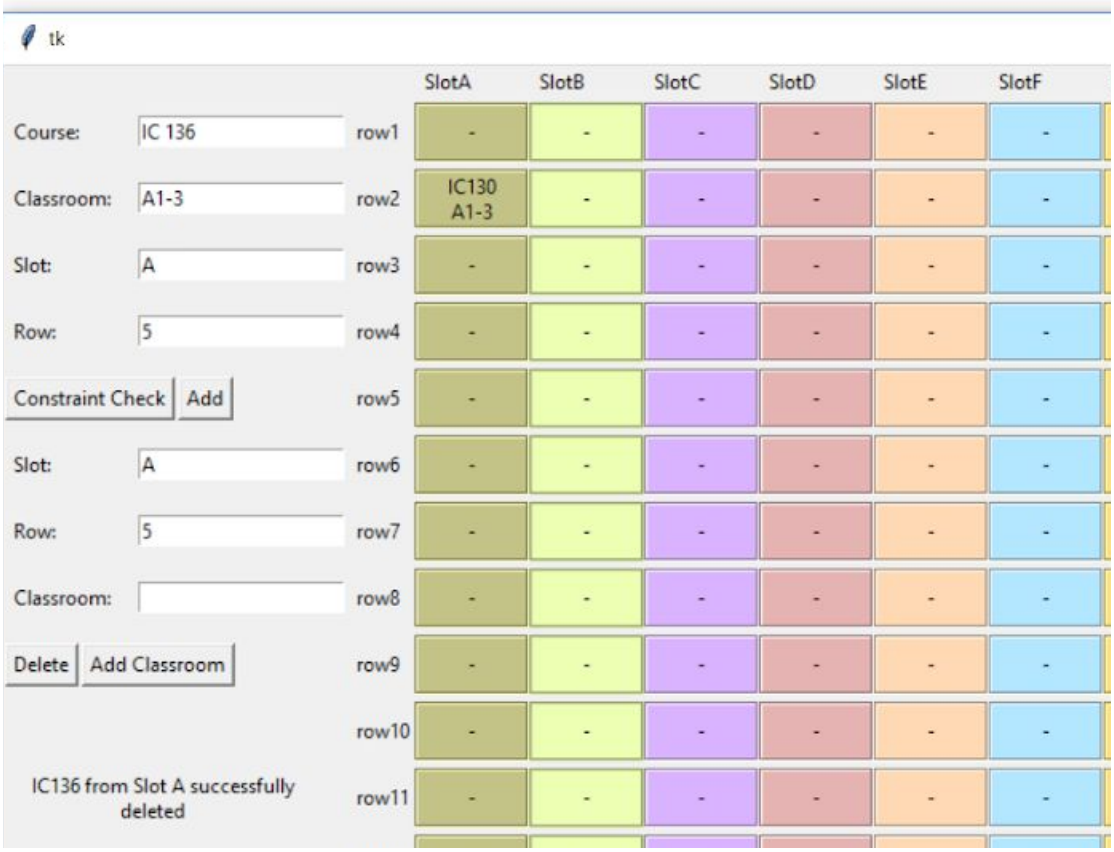


Figure- Feature Delete

Figure- Feature New Time Table

5. **Generate Time Table**
   You can generate a *.csv* file of the time table you have created. The structure is the same as displayed on the user interface. Give the file a name of your choice. The file is saved in the same directory.

