FitBuddy Android App

A Project Report

Submitted in the partial fulfillment of the requirement for the degree

of **Bachelor of Technology**

in

Computer Science & Engineering

by

Nazeem Ahmad (1904230100036)
Komal Kumari (190423010026)

Under the supervision of

Miss. Niyati Gaur
Assistant Professor, School of Management Sciences, Lucknow

to the

Department of Computer Science & Engineering
School of Management Sciences , Lucknow

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

May, 2023

DECLARATION

We Nazeem Ahmad and Komal Kumari hereby declare that the project work entitled **" FitBuddy Android App "** is an authenticated work carried out by us at School of Management Sciences Lucknow Under the guidance Miss. Niyati Gaur for the partial fulfillment of the award of the degree of **COMPUTER SCIENCE & ENGINEERING.** and this work has not been submitted for similar purpose anywhere else except to **School of Management Sciences, Lucknow.**

Date: **Nazeem Ahmad**

Place: Lucknow **Komal Kumari**

CERTIFICATE

This is to certify that the project report entitled " **FitBuddy Android App** " submitted by " Komal Kumari (1904230100026) and Nazeem Ahmad (1904230100036) to Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a bonafide record of work carried out by him/her in the partial fulfillment forthe award of the degree Computer Science & Engineering.

Miss. Niyati Gaur
(Project Supervisor)

Mr. Sunit Kumar Mshra
HoD (CS&E)

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B.Tech. Project undertaken during B.Tech. final Year. We owe special debt of gratitude to my faculty supervisor, **Miss. Niyati Gaur** and all the professors of Computer Science & Engineering department, School of Management Science Lucknow for their constant support and guidance throughout the course of our work. We also take the opportunity to acknowledge the contribution of Professor **Mr. Sunit Kumar Mishra** Head, Department of Computer Science & Engineering, School of management Science Lucknow for his full Support and assistance during the development of the project.

Name: Nazeem Ahmad

Roll No: 1904230100036

Name: Komal Kumari

Roll No: 1904230100026

Date:

CHAPTERS

Chapters of a project report may be broadly divided in following manner

Chapter 1 - Introduction

- Overview of the project
- Objectives & Scope

Chapter 2- Requirement Analysis and Specification

- Feasibility Study
- Technical Feasibility
- Economical Feasibility
- Operational Feasibility
- Problem Definition
- Hardware and Software Specification

Chapter 3 – System Design

- Data Flow Diagrams, Entity Relationship Diagrams

Chapter 4 - Technologies Used

Chapter 5- Coding

Chapter 6 - Testing

- Introduction

List of Tables and Figures

CHAPTER 1 - INTRODUCTION

Overview of the project

The FitBuddy Android App is an application designed to work as a pocket trainer for users. It has user-friendly interface whit makes it easier to use and allows user to works users on their fitness and track their progress.It offers various features like Personalized workout plans, Customized Diet chart, Progress tracker, BMI Calculator, Step count, Workout and meal reminders and many more. This Application is build using Java, Kotlin, Xml, Json and it is totally build on Android Studio. It has separate modules for every feature and uses GIF to demonstrate the exercises to the users. To store users progress and other data it uses Firebase Database with also allows username and password based authentication for first login in device. It have dynamic features like Step count, Calorie meter, etc which uses phones sensors to work with and dose not require any fitness band. This app is perfect solution for people who travel a lot, who are new to fitness and don't have enough money to invest in fitness initially, people having specific time problem to dedicate for fitness. It is portable and can be accessible from any location.

Objectives & Scope

The main objectives of the FitBuddy Android App are:

- Develop a fitness Android app that is easy to use and helps users track their workouts, progress, and goals.
- To provide users a pocket friendly trainer.
- To help those who have to frequently change their locations and hence can't take gym membership.
- A progress tracker that shows users their progress over time, including their weight, body fat percentage.

- To ensure that app is portable and easy to use.

Scope:

The FitBuddy Android App is your personal fitness trainer in your pocket. It's the perfect app for people who are new to fitness, people who cannot afford a personal trainer, and people who travel a lot. FitBuddy offers personalized workout plans, customized diet chart, progress tracker, BMI Calculator, Step count, etc. The app has a wide variety of workouts to choose from, including cardio, strength training. FitBuddy tracks your progress so you can see how far you've come. You can also track your workouts, calories burned, and steps taken. FitBuddy is the perfect way to stay motivated on your fitness journey. You can also set fitness goals and track your progress towards those goals. It’s a perfect fitness app for anyone who wants to get fit, stay fit and wants a start with busy schedule and minimum investment initially.

Chapter 2- Requirement Analysis and Specification

Feasibility Study

Before embarking on the development of the FitBuddy Android App and conducting a thorough analysis of the existing functionalities and required features, it is essential to perform a feasibility study for the project. While all projects can be deemed feasible when unlimited resources and infinite time are available, the feasibility study entails evaluating different approaches to address the given problem. The proposed solution must fulfill all user requirements and exhibit flexibility to accommodate future changes based on upcoming requirements.

Technical Feasibility

Assess the technical capabilities and resources required for implementing and maintaining the system. Determine if the proposed solution aligns with the existing technology infrastructure and anticipate any potential technical challenges.

Economical Feasibility

Evaluate the project's financial aspects, including development, deployment, and maintenance costs. Analyze the potential return on investment (ROI) and ascertain whether the benefits outweigh the associated expenses.

Operational Feasibility

Examine the operational impact of implementing the system, such as changes in workflow, staff training needs, and overall efficiency improvements. Identify any potential obstacles or resistance to change from users and stakeholders.

Problem Definition

In today’s time most of the gyms are having their own trainers with a paid membership that everyone can't afford initially, it also creates problem for those who have to travel a lot and constantly have relocate for their work.

consuming to fetch the data and in proper manner

Hardware Requirements

Intel Core i3 Processor or above :

A quad-core processor is a chip with four independent units called cores that read and execute central processing unit (CPU) instructions such as add, move data, and branch. Within the chip, each core operates in conjunction with other circuits such as cache, memory management, and input/output (I/O) ports. The individual cores in a quad-core processor can run multiple instructions at the same time, increasing the overall speed for programs compatible with parallel processing. Manufacturers typically integrate the cores onto a single semiconductor wafer, or onto multiple semiconductor wafers within a single IC (integrated circuit) package.

