TIME TABLE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY

AMRITHA A **221701007**

MOHAN VAITHYA E **221701037**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND DESIGN

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM
CHENNAI – 602105

ANNA UNIVERSITY: CHENNAI 600625

BONAFIDE CERTIFICATE

Certified that this project report **"TIME TABLE MANAGEMENT SYSTEM"** is the Bonafide work of **"AMRITHA A (221701007), MOHAN VAITHYA E (221701037)**" who carried out the project work under my supervision.

SIGNATURE

Mr.S.Umamaheshwararao MFA.,
Professor and Head,
Computer Science and Design,
Rajalakshmi Engineering College,
Thndalam, Chennai – 602105.

SIGNATURE

Mr.R.Vijayakumar M.Tech.,
Asst. Professor (SS),
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai – 602105.

EXTERNAL EXAMINER

INTERNAL EXAMINER

ACKNOWLEGEMENT

We are highly obliged in taking the opportunity to thank our Chairman **Mr. S. Meganathan**, Chairperson **Dr.Thangam Meganathan** and our Principal **Dr.S.N.Murugesan** for providing all the facilities which are required to carry out this project work.

We are ineffably indebted to our H.O.D **Mr.S.Umameshwararao MFA** for his conscientious guidance and encouragement to make this project a recognizable one.

We are extremely thankful to our faculty **Mr.R.Vijayakumar M.Tech.,** for his valuable guidance and indefatigable support and extend our heartfelt thanks to all the teaching and non-teaching staff of **Computer Science department** who helped us directly or indirectly in the completion of this project successfully.

At last but not least gratitude goes to our friends who helped us compiling the project and finally to god who made all things possible.

Any omission in this brief acknowledgement doesn't mean lack of gratitude.

AMRITHA A 221701007

MOHAN VAITHYA E 221701037

ABSTRACT

This project presents a Time Table Management System designed to streamline the scheduling of classes, exams, and other academic activities. The system addresses common challenges faced by educational institutions, such as scheduling conflicts, resource allocation, and efficient time management. Utilizing a user-friendly interface, the application allows administrators to input course details, faculty availability, and student preferences. It employs algorithms to optimize the timetable, ensuring maximum utilization of resources while minimizing overlaps. The result is a comprehensive timetable that is accessible to students and staff, enhancing overall academic organization. This project not only improves operational efficiency but also fosters a better learning environment by allowing for clear and structured scheduling.

TABLE OF CONTENTS

CHAPTER – 1

1. INTRODUCTION

1.1 INTRODUCTION

The project helps people to know the necessary information, Easily organize tasks, avoid conflicts, and boost productivity in one place. The system presents information in a user-friendly format for convenient planning and organization.

1.2 SCOPE OF THE WORK

The Timetable Management System provides a centralized platform for users to organize and manage their schedules efficiently, whether for classes, meetings, or daily tasks. It enables users to view, edit, and track commitments, helping to prevent conflicts and improve time management.

1.3 PROBLEM STATEMENT

College students often struggle with managing classes, and activities, leading to missed deadlines and stress. The Timetable Management System provides an organized platform to efficiently schedule and track commitments. This system helps students avoid conflicts, improve time management, and stay on top of their responsibilities.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to enable students to create, view, and update their daily, weekly, to minimize scheduling conflicts by offering conflict detection features. To enhance productivity and reduce stress by helping. Students stay organized and manage their time effectively

CHAPTER – 2

2.1 Hardware and software specification

criterion	Description
OS version	Microsoft® Windows® 7/8/10 (32-bit or 64-bit) Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra) GNOME or KDE desktop Linux (64 bit capable of running 32-bit applications)(GNU C Library (glibc) 2.19+)
RAM	3 GB RAM minimum, 8 GB RAM recommended
Disk space	2 GB of available disk space minimum, 4 GB recommended
Python version	Python 3.11
Screen resolution	1280×800 minimum screen resolution

CHAPTER – 3

3 SOFTWARE DESCRIPTION

3.1 PYTHON , SQLITE

The Timetable Management System is a Python-based application designed to help users, particularly students, efficiently organize and manage their academic schedules. The system allows users to input, track, and update their class schedules, assignments, exams, and extracurricular activities. It is built using Python for the backend, ensuring smooth performance and ease of use. The system stores all data, including user schedules, assignments, and event details, in an SQLite database, which offers a lightweight and efficient way to manage the data locally.

3.1.1 Features

- User Authentication:
  Users can create accounts and log in to manage their individual schedules securely.
- Schedule Management:
  Users can add, edit, and delete class schedules, assignments, and events. The system allows for daily, weekly, and monthly views.
- Conflict Detection:
  The system detects and alerts users when there are overlapping classes ,ensuring no conflicts in scheduling.
- Search & Filter Options:
  Users can search and filter schedules by date, course, or category, making it easier to find specific activities.
- Data Storage with SQLite:
  All user data, including schedules and reminders, are securely stored in an SQLite database, which ensures fast data retrieval and easy backup.

CHAPTER - 4

PROJECT DESCRIPTION

The Timetable Management System helps users efficiently organize and manage their schedules by tracking classes . Built with Python and SQLite, it provides features like conflict detection, reminders, and customizable views. The system aims to enhance productivity and reduce stress by offering a user-friendly platform for time management**.**

4.1 MODULE DESCRIPTION

1. Admin
2. Student

4.1.1 ADMIN

The Site allows admin to create , control and access profiles and upload the details

4.1.2 STUDENT

The Site allows students to view the data which have been updated and created by the admin

CHAPTER - 5

IMPLEMENTATION

5.1 Source code

Main.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import os, sys
sys.path.insert(0, './windows')

from windows import timetable_stud
from windows import timetable_fac

import sqlite3

def challenge():
    conn = sqlite3.connect(r'files/timetable.db')

    user = str(combo1.get())
    if user == "Student":
        cursor = conn.execute(f"SELECT PASSW, DEPARTMENT, NAME, ROLL FROM STUDENT WHERE
SID='{id_entry.get()}'")
        cursor = list(cursor)
        if len(cursor) == 0:
            messagebox.showwarning('Bad id', 'No such user found!')
        elif passw_entry.get() != cursor[0][0]:
            messagebox.showerror('Bad pass', 'Incorret Password!')
        else:
            nw = tk.Tk()
            tk.Label(
                nw,
                text=f'{cursor[0][2]}\tDepartment: {cursor[0][1]}\tRoll No.: {cursor[0][3]}',
                font=('times new roman', 12, 'bold'),
            ).pack()
            m.destroy()
            timetable_stud.student_tt_frame(nw, cursor[0][1])
            nw.mainloop()

    elif user == "Faculty":
        cursor = conn.execute(f"SELECT PASSW, INI, NAME, EMAIL FROM FACULTY WHERE
FID='{id_entry.get()}'")
        cursor = list(cursor)
        if len(cursor) == 0:
            messagebox.showwarning('Bad id', 'No such user found!')
        elif passw_entry.get() != cursor[0][0]:
            messagebox.showerror('Bad pass', 'Incorret Password!')
        else:
            nw = tk.Tk()
            tk.Label(
                nw,
                text=f'{cursor[0][2]} ({cursor[0][1]})\tEmail: {cursor[0][3]}',
                font=('times new roman', 12, 'bold'),
            ).pack()
            m.destroy()
            timetable_fac.fac_tt_frame(nw, cursor[0][1])
```

```
        nw.mainloop()

    elif user == "Admin":
        if id_entry.get() == 'admin' and passw_entry.get() == 'admin':
            m.destroy()
            os.system('python windows\\admin_screen.py')
            # sys.exit()
        else:
            messagebox.showerror('Bad Input', 'Incorret Username/Password!')




m = tk.Tk()

m.geometry('400x430')
m.title('REC')

tk.Label(
    m,
    text='TIMETABLE MANAGEMENT SYSTEM',
    font=('times new roman', 15, 'bold'),
    wrap=400
).pack(pady=15)

tk.Label(
    m,
    text='Welcome to Rajalakshmi Engineering College\nLogin',
    font=('times new roman', 14, 'bold')
).pack(pady=10)

tk.Label(
    m,
    text='Username',
    font=('times new roman', 15)
).pack()

id_entry = tk.Entry(
    m,
    font=('times new roman', 12),
    width=21
)
id_entry.pack()

# Label5
tk.Label(
    m,
    text='Password:',
    font=('times new roman', 15)
).pack()

# toggles between show/hide password
def show_passw():
    if passw_entry['show'] == "●":
        passw_entry['show'] = ""
        B1_show['text'] = '●'
        B1_show.update()
    elif passw_entry['show'] == "":
        passw_entry['show'] = "●"
        B1_show['text'] = '○'
        B1_show.update()
    passw_entry.update()

pass_entry_f = tk.Frame()
pass_entry_f.pack()
```

```
# Entry2
passw_entry = tk.Entry(
    pass_entry_f,
    font=('times new roman', 12),
    width=15,
    show="●"
)
passw_entry.pack(side=tk.LEFT)