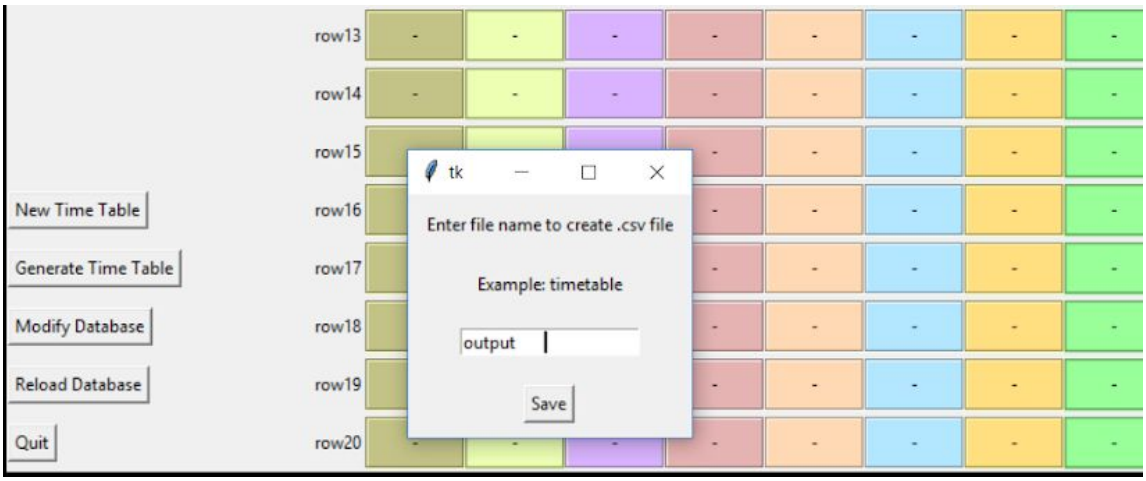


Figure- Feature Generate Time Table

6. **Modify Database**
   If you do not wish to change the *.csv* files but still make changes to the main tables, use this feature. You can modify (add/update/delete) entries from the three main database tables. Pressing the button will open a new window where you can modify the different tables.
   The modifications are independent of the time table but all your progress is erased before making the modifications.