RAM : Minimum 500MB

RAM, or random-access memory, is a type of computer main memory in which certain contents may be retrieved directly by the central processing unit in a very short amount of time, independent of the order (and hence location) in which they were recorded. Random-access circuits can support two forms of memory: static RAM (SRAM) and dynamic RAM (DRAM).

HARD DISK : 2GB

Hard disk, also called hard disk drive or hard drive, magnetic storage medium for a computer. Hard disks are flat circular plates made of aluminum or glass and coated with a magnetic material. Hard disks for personal computers can store terabytes (trillions of bytes) of information. Data are stored on their surfaces in concentric tracks. A small electromagnet, called a magnetic head, writes a binary digit (1 or 0) by magnetizing tiny spots on the spinning disk in different directions and reads digits by detecting the magnetization direction of the spots. A computer’s hard drive is a device consisting of several hard disks, read/write heads, a drive motor to spin the disks, and a small amount of circuitry, all sealed in a metal case to protect the disks from dust. In addition to referring to the disks themselves, the term hard disk is also used to refer to the whole of a computer’s internal data storage. Beginning in the early 21st century, some personal computers and laptops were produced that used solid-state drives (SSDs) that relied on flash memory chips instead of hard disks to store information

Software Requirements:-

Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Sqlite3

SQLite in general is a server-less database that you can use within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database. SQLite is a lightweight database that can provide a relational database management system with zero-configuration because there is no need to configure or set up anything to use it.

Operating System : Window 10 or above

An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface, such as a command-line interface (CLI) or a graphical UI (GUI).

Chapter 3- System Design

Data Flow Diagrams

Level 0:

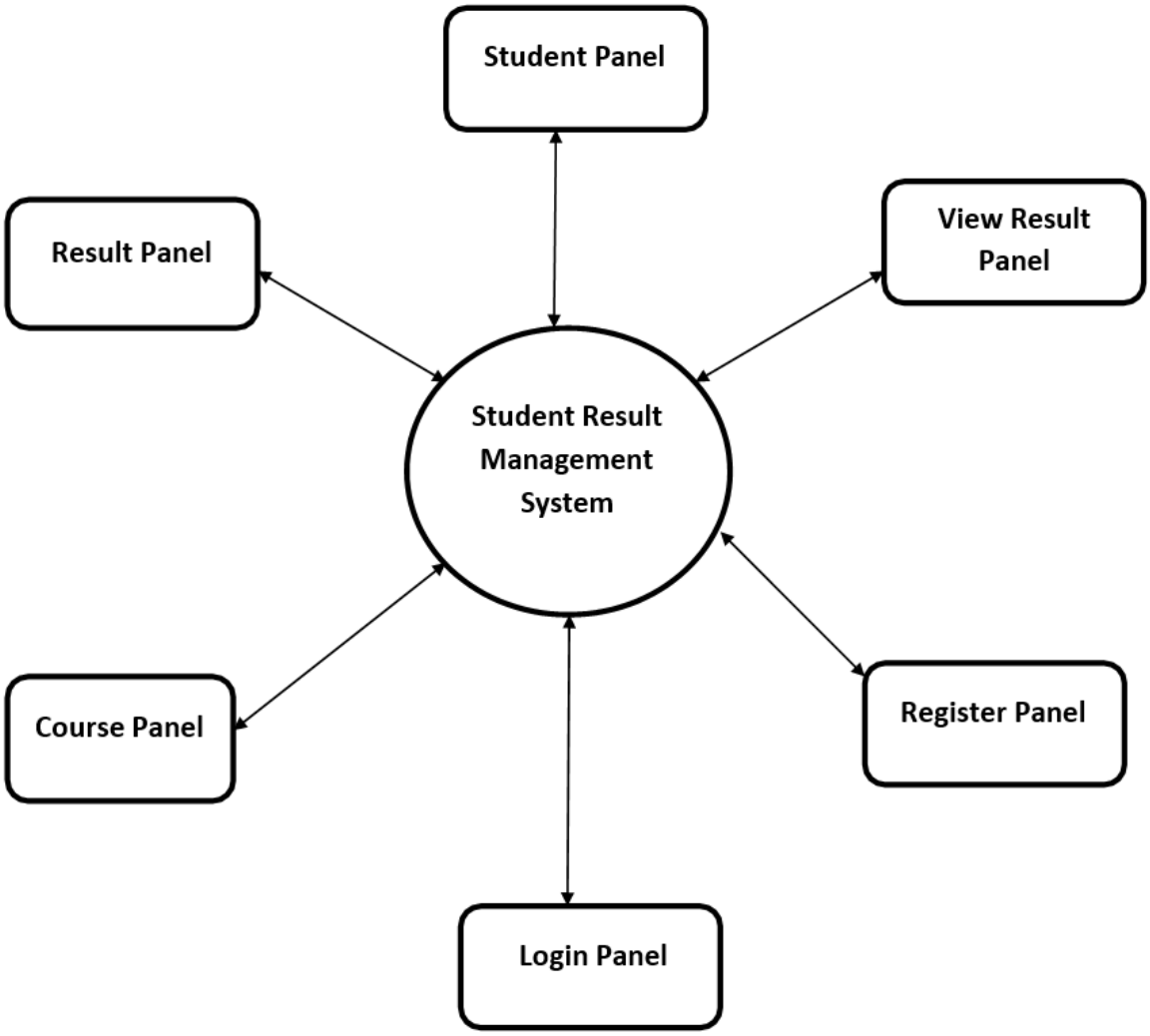

Fig. (3.1)

Level 1:

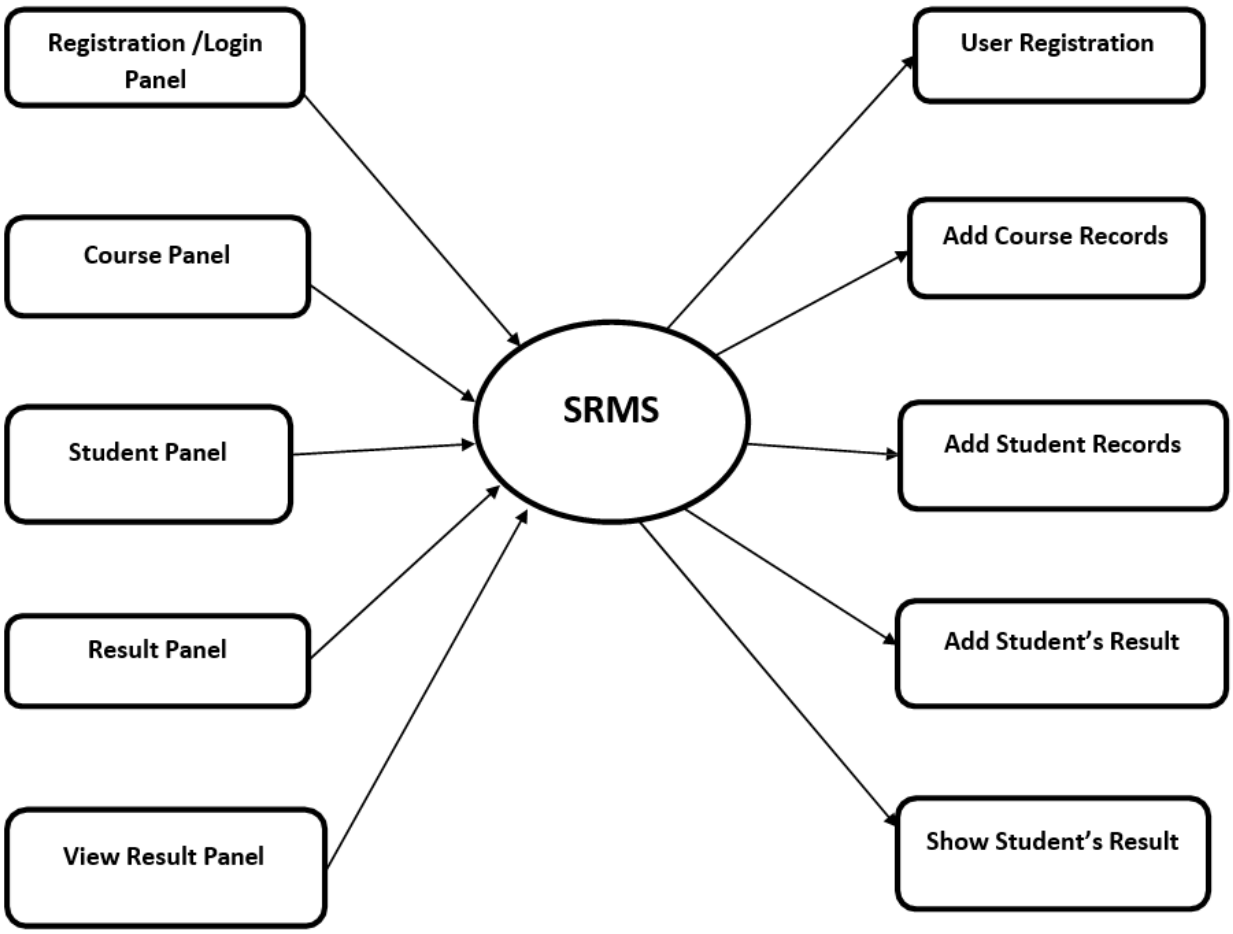

Fig. (3.2)

Level 2:

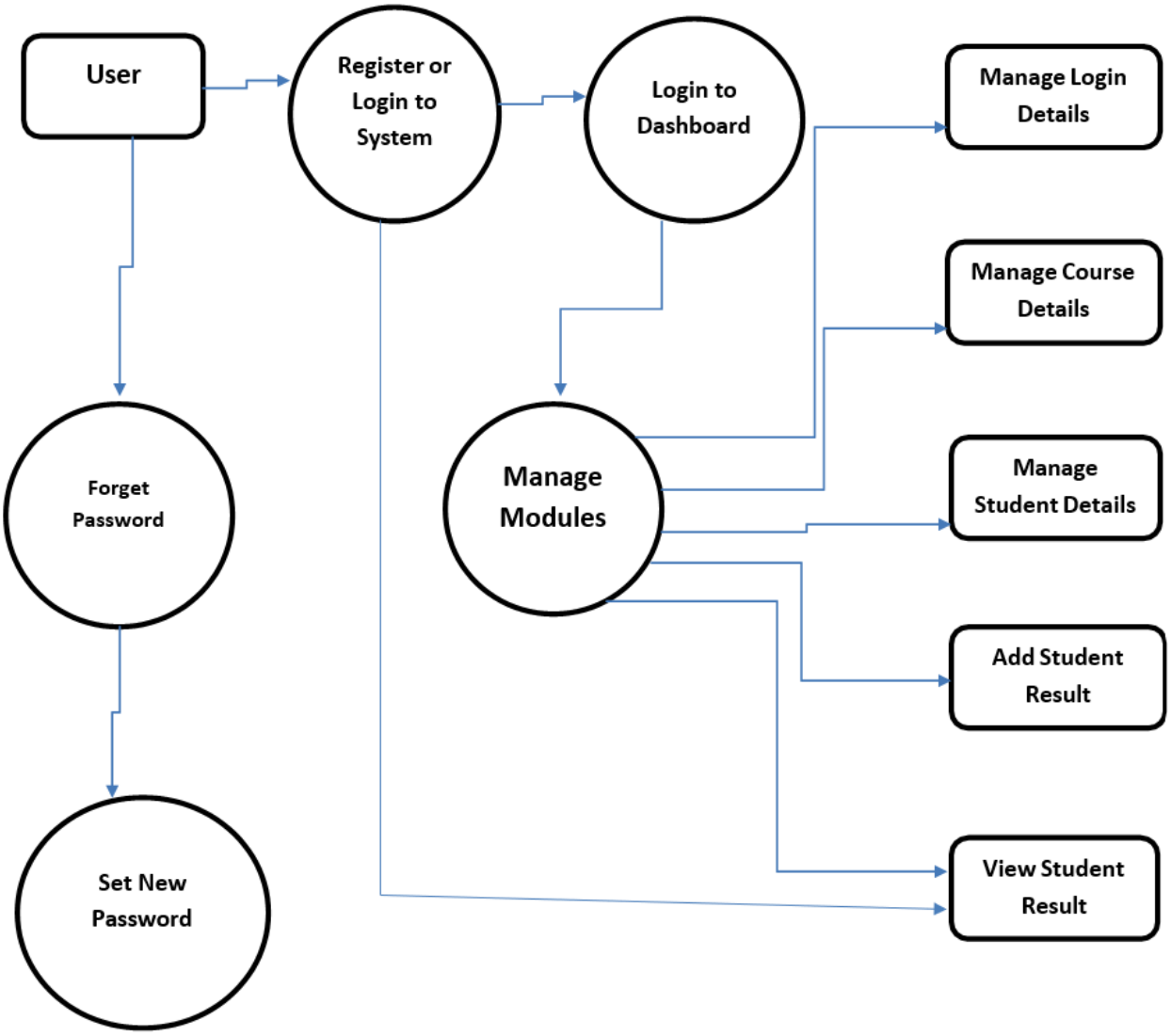

Fig. (3.3)

ER Diagram

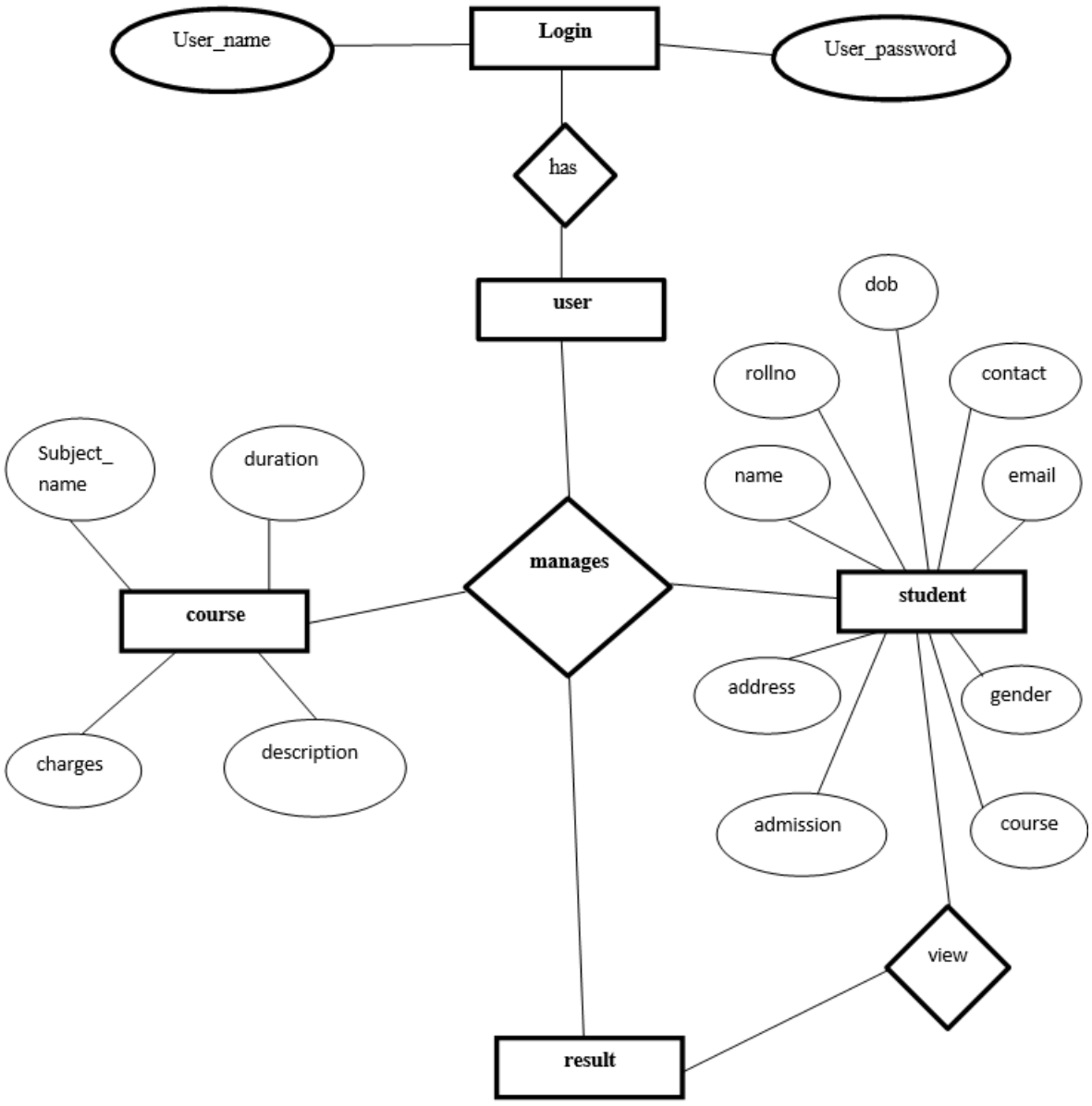

Fig. (3.4)

Chapter 4 – Technology Used

Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Sqlite3

SQLite in general is a server-less database that you can use within almost all programming languages including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database. SQLite is a lightweight database that can provide a relational database management system with zero-configuration because there is no need to configure or set up anything to use it.

Operating System : Window 10 or above

An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface, such as a command-line interface (CLI) or a graphical UI (GUI).