B1_show = tk.Button(
    pass_entry_f,
    text='○',
    font=('times new roman', 12, 'bold'),
    command=show_passw,
    padx=5
)
B1_show.pack(side=tk.LEFT, padx=15)

combo1 = ttk.Combobox(
    m,
    values=['Student', 'Faculty', 'Admin']
)
combo1.pack(pady=15)
combo1.current(0)

tk.Button(
    m,
    text='Login',
    font=('times new roman', 12, 'bold'),
    padx=30,
    command=challenge
).pack(pady=10)

m.mainloop()
```

Timetable_stud.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3

days = 5
periods = 6
recess_break_aft = 3 # recess after 3rd Period
section = None
butt_grid = []

period_names = list(map(lambda x: 'Period ' + str(x), range(1, 6+1)))
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thrusday', 'Friday']


def select_sec():
    global section
    section = str(combo1.get())
    print(section)
    update_table(section)


def update_table(sec):
    for i in range(days):
        for j in range(periods):
            cursor = conn.execute(f"SELECT SUBCODE, FINI FROM SCHEDULE\
                WHERE DAYID={i} AND PERIODID={j} AND SECTION='{sec}'")
```

```
            cursor = list(cursor)
            print(cursor)

            butt_grid[i][j]['bg'] = 'white'
            if len(cursor) != 0:
                subcode = cursor[0][0]
                cur1 = conn.execute(F"SELECT SUBTYPE FROM SUBJECTS WHERE SUBCODE='{subcode}'")
                cur1 = list(cur1)
                subtype = cur1[0][0]
                butt_grid[i][j]['fg'] = 'white'
                if subtype == 'T':
                    butt_grid[i][j]['bg'] = 'green'
                elif subtype == 'P':
                    butt_grid[i][j]['bg'] = 'blue'

                butt_grid[i][j]['text'] = str(cursor[0][0]) + '\n' + str(cursor[0][1])
                butt_grid[i][j].update()
                print(i, j, cursor[0][0])
            else:
                butt_grid[i][j]['fg'] = 'black'
                butt_grid[i][j]['text'] = "No Class"
                butt_grid[i][j].update()


def process_button(d, p, sec):
    details = tk.Tk()
    cursor = conn.execute(f"SELECT SUBCODE, FINI FROM SCHEDULE\
                WHERE ID='{section+str((d*periods)+p)}'")
    cursor = list(cursor)
    if len(cursor) != 0:
        subcode = str(cursor[0][0])
        fini =  str(cursor[0][1])

        cur1 = conn.execute(f"SELECT SUBNAME, SUBTYPE FROM SUBJECTS\
            WHERE SUBCODE='{subcode}'")
        cur1 = list(cur1)
        subname = str(cur1[0][0])
        subtype = str(cur1[0][1])

        cur2 = conn.execute(f"SELECT NAME, EMAIL FROM FACULTY\
            WHERE INI='{fini}'")
        cur2 = list(cur2)
        fname = str(cur2[0][0])
        femail = str(cur2[0][1])

        if subtype == 'T':
            subtype = 'Theory'
        elif subtype == 'P':
            subtype = 'Practical'

    else:
        subcode = fini = subname = subtype = fname = femail = 'None'

    print(subcode, fini, subname, subtype, fname, femail)
    tk.Label(details, text='Class Details', font=('times new roman', 15, 'bold')).pack(pady=15)
    tk.Label(details, text='Day: '+day_names[d], font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Period: '+str(p+1), font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X, padx=20)
    tk.Label(details, text='Subject Code: '+subcode, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Subect Name: '+subname, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Subject Type: '+subtype, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Faculty Initials: '+fini, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
```

```
padx=20)
    tk.Label(details, text='Faculty Name: '+fname, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Faculty Email: '+femail, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)

    tk.Button(
        details,
        text="OK",
        font=('times new roman'),
        width=10,
        command=details.destroy
    ).pack(pady=10)

    details.mainloop()


def student_tt_frame(tt, sec):
    title_lab = tk.Label(
        tt,
        text='T  I  M  E  T  A  B  L  E',
        font=('times new roman', 20, 'bold'),
        pady=5
    )
    title_lab.pack()

    legend_f = tk.Frame(tt)
    legend_f.pack(pady=15)
    tk.Label(
        legend_f,
        text='Legend: ',
        font=('times new roman', 10, 'bold')
    ).pack(side=tk.LEFT)

    tk.Label(
        legend_f,
        text='Theory Classes',
        bg='green',
        fg='white',
        relief='raised',
        font=('times new roman', 10, 'bold'),
        height=2
    ).pack(side=tk.LEFT, padx=10)

    tk.Label(
        legend_f,
        text='Practical Classes',
        bg='blue',
        fg='white',
        relief='raised',
        font=('times new roman', 10, 'bold'),
        height=2
    ).pack(side=tk.LEFT, padx=10)

    global butt_grid
    global section
    section = sec

    table = tk.Frame(tt)
    table.pack()

    first_half = tk.Frame(table)
    first_half.pack(side='left')

    recess_frame = tk.Frame(table)
```

```
recess_frame.pack(side='left')

second_half = tk.Frame(table)
second_half.pack(side='left')

recess = tk.Label(
    recess_frame,
    text='R\n\nE\n\nC\n\nE\n\nS\n\nS',
    font=('times new roman', 18, 'bold'),
    width=3,
    relief='sunken'
)
recess.pack()

for i in range(days):
    b = tk.Label(
        first_half,
        text=day_names[i],
        font=('times new roman', 12, 'bold'),
        width=9,
        height=2,
        bd=5,
        relief='raised'
    )
    b.grid(row=i+1, column=0)

for i in range(periods):
    if i < recess_break_aft:
        b = tk.Label(first_half)
        b.grid(row=0, column=i+1)
    else:
        b = tk.Label(second_half)
        b.grid(row=0, column=i)

    b.config(
        text=period_names[i],
        font=('times new roman', 12, 'bold'),
        width=9,
        height=1,
        bd=5,
        relief='raised'
    )

for i in range(days):
    b = []
    for j in range(periods):
        if j < recess_break_aft:
            bb = tk.Button(first_half)
            bb.grid(row=i+1, column=j+1)
        else:
            bb = tk.Button(second_half)
            bb.grid(row=i+1, column=j)

        bb.config(
            text='Hello World!',
            font=('times new roman', 10),
            width=13,
            height=3,
            bd=5,
            relief='raised',
            wraplength=80,
            justify='center',
            command=lambda x=i, y=j, z=sec: process_button(x, y, z)
        )
        b.append(bb)
```

```
        butt_grid.append(b)
        # print(b)
        b = []

    print(butt_grid[0][1], butt_grid[1][1])
    update_table(sec)


conn = sqlite3.connect(r'files/timetable.db')
if __name__ == "__main__":

    # connecting database

    tt = tk.Tk()
    tt.title('Student Timetable')

    student_tt_frame(tt, section)

    sec_select_f = tk.Frame(tt, pady=15)
    sec_select_f.pack()

    tk.Label(
        sec_select_f,
        text='Select Department:  ',
        font=('times new roman', 12, 'bold')
    ).pack(side=tk.LEFT)

    cursor = conn.execute("SELECT DISTINCT SECTION FROM STUDENT")
    sec_li = [row[0] for row in cursor]
    # sec_li.insert(0, 'NULL')
    print(sec_li)
    combo1 = ttk.Combobox(
        sec_select_f,
        values=sec_li,
    )
    combo1.pack(side=tk.LEFT)
    combo1.current(0)

    b = tk.Button(
        sec_select_f,
        text="OK",
        font=('times new roman', 12, 'bold'),
        padx=10,
        command=select_sec
    )
    b.pack(side=tk.LEFT, padx=10)
    b.invoke()

    tt.mainloop()
```

Timetable_fac.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3

days = 5
periods = 6
recess_break_aft = 3 # recess after 3rd Period
fini = None
butt_grid = []
```

```
period_names = list(map(lambda x: 'Period ' + str(x), range(1, 6+1)))
day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thrusday', 'Friday']


def select_fac():
    global fini
    fini = str(combo1.get())
    print(fini)
    update_table(fini)


def update_table(fini):
    for i in range(days):
        for j in range(periods):
            cursor = conn.execute(f"SELECT SECTION, SUBCODE FROM SCHEDULE\
                WHERE DAYID={i} AND PERIODID={j} AND FINI='{fini}'")
            cursor = list(cursor)
            print(cursor)

            butt_grid[i][j]['bg'] = 'white'
            if len(cursor) != 0:
                subcode = cursor[0][1]
                cur1 = conn.execute(F"SELECT SUBTYPE FROM SUBJECTS WHERE SUBCODE='{subcode}'")
                cur1 = list(cur1)
                subtype = cur1[0][0]
                butt_grid[i][j]['fg'] = 'white'
                if subtype == 'T':
                    butt_grid[i][j]['bg'] = 'green'
                elif subtype == 'P':
                    butt_grid[i][j]['bg'] = 'blue'

                sec_li = [x[0] for x in cursor]
                t = ', '.join(sec_li)
                butt_grid[i][j]['text'] = "Sections: " + t
                print(i, j, cursor[0][0])
            else:
                butt_grid[i][j]['fg'] = 'black'
                butt_grid[i][j]['text'] = "No Class"
                butt_grid[i][j].update()


def process_button(d, p):
    print(d, p, fini)
    details = tk.Tk()
    cursor = conn.execute(f"SELECT SECTION, SUBCODE FROM SCHEDULE\
                WHERE DAYID={d} AND PERIODID={p} AND FINI='{fini}'")
    cursor = list(cursor)
    print("section", cursor)
    if len(cursor) != 0:
        sec_li = [x[0] for x in cursor]
        t = ', '.join(sec_li)
        subcode = cursor[0][1]
        cur1 = conn.execute(f"SELECT SUBNAME, SUBTYPE FROM SUBJECTS\
            WHERE SUBCODE='{subcode}'")
        cur1 = list(cur1)
        subname = str(cur1[0][0])
        subtype = str(cur1[0][1])

        if subtype == 'T':
            subtype = 'Theory'
        elif subtype == 'P':
            subtype = 'Practical'