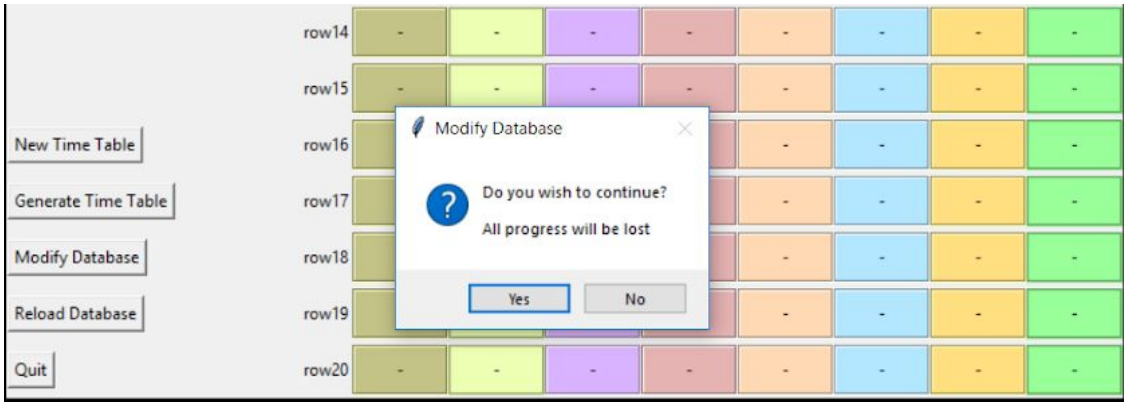


Figure- Feature Modify Database

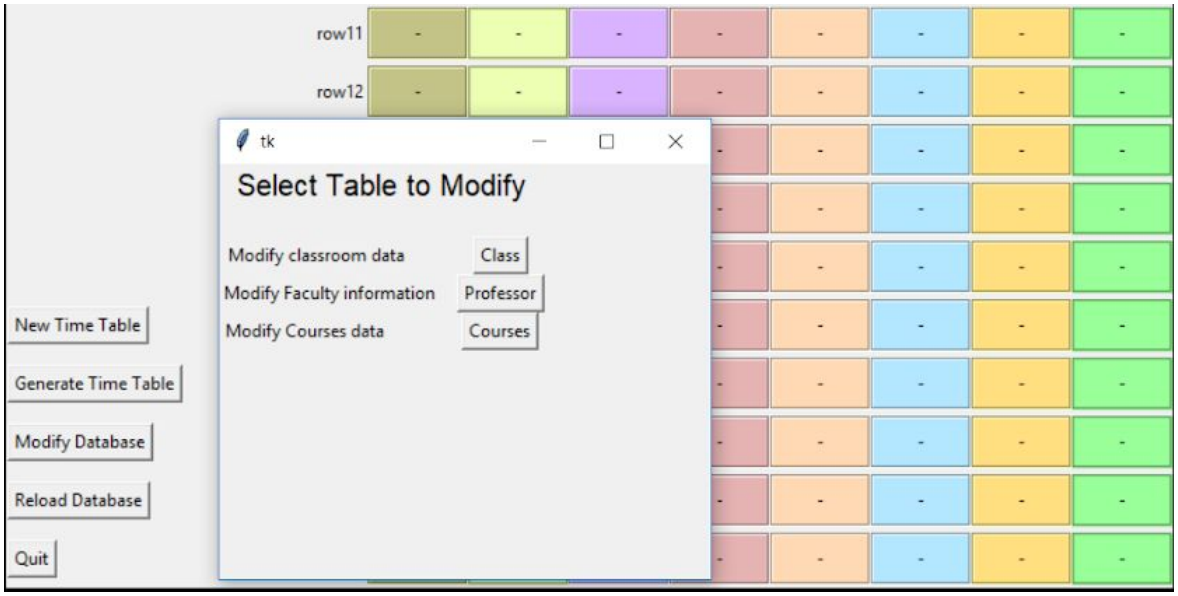


Figure- Feature Modify Database contd.

7. **Reload Database**
   This feature is to reload the database when you change the *.csv* files as explained in section 2.1.1.

Figure- Feature Reload Database

8. **Add Classroom**
   You have the freedom not to choose a classroom and make allocation by leaving classroom input blank. When you later wish to add the classroom use this feature.

9. **Quit**
   Exit the Software and terminate the program.

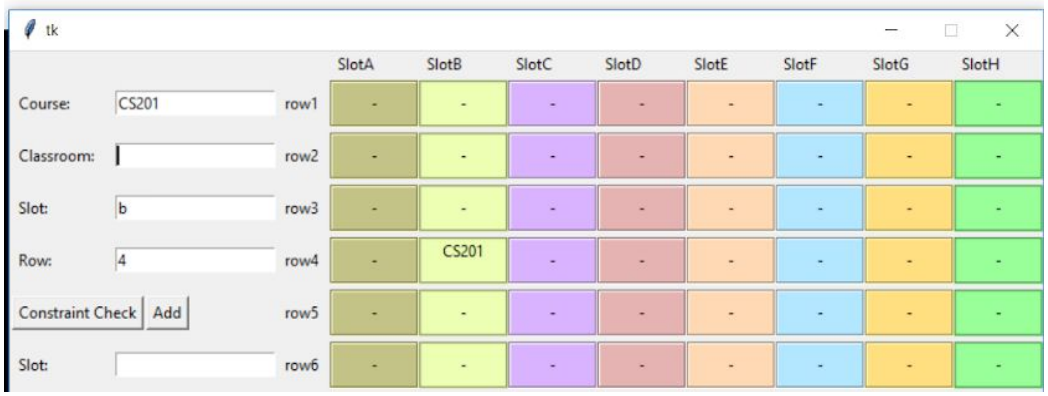


Figure- Feature Add Classroom

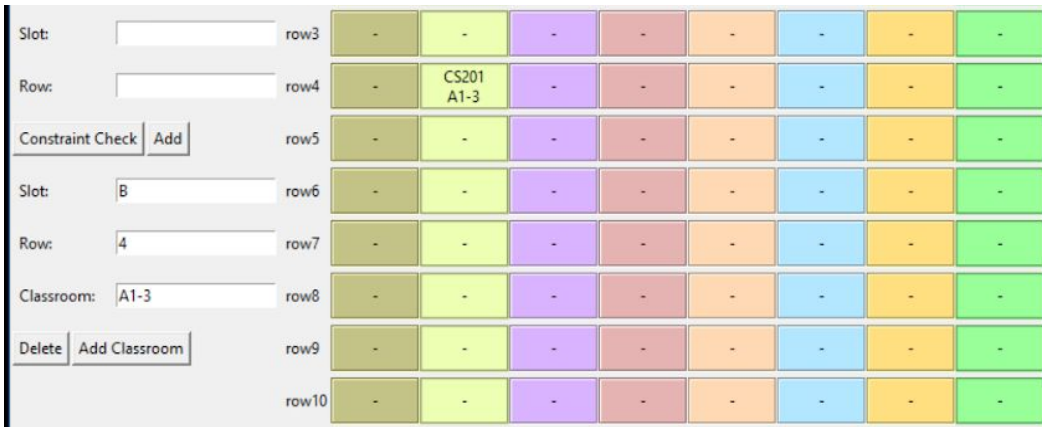


Figure- Feature Add Classroom contd.