Chapter 5 – Coding

Registration Window Code:

```
from tkinter import*
from tkinter import ttk,messagebox
from PIL import Image,ImageTk
import sqlite3
import os

class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("Registration Window")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="white")
          #=============background colors==============================
        left_lbl=Label(self.root,bg="#031F3C",bd=0)
        left_lbl.place(x=0,y=0,relheight=1,width=750)

        right_lbl=Label(self.root,bg="#08A3D2",bd=0)
        right_lbl.place(x=750,y=0,relheight=1,relwidth=1)

        #=====left Image======
        self.left=Image.open("image/left.png")
        self.left=ImageTk.PhotoImage(self.left)

left=Label(self.root,image=self.left).place(x=80,y=100,width=400,height=500)
        #=======Register Frame====
        frame1=Frame(self.root,bg="white")
        frame1.place(x=480,y=100,width=700,height=500)
        #======title=======
```

```
        title=Label(frame1,text="REGISTER HERE",font=("times new
roman",20,"bold"),bg="white",fg="green").place(x=50,y=30)

        #=======row1=====


        f_name=Label(frame1,text="First Name",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)

        self.txt_f_name=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_f_name.place(x=50,y=130,width=250)


        l_name=Label(frame1,text="Last Name",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=100)

        self.txt_l_name=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_l_name.place(x=370,y=130,width=250)


        #==========Row2========

        contact=Label(frame1,text="Contact No.",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=170)

        self.txt_contact=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_contact.place(x=50,y=200,width=250)


        email=Label(frame1,text="Email",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=170)

        self.txt_email=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_email.place(x=370,y=200,width=250)


        #==========Row3========

        question=Label(frame1,text="Security Question",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=240)


        self.cmb_quest=ttk.Combobox(frame1,font=("times new
roman",13),state='readonly',justify=CENTER)

        self.cmb_quest['values']=("Select","Your First Pet Name","Your Birth
Place","Your Best Friend Name","In what city or town was your first
job?","What elementary school/high school did you attend?","What is your
```

```
favorite movie?","What was your favorite sport in high school? ","What was
the first exam you failed? ","What is your best friend name?")

        self.cmb_quest.place(x=50,y=270,width=250)

        self.cmb_quest.current(0)


        answer=Label(frame1,text="Answer",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=240)

        self.txt_answer=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_answer.place(x=370,y=270,width=250)


        password=Label(frame1,text="Password",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=310)

        self.txt_password=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_password.place(x=50,y=340,width=250)


        cpassword=Label(frame1,text="Confirm Password",font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=310)

        self.txt_cpassword=Entry(frame1,font=("times new
roman",15,),bg='lightgray')

        self.txt_cpassword.place(x=370,y=340,width=250)

         #============terms==============

        self.var_chk=IntVar()

        chk=Checkbutton(frame1,text="I Agree The Terms &
Conditions",variable=self.var_chk,onvalue=1,offvalue=0,bg="white",font=("time
s new roman",12)).place(x=50,y=380)


        btn_register=Button(frame1,text='Register',font=("times new
roman",16),bg="green",fg="white",cursor="hand2",command=self.register_data).p
lace(x=150,y=430,width=150,height=40)

        btn_login=Button(frame1,text='Sign
In',command=self.login_window,font=("times new
roman",16),bg="blue",fg="white",cursor="hand2").place(x=380,y=430,width=150,h
eight=40)



    def login_window(self):
```

```
        self.root.destroy()
        os.system("python login.py")


    def clear(self):
        self.txt_f_name.delete(0,END)
        self.txt_l_name.delete(0,END)
        self.txt_contact.delete(0,END)
        self.txt_email.delete(0,END)
        self.txt_password.delete(0,END)
        self.txt_cpassword.delete(0,END)
        self.txt_answer.delete(0,END)


        self.txt_cpassword.delete(0,END)
        self.cmb_quest.current(0)
          def register_data(self):
        if self.txt_f_name.get()=="" or self.txt_contact.get()==""or
self.txt_email.get()=="" or self.cmb_quest.get()=="Select" or
self.txt_answer.get()=="" or self.txt_password.get()=="" or
self.txt_cpassword.get()=="":
            messagebox.showerror("Error","All Fields are
required",parent=self.root)
        elif self.txt_password.get()!= self.txt_cpassword.get():
            messagebox.showerror("Error","Password & Confirm Password should
be same ",parent=self.root)
        elif self.var_chk.get()==0:
            messagebox.showerror("Error","Please Agree our terms &
condition",parent=self.root)


        else:
            try:
                con=sqlite3.connect(database="rms.db")
                cur=con.cursor()
                cur.execute("select * from employee where
email=?",(self.txt_email.get(),))
                row=cur.fetchone()
```

```
                if row!=None:
                    messagebox.showerror("Error","User already Exists,Please
try with another email",parent=self.root)
                else:
                    cur.execute("insert into
employee(f_name,l_name,contact,email,question,answer,password)
values(?,?,?,?,?,?,?)",
                                (self.txt_f_name.get(),
                                self.txt_l_name.get(),
                                self.txt_contact.get(),
                                self.txt_email.get(),
                                self.cmb_quest.get(),
                                self.txt_answer.get(),
                                self.txt_password.get()
                                ))
                con.commit()
                con.close()
                messagebox.showinfo("Success","Register
Successfull",parent=self.root)
                self.clear()
                self.login_window()
            except Exception as ex:
                messagebox.showerror("Error",f"Error due to:
{str(ex)}",parent=self.root)
root=Tk()
obj=Register(root)
root.mainloop()
```