    #     print(subcode, fini, subname, subtype, fname, femail)
    else:
```

```
        sec_li = subcode = subname = subtype = t = 'None'

    tk.Label(details, text='Class Details', font=('times new roman', 15, 'bold')).pack(pady=15)
    tk.Label(details, text='Day: '+day_names[d], font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Period: '+str(p+1), font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X, padx=20)
    tk.Label(details, text='Subject Code: '+subcode, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Subect Name: '+subname, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Subject Type: '+subtype, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Faculty Initials: '+fini, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X,
padx=20)
    tk.Label(details, text='Sections: '+t, font=('times new roman'), anchor="w").pack(expand=1, fill=tk.X, padx=20)

    tk.Button(
        details,
        text="OK",
        font=('times new roman'),
        width=10,
        command=details.destroy
    ).pack(pady=10)

    details.mainloop()


def fac_tt_frame(tt, f):
    title_lab = tk.Label(
        tt,
        text='T  I  M  E  T  A  B  L  E',
        font=('times new roman', 20, 'bold'),
        pady=5
    )
    title_lab.pack()

    legend_f = tk.Frame(tt)
    legend_f.pack(pady=15)
    tk.Label(
        legend_f,
        text='Legend: ',
        font=('times new roman', 10, 'italic')
    ).pack(side=tk.LEFT)

    tk.Label(
        legend_f,
        text='Theory Classes',
        bg='green',
        fg='white',
        relief='raised',
        font=('times new roman', 10, 'italic'),
        height=2
    ).pack(side=tk.LEFT, padx=10)

    tk.Label(
        legend_f,
        text='Practical Classes',
        bg='blue',
        fg='white',
        relief='raised',
        font=('times new roman', 10, 'italic'),
        height=2
    ).pack(side=tk.LEFT, padx=10)

    global butt_grid
```

```
    global fini
    fini = f

    table = tk.Frame(tt)
    table.pack()

    first_half = tk.Frame(table)
    first_half.pack(side='left')

    recess_frame = tk.Frame(table)
    recess_frame.pack(side='left')

    second_half = tk.Frame(table)
    second_half.pack(side='left')

    recess = tk.Label(
        recess_frame,
        text='R\n\nE\n\nC\n\nE\n\nS\n\nS',
        font=('times new roman', 18, 'italic'),
        width=3,
        relief='sunken'
    )
    recess.pack()

    for i in range(days):
        b = tk.Label(
            first_half,
            text=day_names[i],
            font=('times new roman', 12, 'bold'),
            width=9,
            height=2,
            bd=5,
            relief='raised'
        )
        b.grid(row=i+1, column=0)

    for i in range(periods):
        if i < recess_break_aft:
            b = tk.Label(first_half)
            b.grid(row=0, column=i+1)
        else:
            b = tk.Label(second_half)
            b.grid(row=0, column=i)

        b.config(
            text=period_names[i],
            font=('times new roman', 12, 'bold'),
            width=9,
            height=1,
            bd=5,
            relief='raised'
        )

    for i in range(days):
        b = []
        for j in range(periods):
            if j < recess_break_aft:
                bb = tk.Button(first_half)
                bb.grid(row=i+1, column=j+1)
            else:
                bb = tk.Button(second_half)
                bb.grid(row=i+1, column=j)

            bb.config(
                text='Hello World!',
```

```
            font=('times new roman', 10),
            width=13,
            height=3,
            bd=5,
            relief='raised',
            wraplength=80,
            justify='center',
            command=lambda x=i, y=j: process_button(x, y)
        )
        b.append(bb)

    butt_grid.append(b)
    # print(b)
    b = []

print(butt_grid[0][1], butt_grid[1][1])
update_table(fini)


conn = sqlite3.connect(r'files/timetable.db')
if __name__ == "__main__":

    # connecting database

    tt = tk.Tk()
    tt.title('Faculty Timetable')

    fac_tt_frame(tt, fini)

    fac_select_f = tk.Frame(tt, pady=15)
    fac_select_f.pack()

    tk.Label(
        fac_select_f,
        text='Select Faculty:  ',
        font=('times new roman', 12, 'bold')
    ).pack(side=tk.LEFT)

    cursor = conn.execute("SELECT DISTINCT INI FROM FACULTY")
    fac_li = [row[0] for row in cursor]
    print(fac_li)
    combo1 = ttk.Combobox(
        fac_select_f,
        values=fac_li,
    )
    combo1.pack(side=tk.LEFT)
    combo1.current(0)

    b = tk.Button(
        fac_select_f,
        text="OK",
        font=('times new roman', 12, 'bold'),
        padx=10,
        command=select_fac
    )
    b.pack(side=tk.LEFT, padx=10)
    b.invoke()

    tt.mainloop()
```

Subjects.py

```
import sqlite3
import tkinter as tk
```

```
from tkinter import ttk
from tkinter import messagebox
import sys

# inputs in this window
subcode = subname = subtype = None

'''
    LIST OF FUNCTIONS USED FOR VARIOUS FUNCTIONS THROUGH TKinter INTERFACE
        * create_treeview()
        * update_treeview()
        * parse_data()
        * update_data()
        * remove_data()
'''

# create treeview (call this function once)
def create_treeview():
    tree['columns'] = ('one', 'two', 'three')
    tree.column("#0", width=0, stretch=tk.NO)
    tree.column("one", width=70, stretch=tk.NO)
    tree.column("two", width=300, stretch=tk.NO)
    tree.column("three", width=60, stretch=tk.NO)
    tree.heading('#0', text="")
    tree.heading('one', text="Code")
    tree.heading('two', text="Name")
    tree.heading('three', text="Type")

# update treeview (call this function after each update)
def update_treeview():
    for row in tree.get_children():
        tree.delete(row)
    cursor = conn.execute("SELECT * FROM SUBJECTS")
    for row in cursor:
        # print(row[0], row[1], row[2])
        if row[2] == 'T':
            t = 'Theory'
        elif row[2] == 'P':
            t = 'Practical'
        tree.insert(
            "",
            0,
            values=(row[0],row[1],t)
        )
    tree.place(x=500, y=100)

# Parse and store data into database and treeview upon clcicking of the add button
def parse_data():
    subcode = str(subcode_entry.get())
    subname = str(subname_entry.get("1.0", tk.END)).upper().rstrip()
    subtype = str(radio_var.get()).upper()

    if subcode==="":
        subcode = None
    if subname==="":
        subname = None

    if subcode is None or subname is None:
        messagebox.showerror("Bad Input", "Please fill up Subject Code and/or Subject Name!")
        subcode_entry.delete(0, tk.END)
        subname_entry.delete("1.0", tk.END)
        return

    conn.execute(f'REPLACE INTO SUBJECTS (SUBCODE, SUBNAME, SUBTYPE)\
        VALUES ('{subcode}','{subname}','{subtype}')")
```

```
    conn.commit()
    update_treeview()

    subcode_entry.delete(0, tk.END)
    subname_entry.delete("1.0", tk.END)

# update a row in the database
def update_data():
    subcode_entry.delete(0, tk.END)
    subname_entry.delete("1.0", tk.END)
    try:
        # print(tree.selection())
        if len(tree.selection()) > 1:
            messagebox.showerror("Bad Select", "Select one subject at a time to update!")
            return

        row = tree.item(tree.selection()[0])['values']
        subcode_entry.insert(0, row[0])
        subname_entry.insert("1.0", row[1])
        if row[2][0] == "T":
            R1.select()
        elif row[2][0] == "P":
            R2.select()

        conn.execute(f"DELETE FROM SUBJECTS WHERE SUBCODE = '{row[0]}'")
        conn.commit()
        update_treeview()

    except IndexError:
        messagebox.showerror("Bad Select", "Please select a subject from the list first!")
        return

# remove selected data from databse and treeview
def remove_data():
    if len(tree.selection()) < 1:
        messagebox.showerror("Bad Select", "Please select a subject from the list first!")
        return
    for i in tree.selection():
        # print(tree.item(i)['values'][0])
        conn.execute(f"DELETE FROM SUBJECTS WHERE SUBCODE = '{tree.item(i)['values'][0]}'")
        conn.commit()
        tree.delete(i)
        update_treeview()


# main
if __name__ == "__main__":

    '''
        DATABASE CONNECTIONS AND SETUP
    '''

    # connecting database
    conn = sqlite3.connect(r'files/timetable.db')

    # creating Tabe in the database
    conn.execute('CREATE TABLE IF NOT EXISTS SUBJECTS\
    (SUBCODE CHAR(10) NOT NULL PRIMARY KEY,\
    SUBNAME CHAR(50) NOT NULL,\
    SUBTYPE CHAR(1) NOT NULL)')

    '''
        TKinter WINDOW SETUP WITH WIDGETS
            * Label(1-6)
            * Entry(1)
```

```
        * Text(1)
        * Radiobutton(1-2)
        * Treeview(1)
        * Button(1-2)
'''

# TKinter Window
subtk = tk.Tk()
subtk.geometry('1000x450')
subtk.title('Add/Update Subjects')

# Label1
tk.Label(
    subtk,
    text='List of Subjects',
    font=('times new roman', 20, 'bold')
).place(x=600, y=50)

# Label2
tk.Label(
    subtk,
    text='Add/Update Subjects',
    font=('times new roman', 20, 'bold')
).place(x=100, y=50)

# Label3
tk.Label(
    subtk,
    text='Add information in the following prompt!',
    font=('times new roman', 10, 'italic')
).place(x=100, y=85)

# Label4
tk.Label(
    subtk,
    text='Subject code:',
    font=('times new roman', 15)
).place(x=100, y=150)

# Entry1
subcode_entry = tk.Entry(
    subtk,
    font=('times new roman', 15),
    width=11
)
subcode_entry.place(x=270, y=150)

# Label5
tk.Label(
    subtk,
    text='Subject Name:',
    font=('times new roman', 15)
).place(x=100, y=200)

# Text
subname_entry = tk.Text(
    subtk,
    font=('times new roman', 10),
    width=17,
    height=3,
    wrap=tk.WORD
)
subname_entry.place(x=270, y=200)