3.2.2 Modify Database

Button **Modify Database** open up a new window where you can choose one of the three databases to modify:

1. Course_Data <- **Courses**
2. Professor_Data <- **Professor**
3. Classroom_Data <- **Class**

In each database you can:

1. **Search**
2. **Add** (Not for **Courses**)
3. **Replace**
4. **Delete**

If you wish to see the changes in the database, refer to section 2.2.

3.3 Warnings and Error Messages

3.3.1 Warnings and Errors while Addition

1. Course does not exist
   You have entered wrong course code. The course does not exist in the database. Since the code takes care of removing case sensitivity and extra spaces, you have either entered a wrong course code or the course code is absent from the database. Check the *courses.csv* file and the database by referring to section 2.2.

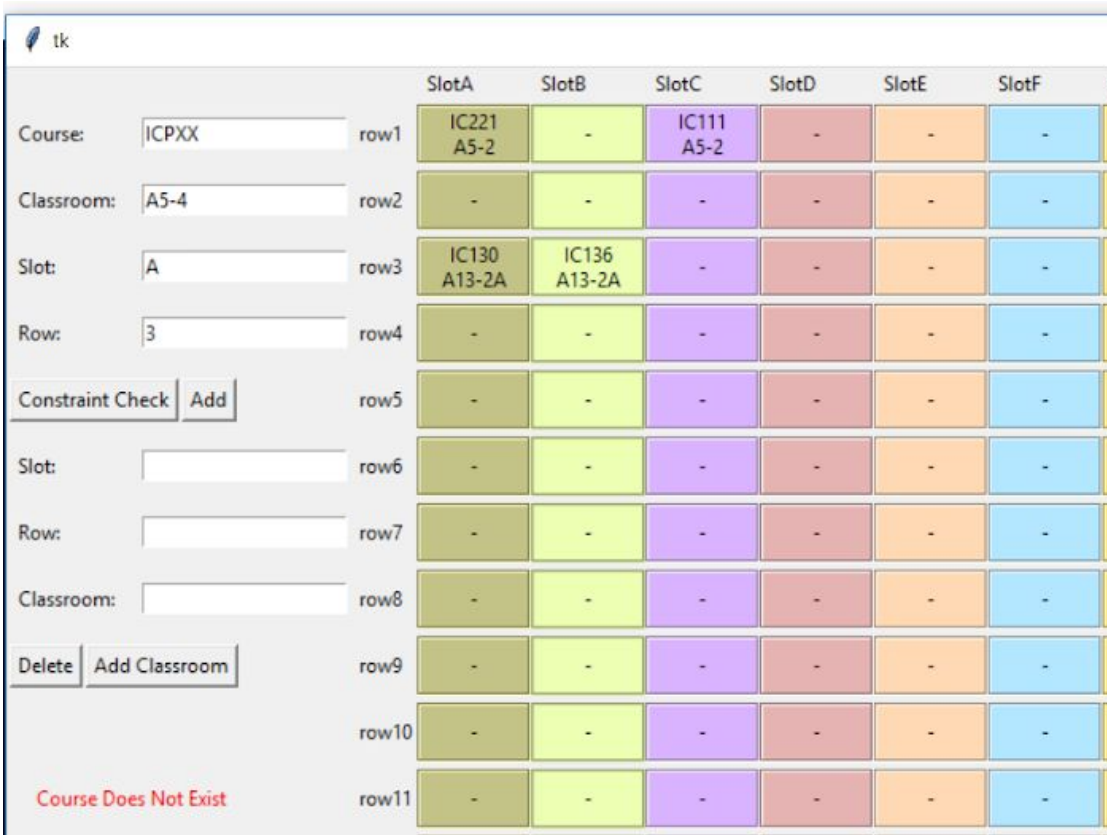


Figure- Error 1

2. Slot does not exist
   You have entered an invalid slot. Enter only letters A-H. Since the code takes care of removing case sensitivity and extra spaces, you have either entered a wrong slot.