Login Window Code:

```
from tkinter import*
from PIL import Image,ImageTk,ImageDraw #pip install pillow
from datetime import*
import time
from math import*
#import pymysql #pip install pymysql
import sqlite3
```

```
import os
from tkinter import messagebox,ttk

class Login_window:
    def __init__(self,root):
        self.root=root
        self.root.title("LOGIN SYSTEM")
        self.root.geometry("1520x780+0+0")
        self.root.config(bg="#021e2f")

        #============background colors=============================
        left_lbl=Label(self.root,bg="#08A3D2",bd=0)
        left_lbl.place(x=0,y=0,relheight=1,width=750)

        right_lbl=Label(self.root,bg="#031F3C",bd=0)
        right_lbl.place(x=750,y=0,relheight=1,relwidth=1)
       #=======frames===============

        login_frame=Frame(self.root,bg="white")
        login_frame.place(x=340,y=150,width=800,height=500)



        title=Label(login_frame,text="LOGIN HERE",font=("times new
roman",30,"bold"),bg="white",fg="#08A3D2").place(x=250,y=50)

        email=Label(login_frame,text="EMAIL ADDRESS",font=("times new
roman",16,"bold"),bg="white",fg="gray").place(x=250,y=150)
        self.txt_email=Entry(login_frame,font=("times new
roman",15,),bg="lightgray")
        self.txt_email.place(x=250,y=180,width=350,height=35)

        password=Label(login_frame,text="PASSWORD",font=("times new
roman",16,"bold"),bg="white",fg="gray").place(x=250,y=250)
        self.txt_password=Entry(login_frame,font=("times new
roman",15,),bg="lightgray")
        self.txt_password.place(x=250,y=280,width=350,height=35)
```

```
btn_reg=Button(login_frame,cursor="hand2",command=self.register_window,text="
Register new Account?",font=("times new
roman",14),bg="white",bd=0,fg="#b00857").place(x=250,y=320)


btn_forget=Button(login_frame,cursor="hand2",command=self.forget_password_win
dow,text="Forget Password?",font=("times new
roman",14),bg="white",bd=0,fg="red").place(x=450,y=320)


btn_login=Button(login_frame,text="Login",command=self.login,font=("times new
roman",20,"bold"),fg="white",bg="#b00857",cursor="hand2").place(x=250,y=380,w
idth=180,height=40)


#btn_login=Button(login_frame,text="Login(Student)",command=self.login2,font=
("times new
roman",20,"bold"),fg="white",bg="green",cursor="hand2").place(x=500,y=380,wid
th=180,height=40)
       #============clock==============
        self.lbl=Label(self.root,text="Login Window",font=("\nBook
Antiqua",25,"bold"),compound=BOTTOM,fg="white",bg="#081923",bd=0)
        self.lbl.place(x=180,y=175,height=450,width=350)
        #self.clock_image()
        self.working()


    def reset(self):
        self.cmb_quest.current(0)
        self.txt_new_password.delete(0,END)
        self.txt_answer.delete(0,END)
        self.txt_password.delete(0,END)
        self.txt_email.delete(0,END)


 def forget_password(self):
        if self.cmb_quest.get()=="Select" or self.txt_answer.get()=="" or
self.txt_new_password.get()=="":
            messagebox.showerror("Error","All fields are
required",parent=self.root2)
        else:
            try:
                con=sqlite3.connect(database="rms.db")
```

```
                cur=con.cursor()
                cur.execute("select * from employee where email=? and
question=? and
answer=?",(self.txt_email.get(),self.cmb_quest.get(),self.txt_answer.get()))

                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please select the correct
Security Question / Enter Answer",parent=self.root2)

                else:
                     cur.execute("update employee set password=? where
email=? ",(self.txt_new_password.get(),self.txt_email.get()))

                     con.commit()
                     con.close()
                     messagebox.showinfo("Success","Your Password has been
reset, Please login with new password.",parent=self.root2)

                     self.reset()
                     self.root2.destroy()

            except Exception as ex:
                 messagebox.showerror("Error",f"Error due to:
{str(ex)}",parent=self.root)


def forget_password_window(self):
        if self.txt_email.get()=="":
            messagebox.showerror("Error","Please enter the email address to
reset your password",parent=self.root)
        else:
            try:
                con=sqlite3.connect(database="rms.db")
                cur=con.cursor()
                cur.execute("select * from employee where
email=?",(self.txt_email.get(),))
```

```
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please enter the valid
email address to reset your password",parent=self.root)

                else:
                    con.close()
                    self.root2=Toplevel()
                    self.root2.title("Forget Password")
                    self.root2.geometry("350x400+495+150")
                    self.root2.config(bg="white")
                    self.root2.focus_force()
                    self.root2.grab_set()

                    t=Label(self.root2,text="Forget Password",font=("times
new roman",20,"bold"),bg="white",fg="red").place(x=0,y=10,relwidth=1)



                    #=================forget password=========
                    question=Label(self.root2,text="Security
Question",font=("times new
roman",15,'bold'),bg="white",fg="gray").place(x=50,y=100)

                    self.cmb_quest=ttk.Combobox(self.root2,font=("times new
roman",13),state='readonly',justify=CENTER)
                    self.cmb_quest['values']=("Select","Your First Pet
Name","Your Birth Place","Your Best Friend Name")
                    self.cmb_quest.place(x=50,y=130,width=250)
                    self.cmb_quest.current(0)

                    answer=Label(self.root2,text="Answer",font=("times new
roman",15,'bold'),bg="white",fg="gray").place(x=50,y=180)

                    self.txt_answer=Entry(self.root2,font=("times new
roman",15),bg="lightgray")
                    self.txt_answer.place(x=50,y=210,width=250)
```

```
                    new_password=Label(self.root2,text="New
Password",font=("times new
roman",15,'bold'),bg="white",fg="gray").place(x=50,y=260)


                    self.txt_new_password=Entry(self.root2,font=("times new
roman",15),bg="lightgray")
                    self.txt_new_password.place(x=50,y=290,width=250)



btn_change_password=Button(self.root2,command=self.forget_password,text="Rese
t Password",bg="green",fg="white",font=("times new
roman",15,"bold")).place(x=90,y=340)


            except Exception as ex:
                 messagebox.showerror("Error",f"Error due to:
{str(ex)}",parent=self.root)


    def register_window(self):
        self.root.destroy()
        import register


    def login(self):
        if self.txt_email.get()=="" or self.txt_password.get()=="":
            messagebox.showerror("Error","All fields are
required",parent=self.root)


        else:
            try:
                con=sqlite3.connect(database="rms.db")
                cur=con.cursor()
                cur.execute("select * from employee where email=? and
password=?",(self.txt_email.get(),self.txt_password.get()))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Username and
Password",parent=self.root)
```

```
                else:
                    messagebox.showinfo("Success",f"Welcome:
{self.txt_email.get()}",parent=self.root)
                    self.root.destroy()
                    os.system("python sample.py")
                    con.close()
            except Exception as ex:
                 messagebox.showerror("Error",f"Error due to:
{str(ex)}",parent=self.root)


    def clock_image(self,hr,min_,sec_):
        clock=Image.new("RGB",(400,400),(8,25,35))
        draw=ImageDraw.Draw(clock)
        #========for clock image==============
        bg=Image.open("image2/Frame.png")
        bg=bg.resize((300,300),Image.ANTIALIAS)
        clock.paste(bg,(50,50))



        #formula to rotate the anticlock
        #angle_in_radians = angle_in_degrees * math.pi / 180
        #line_length=100
        #center_x=250
        #center_y=250
        #end_x=center_x + line_length * math.cos(angle_in_radians)
        #end_y=center_y - line_length * math.sin(angle_in_radians)


        clock.save("clock_new.png")
        #=============hour line image==============
        #           x1,y1,x2,y2
        origin=200,200
        draw.line((origin,200+50*sin(radians(hr)),200-
50*cos(radians(hr))),fill="black",width=4)
        clock.save("clock_new.png")
```

```
        #=============minute line image===============
        draw.line((origin,200+80*sin(radians(min_)),200-
80*cos(radians(min_))),fill="green",width=3)
        clock.save("clock_new.png")
        #=============second line image===============
        draw.line((origin,200+100*sin(radians(sec_)),200-
100*cos(radians(sec_))),fill="red",width=4)
        draw.ellipse((194,194,205,205),fill="red")
        clock.save("clock_new.png")


    def working(self):
         h=datetime.now().time().hour
         m=datetime.now().time().minute
         s=datetime.now().time().second


         hr=(h/12)*360
         min_=(m/60)*360
         sec_=(s/60)*360
         #print(h,m,s)
         #print(hr,min_,sec_)
         self.clock_image(hr,min_,sec_)
         self.img=ImageTk.PhotoImage(file="clock_new.png")
         self.lbl.config(image=self.img)
         self.lbl.after(200,self.working)
root=Tk()
obj=Login_window(root)
root.mainloop()
```