# Label6
```

```
tk.Label(
    subtk,
    text='Subject Type:',
    font=('times new roman', 15)
).place(x=100, y=270)

# RadioButton variable to store RadioButton Status
radio_var = tk.StringVar()

# RadioButton1
R1 = tk.Radiobutton(
    subtk,
    text='Theory',
    font=('times new roman', 12),
    variable=radio_var,
    value="T"
)
R1.place(x=270, y=270)
R1.select()

# RadioButton2
R2 = tk.Radiobutton(
    subtk,
    text='Practical',
    font=('times new roman', 12),
    variable=radio_var,
    value="P"
)
R2.place(x=270, y=300)
R2.select()

# Button1
B1 = tk.Button(
    subtk,
    text='Add Subject',
    font=('times new roman', 12),
    command=parse_data
)
B1.place(x=150,y=350)

# Button2
B2 = tk.Button(
    subtk,
    text='Update Subject',
    font=('times new roman', 12),
    command=update_data
)
B2.place(x=410,y=350)

# Treeview1
tree = ttk.Treeview(subtk)
create_treeview()
update_treeview()

# Button3
B3 = tk.Button(
    subtk,
    text='Delete Subject(s)',
    font=('times new roman', 12),
    command=remove_data
)
B3.place(x=650,y=350)

# looping Tkiniter window
subtk.mainloop()
```

```
    conn.close() # close database ad=fter all operations
```

Student.py

```
import sqlite3
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sys

fid = passw = conf_passw = name = roll = section = None

'''
    LIST OF FUNCTIONS USED FOR VARIOUS FUNCTIONS THROUGH TKinter INTERFACE
        * create_treeview()
        * update_treeview()
        * parse_data()
        * update data()
        * remove_data()
        * show_passw()
'''

# create treeview (call this function once)
def create_treeview():
    tree['columns'] = list(map(lambda x: '#' + str(x), range(1, 5)))
    tree.column("#0", width=0, stretch=tk.NO)
    tree.column("#1", width=70, stretch=tk.NO)
    tree.column("#2", width=200, stretch=tk.NO)
    tree.column("#3", width=80, stretch=tk.NO)
    tree.column("#4", width=80, stretch=tk.NO)
    tree.heading('#0', text="")
    tree.heading('#1', text="sid")
    tree.heading('#2', text="Name")
    tree.heading('#3', text="Roll")
    tree.heading('#4', text="Department")
    tree['height'] = 12

# update treeview (call this function after each update)
def update_treeview():
    for row in tree.get_children():
        tree.delete(row)
    cursor = conn.execute("SELECT SID, NAME, ROLL, SECTION FROM STUDENT")
    for row in cursor:
        tree.insert(
            "",
            0,
            values=(row[0], row[1], row[2], row[3])
        )
    tree.place(x=530, y=100)

# Parse and store data into database and treeview upon clcicking of the add button
def parse_data():
    fid = str(fid_entry.get())
    passw = str(passw_entry.get())
    conf_passw = str(conf_passw_entry.get())
    name = str(name_entry.get()).upper()
    roll = str(roll_entry.get())
    section = str(sec_entry.get()).upper()

    if fid == "" or passw == "" or \
        conf_passw == "" or name == "" or \
        roll == "" or section == "":
```

```
        messagebox.showwarning("Bad Input", "Some fields are empty! Please fill them out!")
        return

    if passw != conf_passw:
        messagebox.showerror("Passwords mismatch", "Password and confirm password didnt match. Try again!")
        passw_entry.delete(0, tk.END)
        conf_passw_entry.delete(0, tk.END)
        return

    conn.execute(f"REPLACE INTO STUDENT (SID, PASSW, NAME, ROLL, SECTION)\
        VALUES ('{fid}','{passw}','{name}', '{roll}', '{section}')")
    conn.commit()
    update_treeview()

    fid_entry.delete(0, tk.END)
    passw_entry.delete(0, tk.END)
    conf_passw_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    roll_entry.delete(0, tk.END)
    sec_entry.delete(0, tk.END)


# update a row in the database
def update_data():
    fid_entry.delete(0, tk.END)
    passw_entry.delete(0, tk.END)
    conf_passw_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    roll_entry.delete(0, tk.END)
    sec_entry.delete(0, tk.END)
    try:
        # print(tree.selection())
        if len(tree.selection()) > 1:
            messagebox.showerror("Bad Select", "Select one student at a time to update!")
            return

        q_fid = tree.item(tree.selection()[0])['values'][0]
        cursor = conn.execute(f"SELECT * FROM STUDENT WHERE SID = '{q_fid}'")

        cursor = list(cursor)
        fid_entry.insert(0, cursor[0][0])
        passw_entry.insert(0, cursor[0][1])
        conf_passw_entry.insert(0, cursor[0][1])
        name_entry.insert(0, cursor[0][2])
        roll_entry.insert(0, cursor[0][3])
        sec_entry.insert(0, cursor[0][4])

        conn.execute(f"DELETE FROM STUDENT WHERE SID = '{cursor[0][0]}'")
        conn.commit()
        update_treeview()

    except IndexError:
        messagebox.showerror("Bad Select", "Please select a student from the list first!")
        return

# remove selected data from databse and treeview
def remove_data():
    if len(tree.selection()) < 1:
        messagebox.showerror("Bad Select", "Please select a student from the list first!")
        return
    for i in tree.selection():
        # print(tree.item(i)['values'][0])
        conn.execute(f"DELETE FROM STUDENT WHERE SID = '{tree.item(i)['values'][0]}'")
        conn.commit()
        tree.delete(i)
```

```
        update_treeview()

# toggles between show/hide password
def show_passw():
    if passw_entry['show'] == "●":
        passw_entry['show'] = ""
        B1_show['text'] = '●'
        B1_show.update()
    elif passw_entry['show'] == "":
        passw_entry['show'] = "●"
        B1_show['text'] = '○'
        B1_show.update()
    passw_entry.update()



# main
if __name__ == "__main__":

    '''
        DATABASE CONNECTIONS AND SETUP
    '''

    # connecting database
    conn = sqlite3.connect(r'files/timetable.db')

    # creating Tabe in the database
    conn.execute('CREATE TABLE IF NOT EXISTS STUDENT\
    (SID CHAR(10) NOT NULL PRIMARY KEY,\
    PASSW CHAR(50) NOT NULL,\
    NAME CHAR(50) NOT NULL,\
    ROLL INTEGER NOT NULL,\
    SECTION CHAR(5) NOT NULL)')

    '''
        TKinter WINDOW SETUP WITH WIDGETS
            * Label(1-11)
            * Entry(6)
            * ComboBox(1-2)
            * Treeview(1)
            * Button(1-3)
    '''

    # TKinter Window
    subtk = tk.Tk()
    subtk.geometry('1000x470')
    subtk.title('Add/Update Students')

    # Label1
    tk.Label(
        subtk,
        text='List of Students',
        font=('times new roman', 20, 'bold')
    ).place(x=620, y=50)

    # Label2
    tk.Label(
        subtk,
        text='Add/Update Students',
        font=('times new roman', 20, 'bold')
    ).place(x=110, y=50)

    # Label3
    tk.Label(
        subtk,
```

```
    text='Add information in the following prompt!',
    font=('times new roman', 10, 'italic')
).place(x=110, y=85)

# Label4
tk.Label(
    subtk,
    text='Student id:',
    font=('times new roman', 12)
).place(x=100, y=130)

# Entry1
fid_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20
)
fid_entry.place(x=260, y=130)

# Label5
tk.Label(
    subtk,
    text='Password:',
    font=('times new roman', 12)
).place(x=100, y=170)

# Entry2
passw_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20,
    show="●"
)
passw_entry.place(x=260, y=170)

B1_show = tk.Button(
    subtk,
    text='○',
    font=('times new roman', 9, 'bold'),
    command=show_passw
)
B1_show.place(x=460,y=170)

# Label6
tk.Label(
    subtk,
    text='Confirm Password:',
    font=('times new roman', 12)
).place(x=100, y=210)

# Entry3
conf_passw_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20,
    show="●"
)
conf_passw_entry.place(x=260, y=210)

# Label7
tk.Label(
    subtk,
    text='Student Name:',
    font=('times new roman', 12)
).place(x=100, y=250)
```

```
# Entry4
name_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=25,
)
name_entry.place(x=260, y=250)

# Label8
tk.Label(
    subtk,
    text='Roll no.:',
    font=('times new roman', 12)
).place(x=100, y=290)

# Entry5
roll_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=5,
)
roll_entry.place(x=260, y=290)

# Label9
tk.Label(
    subtk,
    text='Section:',
    font=('times new roman', 12)
).place(x=100, y=330)

# Entry6
sec_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=5,
)
sec_entry.place(x=260, y=330)

# Button1
B1 = tk.Button(
    subtk,
    text='Add Student',
    font=('times new roman', 12),
    command=parse_data
)
B1.place(x=150,y=400)

# Button2
B2 = tk.Button(
    subtk,
    text='Update Student',
    font=('times new roman', 12),
    command=update_data
)
B2.place(x=410,y=400)

# Treeview1
tree = ttk.Treeview(subtk)
create_treeview()
update_treeview()

# Button3
B3 = tk.Button(
    subtk,
```

```
        text='Delete Student(s)',
        font=('times new roman', 12),
        command=remove_data
    )
    B3.place(x=650,y=400)