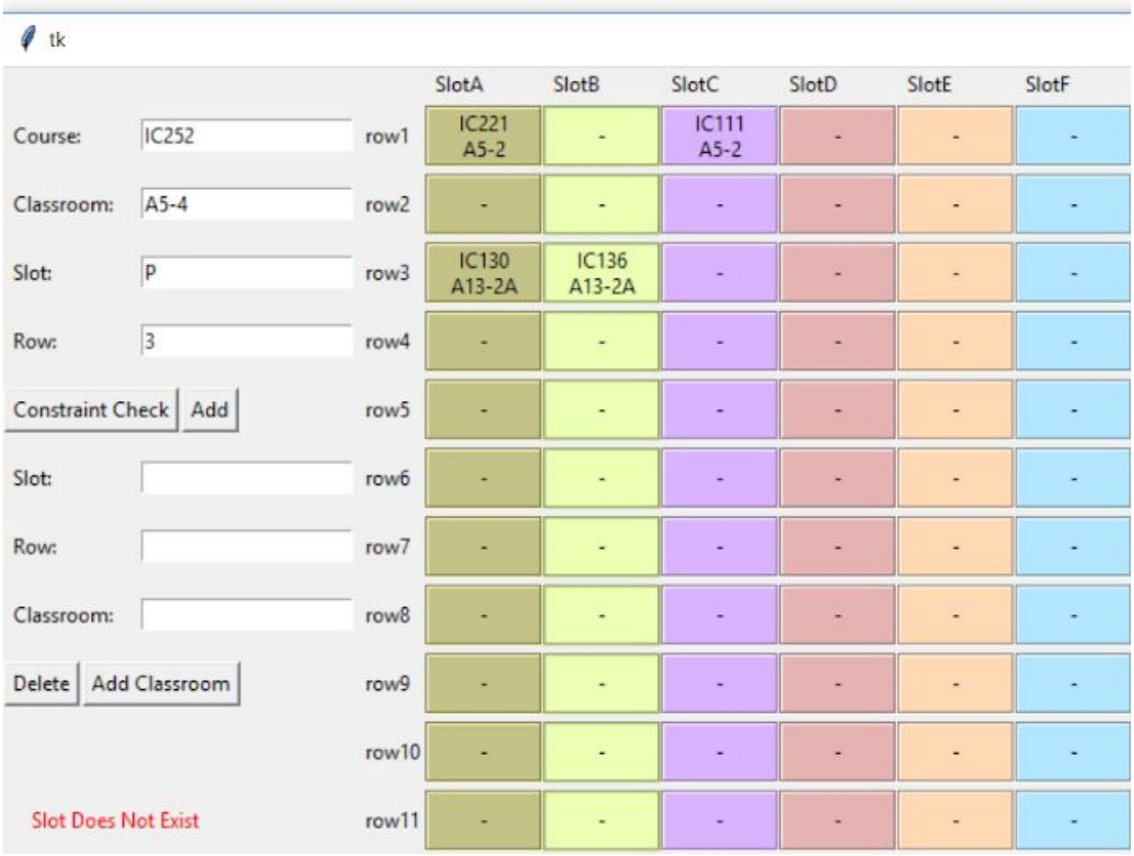


Figure- Error 2

3. Row does not exist
   You have entered an invalid row number. Enter only integers from 1-20.

4. Row already occupied
   You have previously allocated a course to that particular block. Input a new valid row number. If you still wish to make allocation in an occupied block, use **Delete** feature prior

5. Course already added
   You have entered a course which already exists in the timetable. Enter a new course. If you still wish to make the allocation, use **Delete** feature to remove previous occurrence of that course.