Choose Window Code:

```
from tkinter import*
from PIL import Image,ImageTk,ImageDraw #pip install pillow
from datetime import*
import time
from math import*
```

```
#import pymysql #pip install pymysql
import sqlite3
import os
from tkinter import messagebox,ttk

class Choose_window:
    def __init__(self,root):
        self.root=root
        self.root.title("CHOOSE SYSTEM")
        self.root.geometry("1520x780+0+0")
        self.root.config(bg="#021e2f")

        #=============background colors==============================
        left_lbl=Label(self.root,bg="#08A3D2",bd=0)
        left_lbl.place(x=0,y=0,relheight=1,width=750)

        right_lbl=Label(self.root,bg="#031F3C",bd=0)
        right_lbl.place(x=750,y=0,relheight=1,relwidth=1)
       #=======frames===============

        choose_frame=Frame(self.root,bg="white")
        choose_frame.place(x=340,y=150,width=800,height=500)

        self.lbl=Label(self.root,text="choose Window",font=("\nBook
Antiqua",25,"bold"),compound=BOTTOM,fg="white",bg="#081923",bd=0)
        self.lbl.place(x=180,y=175,height=450,width=350)



        # Checkbutton for "Faculty"
        self.faculty_var = IntVar()
        faculty_checkbox = Checkbutton(
            choose_frame,
            text="Faculty",
            variable=self.faculty_var,
```

```
            font=("times new roman", 25),
            bg="white",
        )
        faculty_checkbox.place(x=400, y=180)

        # Checkbutton for "Student"
        self.student_var = IntVar()
        student_checkbox = Checkbutton(
            choose_frame,
            text="Student",
            variable=self.student_var,
            font=("times new roman", 25),
            bg="white",
        )
        student_checkbox.place(x=400, y=300)

        # Submit button
        submit_button = Button(
            choose_frame,
            text="Submit",
            font=("times new roman", 20),
            bg="white",
            command=self.submit_action,
        )
        submit_button.place(x=400, y=400)

        # Initialize selected value
        self.selected_value = None

        # Update selected value based on the checkbox selection
        self.faculty_var.trace("w", lambda *args:
self.update_selected_value("faculty"))
        self.student_var.trace("w", lambda *args:
self.update_selected_value("student"))
```

```
        #self.clock_image()
        self.working()

    def update_selected_value(self, value):
        self.selected_value = value

    def submit_action(self):
        if self.selected_value == "faculty":
            # Redirect to faculty dashboard section
           self.root.destroy()
           os.system("python dashboard.py")


        elif self.selected_value == "student":
            # Redirect to student report section
            self.root.destroy()
            os.system("python report.py")
        else:
            messagebox.showerror("Error", "Please select an option")  # Show
error message if no option is selected

    def clock_image(self,hr,min_,sec_):
        clock=Image.new("RGB",(400,400),(8,25,35))
        draw=ImageDraw.Draw(clock)
        #========for clock image===============
        bg=Image.open("image2/Frame.png")
        bg=bg.resize((300,300),Image.ANTIALIAS)
        clock.paste(bg,(50,50))


        #formula to rotate the anticlock
        #angle_in_radians = angle_in_degrees * math.pi / 180
        #line_length=100
```

```
        #center_x=250
        #center_y=250
        #end_x=center_x + line_length * math.cos(angle_in_radians)
        #end_y=center_y - line_length * math.sin(angle_in_radians)

        clock.save("clock_new.png")
        #=============hour line image==============
        #           x1,y1,x2,y2
        origin=200,200
        draw.line((origin,200+50*sin(radians(hr)),200-
50*cos(radians(hr))),fill="black",width=4)
        clock.save("clock_new.png")
        #=============minute line image==============
        draw.line((origin,200+80*sin(radians(min_)),200-
80*cos(radians(min_))),fill="green",width=3)
        clock.save("clock_new.png")
        #=============second line image==============
        draw.line((origin,200+100*sin(radians(sec_)),200-
100*cos(radians(sec_))),fill="red",width=4)
        draw.ellipse((194,194,205,205),fill="red")
        clock.save("clock_new.png")



    def working(self):
         h=datetime.now().time().hour
         m=datetime.now().time().minute
         s=datetime.now().time().second

         hr=(h/12)*360
         min_=(m/60)*360
         sec_=(s/60)*360
         #print(h,m,s)
         #print(hr,min_,sec_)
         self.clock_image(hr,min_,sec_)
         self.img=ImageTk.PhotoImage(file="clock_new.png")
```

```
            self.lbl.config(image=self.img)
            self.lbl.after(200,self.working)
root=Tk()
obj=Choose_window(root)
root.mainloop()
```