    # looping Tkiniter window
    subtk.mainloop()
    conn.close() # close database after all operations
```

Schedular.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3

days = 5
periods = 6
recess_break_aft = 3 # recess after 3rd Period
section = None
butt_grid = []

period_names = list(map(lambda x: 'Period ' + str(x), range(1, 6+1)))
day_names = ['Tuesday', 'Wednesday', 'Thrusday', 'Friday','Saturday']

def update_p(d, p, tree, parent):
    # print(section, d, p, str(sub.get()))

    try:
        if len(tree.selection()) > 1:
            messagebox.showerror("Bad Select", "Select one subject at a time!")
            parent.destroy()
            return
        row = tree.item(tree.selection()[0])['values']
        if row[0] == 'NULL' and row[1] == 'NULL':
            conn.execute(f"DELETE FROM SCHEDULE WHERE ID='{section+str((d*periods)+p)}'")
            conn.commit()
            update_table()
            parent.destroy()
            return

        conn.commit()
        print(row)
        conn.execute(f"REPLACE INTO SCHEDULE (ID, DAYID, PERIODID, SUBCODE, SECTION, FINI)\
            VALUES ('{section+str((d*periods)+p)}', {d}, {p}, '{row[1]}', '{section}', '{row[0]}')")
        conn.commit()
        update_table()

    except IndexError:
        messagebox.showerror("Bad Select", "Please select a subject from the list!")
        parent.destroy()
        return

    parent.destroy()


def process_button(d, p):
    print(d, p)
    add_p = tk.Tk()
    # add_p.geometry('200x500')

    # get subject code list from the database
    cursor = conn.execute("SELECT SUBCODE FROM SUBJECTS")
```

```
    subcode_li = [row[0] for row in cursor]
    subcode_li.insert(0, 'NULL')

    # Label10
    tk.Label(
        add_p,
        text='Select Subject',
        font=('times new roman', 12, 'bold')
    ).pack()

    tk.Label(
        add_p,
        text=f'Day: {day_names[d]}',
        font=('times new roman', 12)
    ).pack()

    tk.Label(
        add_p,
        text=f'Period: {p+1}',
        font=('times new roman', 12)
    ).pack()

    tree = ttk.Treeview(add_p)
    tree['columns'] = ('one', 'two')
    tree.column("#0", width=0, stretch=tk.NO)
    tree.column("one", width=70, stretch=tk.NO)
    tree.column("two", width=80, stretch=tk.NO)
    tree.heading('#0', text="")
    tree.heading('one', text="Faculty")
    tree.heading('two', text="Subject Code")

    cursor = conn.execute("SELECT FACULTY.INI, FACULTY.SUBCODE1, FACULTY.SUBCODE2,
SUBJECTS.SUBCODE\
    FROM FACULTY, SUBJECTS\
    WHERE FACULTY.SUBCODE1=SUBJECTS.SUBCODE OR FACULTY.SUBCODE2=SUBJECTS.SUBCODE")
    for row in cursor:
        print(row)
        tree.insert(
            "",
            0,
            values=(row[0],row[-1])
        )
    tree.insert("", 0, value=('NULL', 'NULL'))
    tree.pack(pady=10, padx=30)

    tk.Button(
        add_p,
        text="OK",
        padx=15,
        command=lambda x=d, y=p, z=tree, d=add_p: update_p(x, y, z, d)
    ).pack(pady=20)

    add_p.mainloop()

def select_sec():
    global section
    section = str(combo1.get())
    print(section)
    update_table()

def update_table():
    for i in range(days):
        for j in range(periods):
            cursor = conn.execute(f"SELECT SUBCODE, FINI FROM SCHEDULE\
                WHERE DAYID={i} AND PERIODID={j} AND SECTION='{section}'")
```

```
            cursor = list(cursor)
            print(cursor)
            if len(cursor) != 0:
                butt_grid[i][j]['text'] = str(cursor[0][0]) + '\n' + str(cursor[0][1])
                butt_grid[i][j].update()
                print(i, j, cursor[0][0])
            else:
                butt_grid[i][j]['text'] = "No Class"
                butt_grid[i][j].update()


# connecting database
conn = sqlite3.connect(r'files/timetable.db')

# creating Tabe in the database
conn.execute('CREATE TABLE IF NOT EXISTS SCHEDULE\
(ID CHAR(10) NOT NULL PRIMARY KEY,\
DAYID INT NOT NULL,\
PERIODID INT NOT NULL,\
SUBCODE CHAR(10) NOT NULL,\
SECTION CHAR(5) NOT NULL,\
FINI CHAR(10) NOT NULL)')
# DAYID AND PERIODID ARE ZERO INDEXED

tt = tk.Tk()

tt.title('Scheduler')

title_lab = tk.Label(
    tt,
    text='T  I  M  E  T  A  B  L  E',
    font=('times new roman', 20, 'bold'),
    pady=5
)
title_lab.pack()

table = tk.Frame(tt)
table.pack()

first_half = tk.Frame(table)
first_half.pack(side='left')

recess_frame = tk.Frame(table)
recess_frame.pack(side='left')

second_half = tk.Frame(table)
second_half.pack(side='left')

recess = tk.Label(
    recess_frame,
    text='B\nR\nE\nA\nK',
    font=('times new roman', 18, 'bold'),
    width=3,
    relief='sunken'
)
recess.pack()

for i in range(days):
    b = tk.Label(
        first_half,
        text=day_names[i],
        font=('times new roman', 12, 'bold'),
        width=9,
        height=2,
        bd=5,
```

```
        relief='raised'
    )
    b.grid(row=i+1, column=0)

for i in range(periods):
    if i < recess_break_aft:
        b = tk.Label(first_half)
        b.grid(row=0, column=i+1)
    else:
        b = tk.Label(second_half)
        b.grid(row=0, column=i)

    b.config(
        text=period_names[i],
        font=('times new roman', 12, 'bold'),
        width=9,
        height=1,
        bd=5,
        relief='raised'
    )

for i in range(days):
    b = []
    for j in range(periods):
        if j < recess_break_aft:
            bb = tk.Button(first_half)
            bb.grid(row=i+1, column=j+1)
        else:
            bb = tk.Button(second_half)
            bb.grid(row=i+1, column=j)

        bb.config(
            text='Hello World!',
            font=('times new roman', 10),
            width=13,
            height=3,
            bd=5,
            relief='raised',
            wraplength=80,
            justify='center',
            command=lambda x=i, y=j: process_button(x, y)
        )
        b.append(bb)

    butt_grid.append(b)
    # print(b)
    b = []
sec_select_f = tk.Frame(tt, pady=15)
sec_select_f.pack()

tk.Label(
    sec_select_f,
    text='Select department:  ',
    font=('times new roman', 12, 'bold')
).pack(side=tk.LEFT)

cursor = conn.execute("SELECT DISTINCT SECTION FROM STUDENT")
sec_li = [row[0] for row in cursor]
# sec_li.insert(0, 'NULL')
print(sec_li)
combo1 = ttk.Combobox(
    sec_select_f,
    values=sec_li,
)
combo1.pack(side=tk.LEFT)
```

```
combo1.current(0)

b = tk.Button(
    sec_select_f,
    text="OK",
    font=('times new roman', 12, 'bold'),
    padx=10,
    command=select_sec
)
b.pack(side=tk.LEFT, padx=10)
b.invoke()

print(butt_grid[0][1], butt_grid[1][1])
update_table()

tt.mainloop()
```