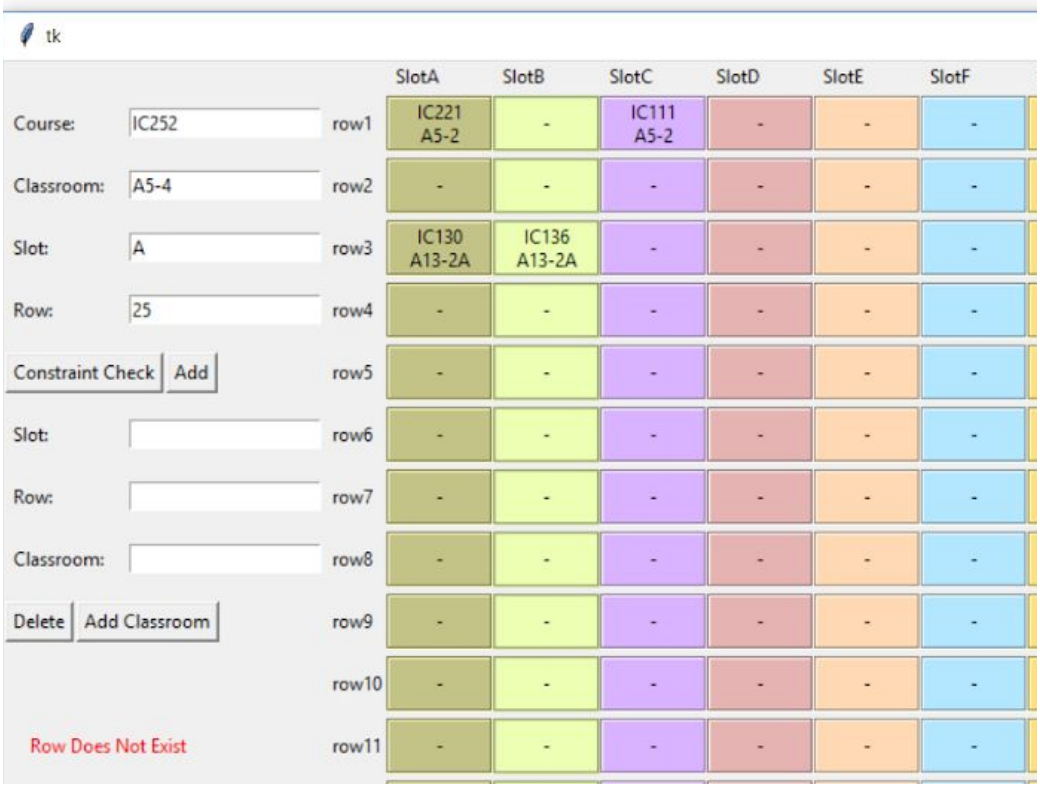


Figure- Error 3

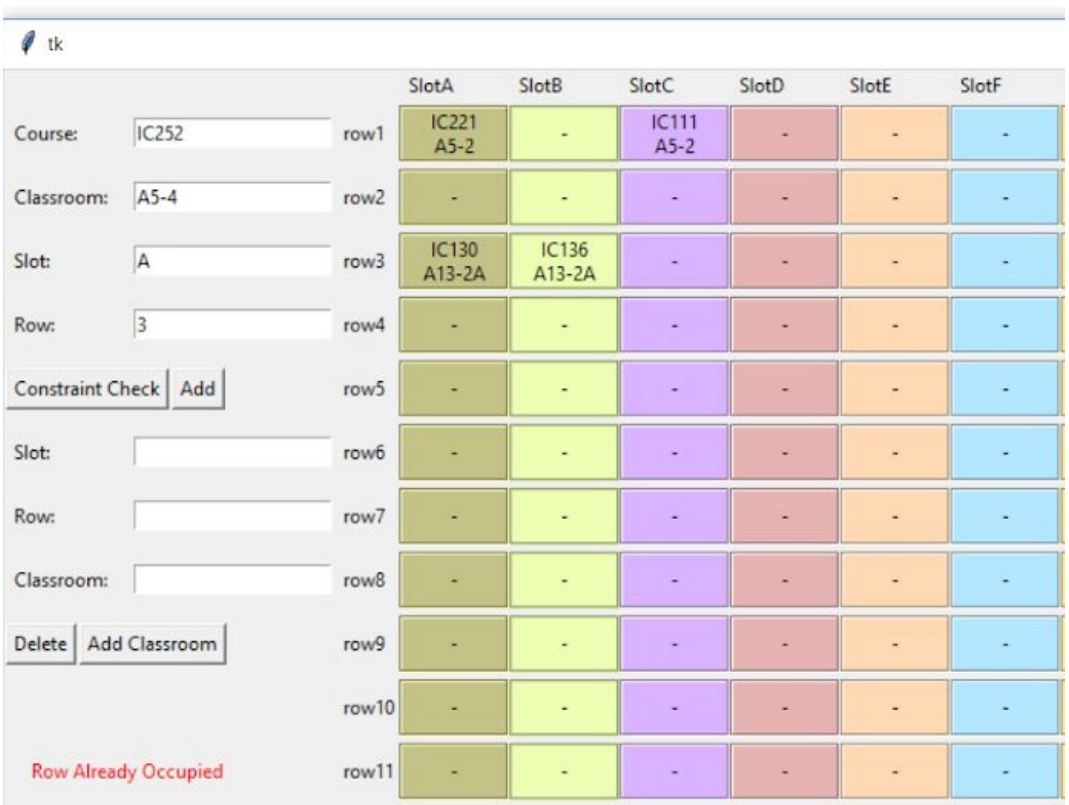


Figure- Error 4

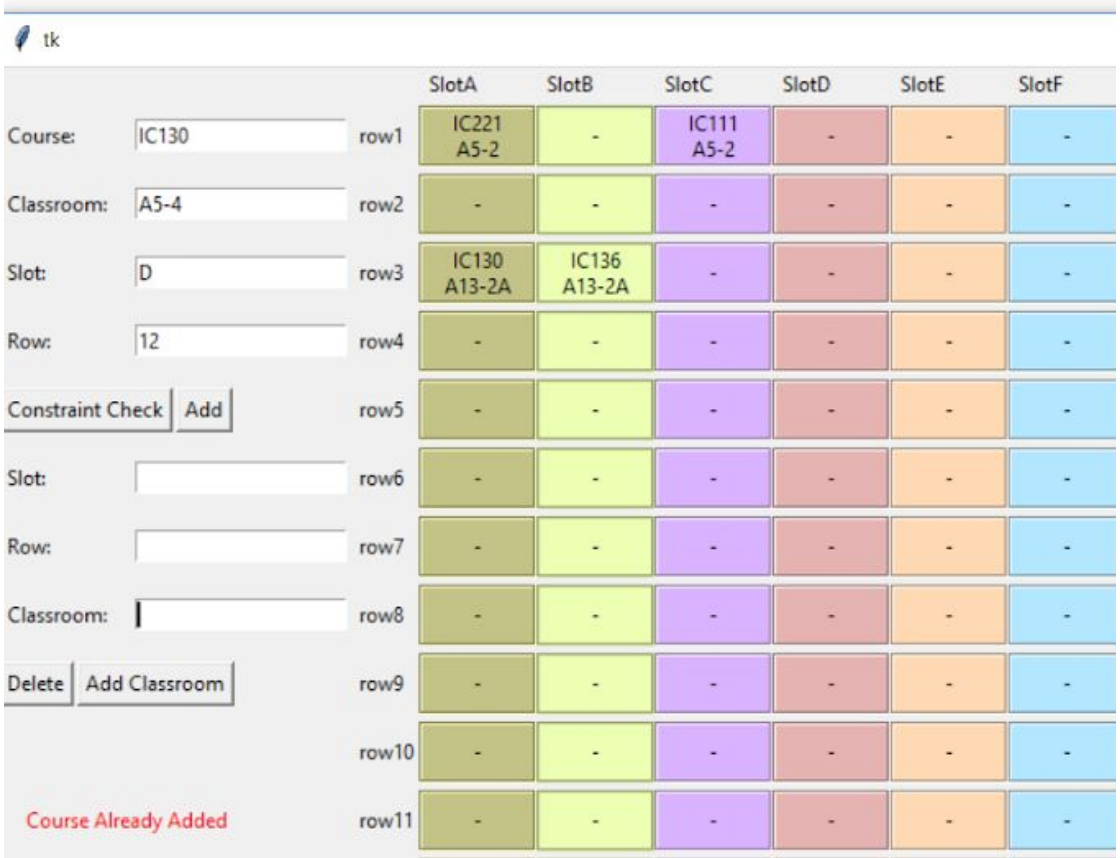


Figure- Error 5

6. ClassroomX allocated twice in slotY to courseA and courseB
   You are allocating two courses A (already added in time-table) and B having common Classroom X in the same slot Y. Make a decision:
   a. Use **Add** feature and make the allocation irrespective of the warning as per risk
   b. Change slot for courseB
   c. Change Classroom for CourseB
   d. **Delete** courseB and make new allocation for both courseA and courseB

7. Faculty X is getting courseB and courseA in slotY
   You are allocating two courses A (already added in time-table) and B having common Faculty X in the same slot Y. Make a decision:
   a. Use **Add** feature and make the allocation irrespective of the warning as per risk
   b. Change slot for courseB
   c. **Delete** courseB and make new allocation for both courseA and courseB

Figure- Warning 7

8. Core Course Clash with courseA

   courseA (already added in time-table) and input course are core for the same batch and cannot be put in the same slot. Make a decision:

   a. Use **Add** feature and make the allocation irrespective of the warning as per risk
   b. Change slot for input course
   c. **Delete** courseA and make new allocation for both courseA and input course