Faculty.py

```
import sqlite3
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sys

fid = passw = conf_passw = name = ini = email = subcode1 = subcode2 = None

'''
    LIST OF FUNCTIONS USED FOR VARIOUS FUNCTIONS THROUGH TKinter INTERFACE
        * create_treeview()
        * update_treeview()
        * parse_data()
        * update data()
        * remove_data()
        * show_passw()
'''

# create treeview (call this function once)
def create_treeview():
    tree['columns'] = list(map(lambda x: '#' + str(x), range(1, 5)))
    tree.column("#0", width=0, stretch=tk.NO)
    tree.column("#1", width=70, stretch=tk.NO)
    tree.column("#2", width=200, stretch=tk.NO)
    tree.column("#3", width=80, stretch=tk.NO)
    tree.column("#4", width=80, stretch=tk.NO)
    tree.heading('#0', text="")
    tree.heading('#1', text="Fid")
    tree.heading('#2', text="Name")
    tree.heading('#3', text="Subject 1")
    tree.heading('#4', text="Subject 2")
    tree['height'] = 15

# update treeview (call this function after each update)
def update_treeview():
    for row in tree.get_children():
        tree.delete(row)
    cursor = conn.execute("SELECT FID, NAME, SUBCODE1, SUBCODE2 FROM FACULTY")
    for row in cursor:
        tree.insert(
            "",
            0,
            values=(row[0], row[1], row[2], row[3])
        )
    tree.place(x=530, y=100)
```

```
# Parse and store data into database and treeview upon clcicking of the add button
def parse_data():
    fid = str(fid_entry.get())
    passw = str(passw_entry.get())
    conf_passw = str(conf_passw_entry.get())
    name = str(name_entry.get()).upper()
    ini = str(ini_entry.get()).upper()
    email = str(email_entry.get())
    subcode1 = str(combo1.get())
    subcode2 = str(combo2.get())

    if fid == "" or passw == "" or \
        conf_passw == "" or name == "":
        messagebox.showwarning("Bad Input", "Some fields are empty! Please fill them out!")
        return

    if passw != conf_passw:
        messagebox.showerror("Passwords mismatch", "Password and confirm password didnt match. Try again!")
        passw_entry.delete(0, tk.END)
        conf_passw_entry.delete(0, tk.END)
        return

    if subcode1 == "NULL":
        messagebox.showwarning("Bad Input", "Subject 1 cant be NULL")
        return

    conn.execute(f"REPLACE INTO FACULTY (FID, PASSW, NAME, INI, EMAIL, SUBCODE1, SUBCODE2)\
        VALUES ('{fid}','{passw}','{name}', '{ini}', '{email}', '{subcode1}', '{subcode2}')")
    conn.commit()
    update_treeview()

    fid_entry.delete(0, tk.END)
    passw_entry.delete(0, tk.END)
    conf_passw_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    ini_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)
    combo1.current(0)
    combo2.current(0)


# update a row in the database
def update_data():
    fid_entry.delete(0, tk.END)
    passw_entry.delete(0, tk.END)
    conf_passw_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    ini_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)
    combo1.current(0)
    combo2.current(0)
    try:
        # print(tree.selection())
        if len(tree.selection()) > 1:
            messagebox.showerror("Bad Select", "Select one faculty at a time to update!")
            return

        q_fid = tree.item(tree.selection()[0])['values'][0]
        cursor = conn.execute(f"SELECT * FROM FACULTY WHERE FID = '{q_fid}'")

        cursor = list(cursor)
        fid_entry.insert(0, cursor[0][0])
        passw_entry.insert(0, cursor[0][1])
        conf_passw_entry.insert(0, cursor[0][1])
        name_entry.insert(0, cursor[0][2])
```

```
        ini_entry.insert(0, cursor[0][3])
        email_entry.insert(0, cursor[0][4])
        combo1.current(subcode_li.index(cursor[0][5]))
        combo2.current(subcode_li.index(cursor[0][6]))

        conn.execute(f"DELETE FROM FACULTY WHERE FID = '{cursor[0][0]}'")
        conn.commit()
        update_treeview()

    except IndexError:
        messagebox.showerror("Bad Select", "Please select a faculty from the list first!")
        return

# remove selected data from databse and treeview
def remove_data():
    if len(tree.selection()) < 1:
        messagebox.showerror("Bad Select", "Please select a faculty from the list first!")
        return
    for i in tree.selection():
        # print(tree.item(i)['values'][0])
        conn.execute(f"DELETE FROM FACULTY WHERE FID = '{tree.item(i)['values'][0]}'")
        conn.commit()
        tree.delete(i)
        update_treeview()

# toggles between show/hide password
def show_passw():
    if passw_entry['show'] == "●":
        passw_entry['show'] = ""
        B1_show['text'] = '●'
        B1_show.update()
    elif passw_entry['show'] == "":
        passw_entry['show'] = "●"
        B1_show['text'] = '○'
        B1_show.update()
    passw_entry.update()



# main
if __name__ == "__main__":

    '''
        DATABASE CONNECTIONS AND SETUP
    '''

    # connecting database
    conn = sqlite3.connect(r'files/timetable.db')

    # creating Tabe in the database
    conn.execute('CREATE TABLE IF NOT EXISTS FACULTY\
    (FID CHAR(10) NOT NULL PRIMARY KEY,\
    PASSW CHAR(50) NOT NULL,\
    NAME CHAR(50) NOT NULL,\
    INI CHAR(5) NOT NULL,\
    EMAIL CHAR(50) NOT NULL,\
    SUBCODE1 CHAR(10) NOT NULL,\
    SUBCODE2 CHAR(10)    )')

    '''
        TKinter WINDOW SETUP WITH WIDGETS
            * Label(1-11)
            * Entry(6)
            * ComboBox(1-2)
            * Treeview(1)
```

```
        * Button(1-3)
'''

# TKinter Window
subtk = tk.Tk()
subtk.geometry('1000x550')
subtk.title('Add/Update Faculties')

# Label1
tk.Label(
    subtk,
    text='List of Faculties',
    font=('times new roman', 20, 'bold')
).place(x=600, y=50)

# Label2
tk.Label(
    subtk,
    text='Add/Update Faculties',
    font=('times new roman', 20, 'bold')
).place(x=90, y=50)

# Label3
tk.Label(
    subtk,
    text='Add information in the following prompt!',
    font=('times new roman', 10, 'italic')
).place(x=100, y=85)

# Label4
tk.Label(
    subtk,
    text='Faculty id:',
    font=('times new roman', 12)
).place(x=100, y=130)

# Entry1
fid_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20
)
fid_entry.place(x=260, y=130)

# Label5
tk.Label(
    subtk,
    text='Password:',
    font=('times new roman', 12)
).place(x=100, y=170)

# Entry2
passw_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20,
    show="●"
)
passw_entry.place(x=260, y=170)

B1_show = tk.Button(
    subtk,
    text='○',
    font=('times new roman', 9, 'bold'),
    command=show_passw
```

```
)
B1_show.place(x=460,y=170)

# Label6
tk.Label(
    subtk,
    text='Confirm Password:',
    font=('times new roman', 12)
).place(x=100, y=210)

# Entry3
conf_passw_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=20,
    show="●"
)
conf_passw_entry.place(x=260, y=210)

# Label7
tk.Label(
    subtk,
    text='Faculty Name:',
    font=('times new roman', 12)
).place(x=100, y=250)

# Entry4
name_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=25,
)
name_entry.place(x=260, y=250)

# Label8
tk.Label(
    subtk,
    text='Initials:',
    font=('times new roman', 12)
).place(x=100, y=290)

# Entry5
ini_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=5,
)
ini_entry.place(x=260, y=290)

# Label9
tk.Label(
    subtk,
    text='Email:',
    font=('times new roman', 12)
).place(x=100, y=330)

# Entry6
email_entry = tk.Entry(
    subtk,
    font=('times new roman', 12),
    width=25,
)
email_entry.place(x=260, y=330)

# get subject code list from the database
```

```
cursor = conn.execute("SELECT SUBCODE FROM SUBJECTS")
subcode_li = [row[0] for row in cursor]
subcode_li.insert(0, 'NULL')

# Label10
tk.Label(
    subtk,
    text='Subject 1:',
    font=('times new roman', 12)
).place(x=100, y=370)

# ComboBox1
combo1 = ttk.Combobox(
    subtk,
    values=subcode_li,
)
combo1.place(x=260, y=370)
combo1.current(0)

# Label11
tk.Label(
    subtk,
    text='Subject 2:',
    font=('times new roman', 12)
).place(x=100, y=410)

# ComboBox2
combo2 = ttk.Combobox(
    subtk,
    values=subcode_li,
)
combo2.place(x=260, y=410)
combo2.current(0)

# Button1
B1 = tk.Button(
    subtk,
    text='Add Faculty',
    font=('times new roman', 12),
    command=parse_data
)
B1.place(x=150,y=465)

# Button2
B2 = tk.Button(
    subtk,
    text='Update Faculty',
    font=('times new roman', 12),
    command=update_data
)
B2.place(x=410,y=465)

# Treeview1
tree = ttk.Treeview(subtk)
create_treeview()
update_treeview()

# Button3
B3 = tk.Button(
    subtk,
    text='Delete Faculty(s)',
    font=('times new roman', 12),
    command=remove_data
)
B3.place(x=650,y=465)
```

```
    # looping Tkiniter window
    subtk.mainloop()
    conn.close() # close database after all operations
```