9. Basket Clash with course A

   courseA (already added in time-table) and input course fall in the same basket and cannot be put in the same slot. Make decision:

   a. Use **Add** feature and make the allocation irrespective of the warning as per risk

b. Change slot for input course
c. **Delete** courseA and make new allocation for both courseA and input course

Figure- Warning 6,8 and 9

3.3.2 Errors while Deletion

1. Slot does not exist
2. Row does not exist
3. Slot is empty
4. Row is empty

3.4 Clashes

1. Classroom Clash
2. Instructor Clash
3. Core Course Clash
4. Basket Clash

All these clashes have been explained in reference with warnings in section 3.3.

Database ----------

4.1 Database Design

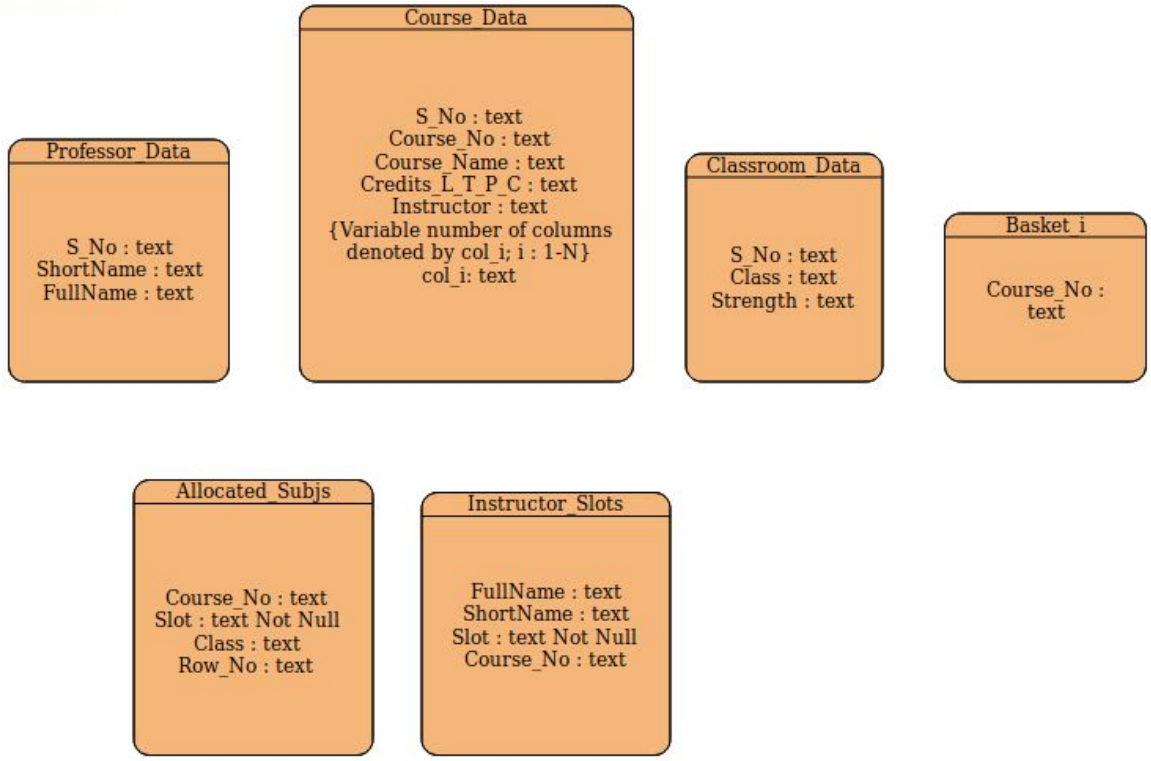


Figure- Database Design

There are 5 fixed tables:

1. Course_Data <- *courses.csv*
2. Professor_Data <- *teacher.csv*
3. Classroom_Data <- *class.csv*

4. Allocated_Subjs
5. Instructor_Slots

There are N number of tables where N is the number of baskets:

1. Basket_n <- *basket.csv*
   n = 1 to N

4.2 How to update database

4.2.1 Modify *.csv* files

Explained in Section 2.1

4.2.2 Modify only Database

Explained in Section 3.2.2

The End