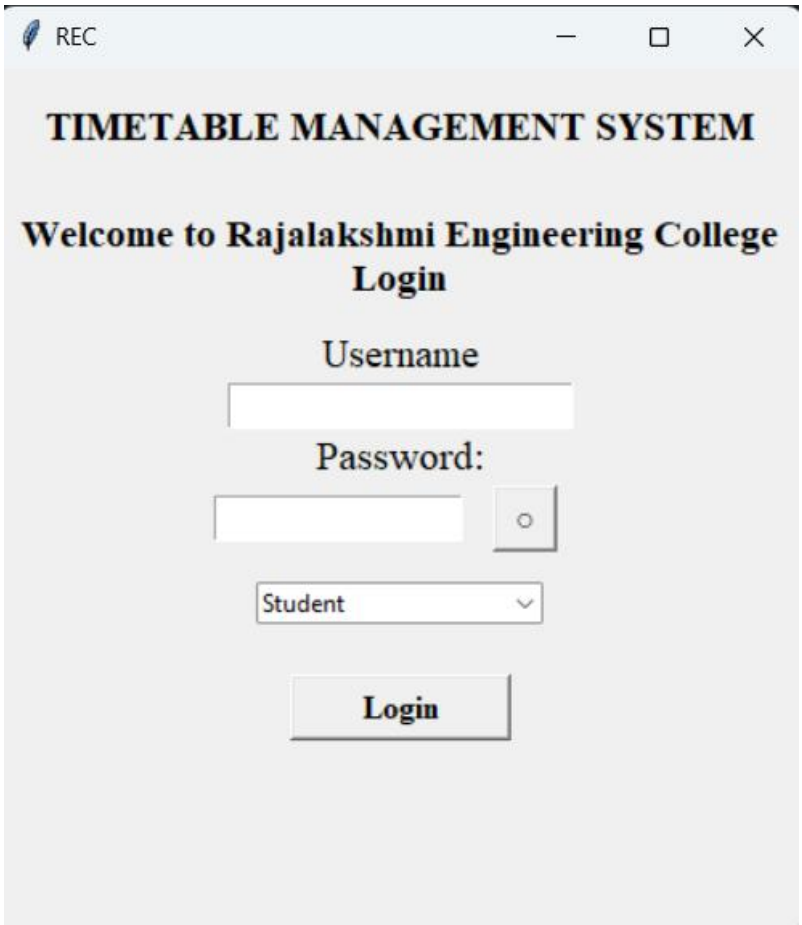
Admin_screen.py

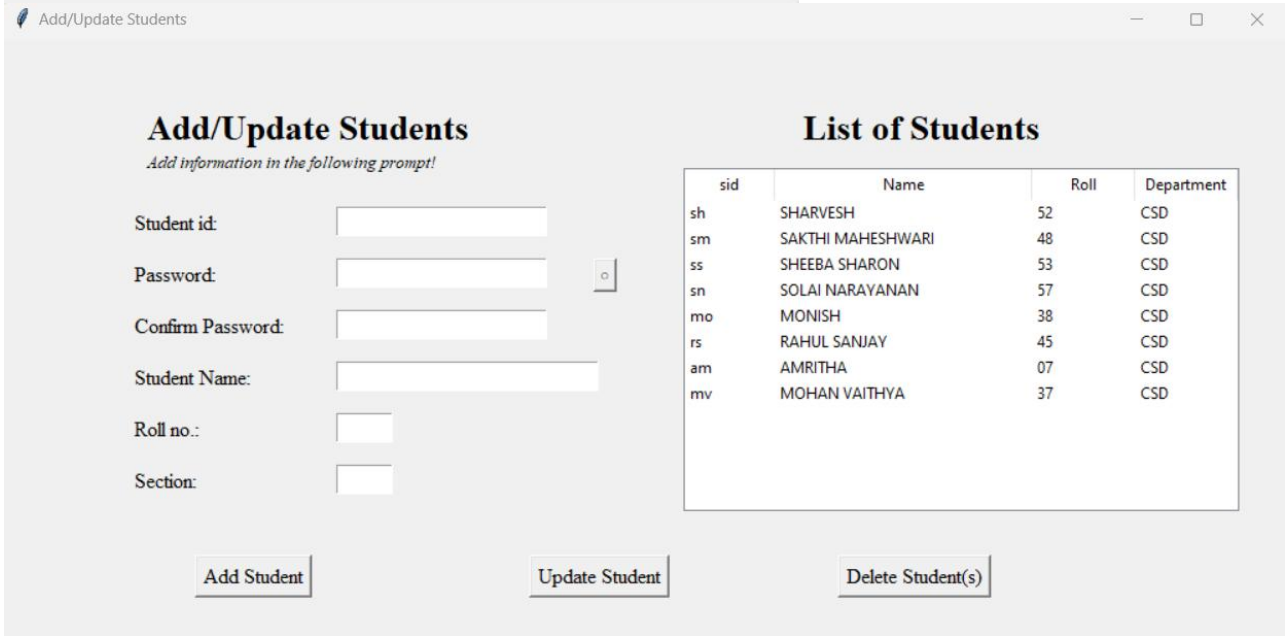
```
import tkinter as tk
import sys
import os
import threading

def run_sub(): os.system('pythonw windows\\subjects.py')
def run_fac(): os.system('pythonw windows\\faculty.py')
def run_stud(): os.system('pythonw windows\\student.py')
def run_sch(): os.system('pythonw windows\\scheduler.py')
def run_tt_s(): os.system('pythonw windows\\timetable_stud.py')
def run_tt_f(): os.system('pythonw windows\\timetable_fac.py')

ad = tk.Tk()
ad.geometry('500x430')

ad.title('Administrator')

tk.Label(
    ad,
    text='A D M I N I S T R A T O R',
    font=('Consolas', 20, 'bold'),
    pady=10
).pack()

tk.Label(
    ad,
    text='You are the Administrator',
    font=('Consolas', 12, 'italic'),
).pack(pady=9)

modify_frame = tk.LabelFrame(text='Modify', font=('Consolas'), padx=20)
modify_frame.place(x=50, y=100)

tk.Button(
    modify_frame,
    text='Subjects',
    font=('Consolas'),
    command=run_sub
).pack(pady=20)

tk.Button(
    modify_frame,
    text='Faculties',
    font=('Consolas'),
    command=run_fac
).pack(pady=20)

tk.Button(
    modify_frame,
    text='Students',
    font=('Consolas'),
    command=run_stud
).pack(pady=20)

tt_frame = tk.LabelFrame(text='Timetable', font=('Consolas'), padx=20)
tt_frame.place(x=250, y=100)
```

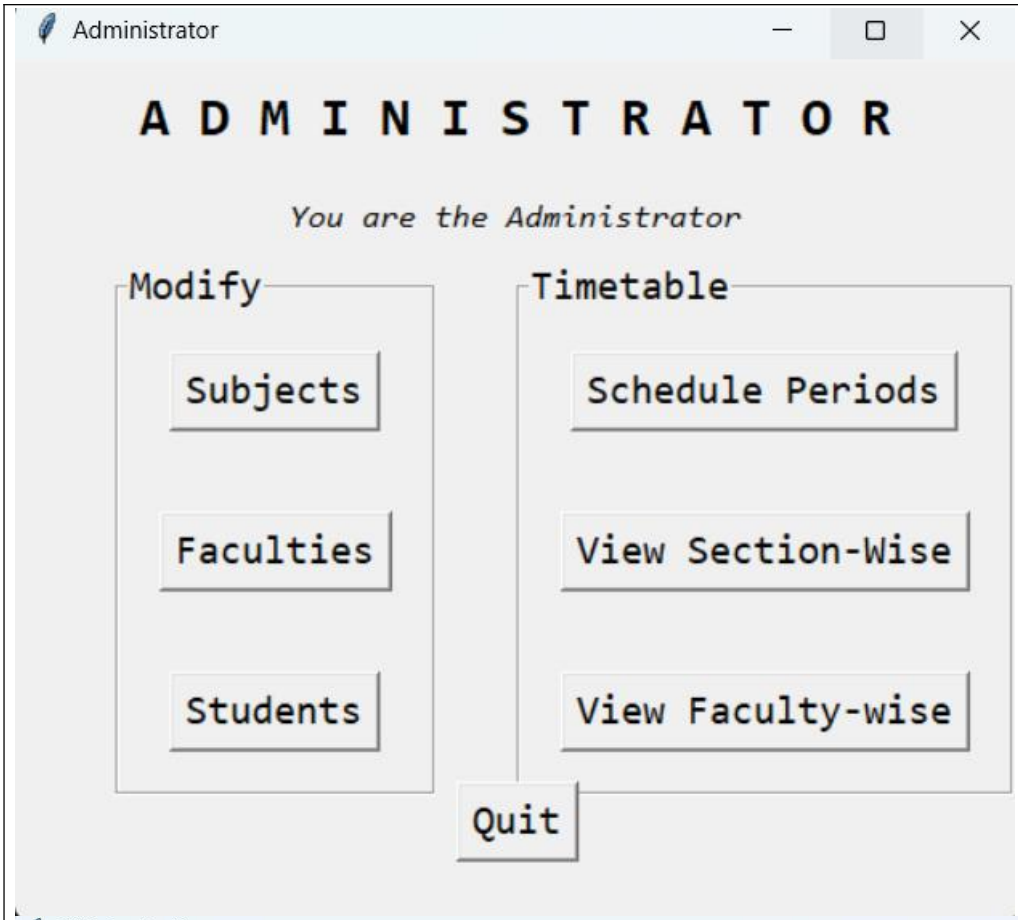
```
tk.Button(
    tt_frame,
    text='Schedule Periods',
    font=('Consolas'),
    command=run_sch
).pack(pady=20)

tk.Button(
    tt_frame,
    text='View Section-Wise',
    font=('Consolas'),
    command=run_tt_s
).pack(pady=20)

tk.Button(
    tt_frame,
    text='View Faculty-wise',
    font=('Consolas'),
    command=run_tt_f
).pack(pady=20)

tk.Button(
    ad,
    text='Quit',
    font=('Consolas'),
    command=ad.destroy
).place(x=220, y=360)

ad.mainloop()
```

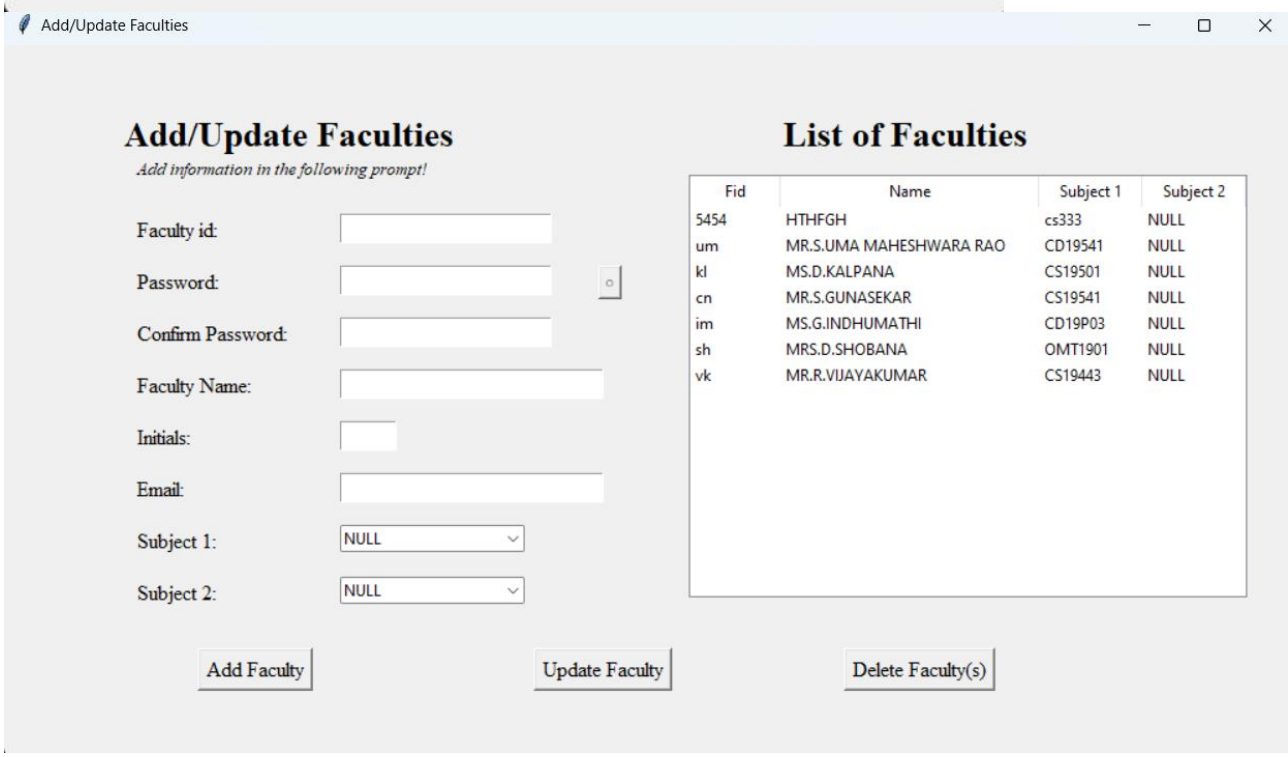
CHAPTER - 5

SCREEN SHOTS






Administrator
ADMINISTRATOR
You are the Administrator
Modify
Subjects
Faculties
Students
Timetable
Schedule Periods
View Section-Wise
View Faculty-wise
Quit


Add/Update Faculties
Add/Update Faculties
Add information in the following prompt!
Faculty id:
Password:
Confirm Password:
Faculty Name:
Initials:
Email:
Subject 1:
NULL
Subject 2:
NULL
Add Faculty
Update Faculty
Delete Faculty(s)
List of Faculties
Fid
Name
Subject 1
Subject 2
5454 HTHFGH cs333 NULL
um MR.S.UMA MAHESHWARA RAO CD19541 NULL
kl MS.D.KALPANA CS19501 NULL
cn MR.S.GUNASEKAR CS19541 NULL
im MS.G.INDHUMATHI CD19P03 NULL
sh MRS.D.SHOBANA OMT1901 NULL
vk MR.R.VIJAYAKUMAR CS19443 NULL

Student Timetable

T I M E T A B L E

Type: Theory Classes | Practical Classes

	Period 1	Period 2	Period 3	Period 4		Period 5	Period 6
Tuesday	CS19443 VIJAYAKUMAR	CS19541 GUNASEKAR	CS19443 VIJAYAKUMAR	CD19541 UMA MAHESHWARA RAO	LUNCH	CS19443 VIJAYAKUMAR	CS19443 VIJAYAKUMAR
Wednesday	CD19P03 INDHUMATHI	CD19P03 INDHUMATHI	CD19541 UMA MAHESHWARA RAO	CD19541 UMA MAHESHWARA RAO		CD19P03 INDHUMATHI	CS19541 GUNASEKAR
Thrusday	OMT1901 SHOBANA	OMT1901 SHOBANA	CS19443 VIJAYAKUMAR	CS19443 VIJAYAKUMAR		CS19541 GUNASEKAR	CS19541 GUNASEKAR
Friday	CS19501 KALPANA	CD19541 UMA MAHESHWARA RAO	CD19P03 INDHUMATHI	CS19501 KALPANA		CS19541 GUNASEKAR	CS19541 GUNASEKAR
Saturday	CD19P03 INDHUMATHI	CD19P03 INDHUMATHI	OMT1901 SHOBANA	CS19501 KALPANA		CS19541 GUNASEKAR	CD19P03 INDHUMATHI

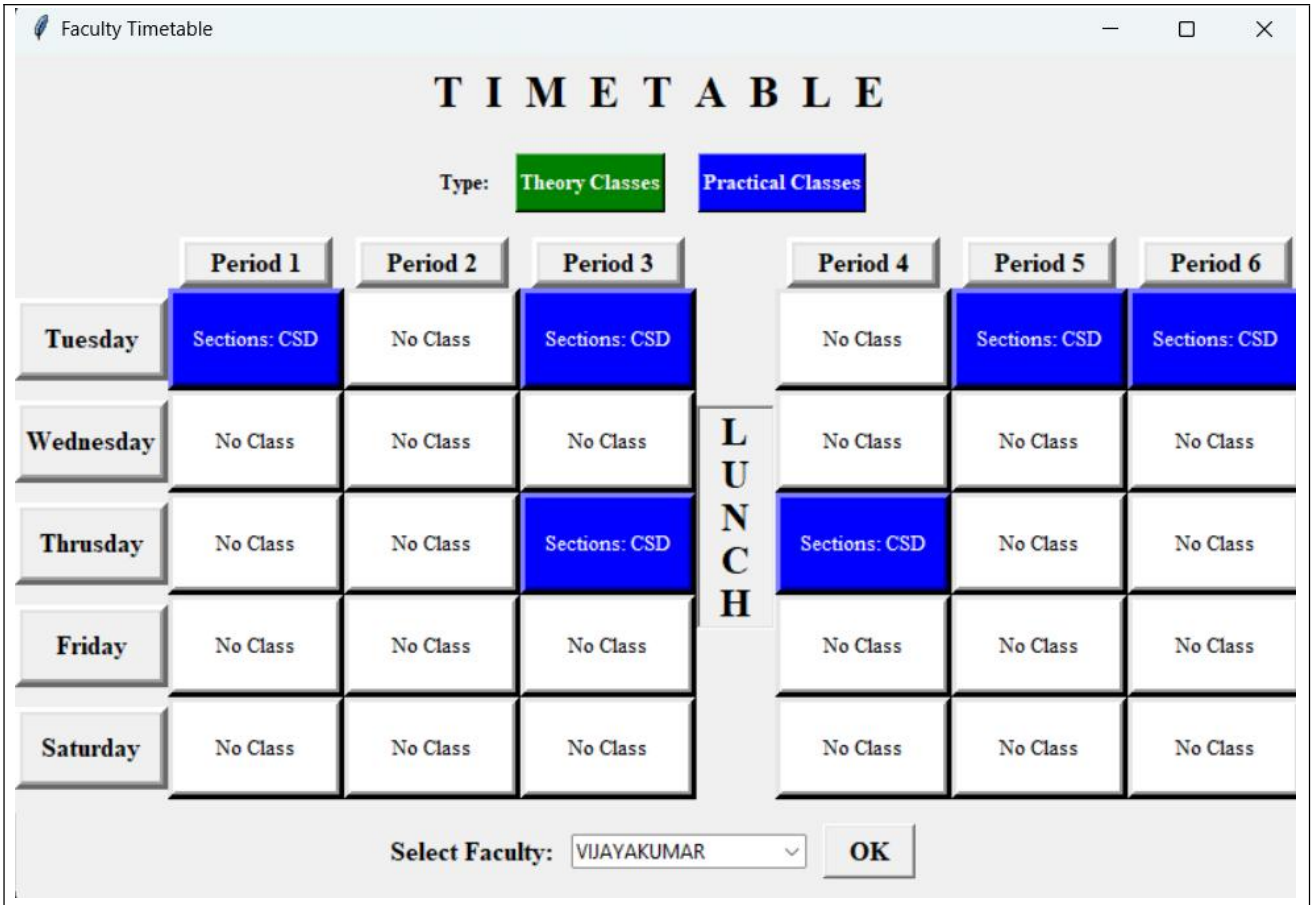
Select Department: CSD OK

Scheduler

T I M E T A B L E

	Period 1	Period 2	Period 3		Period 4	Period 5	Period 6
Tuesday	CS19443 VIJAYAKUMAR	CS19541 GUNASEKAR	CS19443 VIJAYAKUMAR		CD19541 UMA MAHESHWARA RAO	CS19443 VIJAYAKUMAR	CS19443 VIJAYAKUMAR
Wednesday	CD19P03 INDHUMATHI	CD19P03 INDHUMATHI	CD19541 UMA MAHESHWARA RAO	BREAK	CD19541 UMA MAHESHWARA RAO	CD19P03 INDHUMATHI	CS19541 GUNASEKAR
Thrusday	OMT1901 SHOBANA	OMT1901 SHOBANA	CS19443 VIJAYAKUMAR		CS19443 VIJAYAKUMAR	CS19541 GUNASEKAR	CS19541 GUNASEKAR
Friday	CS19501 KALPANA	CD19541 UMA MAHESHWARA RAO	CD19P03 INDHUMATHI		CS19501 KALPANA	CS19541 GUNASEKAR	CS19541 GUNASEKAR
Saturday	CD19P03 INDHUMATHI	CD19P03 INDHUMATHI	OMT1901 SHOBANA		CS19501 KALPANA	CS19541 GUNASEKAR	CD19P03 INDHUMATHI

Select department: CSD OK


Faculty Timetable
TIMETABLE
Type:
Theory Classes
Practical Classes
Period 1
Period 2
Period 3
Period 4
Period 5
Period 6
Tuesday
Sections: CSD
No Class
Sections: CSD
No Class
Sections: CSD
Sections: CSD
Wednesday
No Class
No Class
No Class
No Class
No Class
No Class
Thrusday
No Class
No Class
Sections: CSD
Sections: CSD
No Class
No Class
Friday
No Class
No Class
No Class
No Class
No Class
No Class
Saturday
No Class
No Class
No Class
No Class
No Class
No Class
LUNCH
Select Faculty:
VIJAYAKUMAR
OK

CHAPTER – 6

Conclusion

The Timetable Management System provides an effective solution for managing and organizing schedules, helping users stay on top of their academic and personal commitments. Easy-to-use interfaces, the system ensures improved time management and reduced stress. By leveraging Python and SQLite, the project offers a reliable, cross-platform solution that efficiently stores and retrieves data. Ultimately, this system enhances productivity and helps users maintain a balanced and organized routine, making it a valuable tool for students and professionals alike.

CHAPTER – 7

BIBLIOGRAPHY:

Website referances

https://www.youtube.com/watch?v=dam0GPOAvVI
https://www.youtube.com/watch?v=0ZaC6JaNpic&list=PLeo1K3hjS3uu1hh_qzBt6e379cofVD9Sb
https://www.youtube.com/watch?v=JrAiefGNUq8
https://www.learnpython.org/
https://www.sqlitetutorial.net/