Dashboard Window Code:

```
from tkinter import*
from tkinter import messagebox
from PIL import Image,ImageTk #pip install pillow
from course import CourseClass
from student import studentClass
from result import resultClass
from report import reportClass
from tkinter import messagebox
import os

from PIL import Image,ImageTk,ImageDraw #pip install pillow
from datetime import*
import time
from math import*
#import pymysql #pip install pymysql
import sqlite3


class RMS:
    def __init__(self,root):
        self.root=root
        self.root.title("FitBuddy Android App")
        self.root.geometry("1520x780+0+0")
        self.root.config(bg="alice blue")
        #===icons===
```

```
        self.logo_dash=ImageTk.PhotoImage(file="image/analytics.png")


        #===title===


        title=Label(self.root,text="FitBuddy Android
App",compound=LEFT,padx=10,
              image=self.logo_dash,font=("goudy old
style",20,"bold"),bg="#033054",fg="white").place(x=0,y=0,relwidth=1,height=50
)


        #===menu====


        M_Frame=LabelFrame(self.root,text="Menus",font=("times new
roman",15),bg="white")
        M_Frame.place(x=10,y=70,width=1500,height=80)
        btn_course=Button(M_Frame,text="Course",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_cou
rse).place(x=20,y=5,width=200,height=40)
        btn_student=Button(M_Frame,text="Student",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_stu
dent).place(x=270,y=5,width=200,height=40)
        btn_result=Button(M_Frame,text="Result",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_res
ult).place(x=520,y=5,width=200,height=40)
        btn_view=Button(M_Frame,text="View Student Result",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_rep
ort).place(x=770,y=5,width=200,height=40)
        btn_logout=Button(M_Frame,text="Logout",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.logout)
.place(x=1020,y=5,width=200,height=40)
        btn_exit=Button(M_Frame,text="Exit",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.exit_).
place(x=1270,y=5,width=200,height=40)


        #===content_window====


        self.bg_img=Image.open("image/bg.png")
        self.bg_img=self.bg_img.resize((920,430),Image.ANTIALIAS)
        self.bg_img=ImageTk.PhotoImage(self.bg_img)
```

```
self.lbl_bg=Label(self.root,image=self.bg_img).place(x=590,y=160,width=920,he
ight=430)


        #===update_detail====


        self.lbl_course=Label(self.root,text="Total Courses\n[ 0
]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#e43b06",fg="white")
        self.lbl_course.place(x=590,y=600,width=300,height=100)
        self.lbl_student=Label(self.root,text="Total Students\n[ 0
]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#0676ad",fg="white")
        self.lbl_student.place(x=900,y=600,width=300,height=100)
        self.lbl_result=Label(self.root,text="Total Results\n[ 0
]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#038074",fg="white")
        self.lbl_result.place(x=1210,y=600,width=300,height=100)


        #============clock==============
        self.lbl=Label(self.root,text="Tic! Tic! Tic!",font=("\nBook
Antiqua",25,"bold"),compound=BOTTOM,fg="white",bg="#081923",bd=0)
        self.lbl.place(x=115,y=205,height=450,width=350)
        #self.clock_image()
        self.working()


         #===footer===


        footer=Label(self.root,text="FitBuddy Android App\nA Team Project:-
B.Tech CSE 4th Year\nBy : Nazeem Ahmad & Komal Kumari ",font=("goudy old
style",12),bg="#262626",fg="white").pack(side=BOTTOM,fill=X)


        self.update_details()
    #========================
    def update_details(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select * from course")
```

```
            cr=cur.fetchall()
            self.lbl_course.config(text=f"Total Courses\n[{str(len(cr))}]")

            cur.execute("select * from student")
            cr=cur.fetchall()
            self.lbl_student.config(text=f"Total Students\n[{str(len(cr))}]")

            cur.execute("select * from result")
            cr=cur.fetchall()
            self.lbl_result.config(text=f"Total Results\n[{str(len(cr))}]")

            self.lbl_course.after(200,self.update_details)

        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def working(self):
        h=datetime.now().time().hour
        m=datetime.now().time().minute
        s=datetime.now().time().second

        hr=(h/12)*360
        min_=(m/60)*360
        sec_=(s/60)*360
        #print(h,m,s)
        #print(hr,min_,sec_)
        self.clock_image(hr,min_,sec_)
        self.img=ImageTk.PhotoImage(file="clock_new.png")
        self.lbl.config(image=self.img)
        self.lbl.after(200,self.working)
    #================
    def clock_image(self,hr,min_,sec_):
        clock=Image.new("RGB",(400,400),(8,25,35))
        draw=ImageDraw.Draw(clock)
```

```
        #========for clock image===============
        bg=Image.open("image2/Frame.png")
        bg=bg.resize((300,300),Image.ANTIALIAS)
        clock.paste(bg,(50,50))

        #formula to rotate the anticlock
        #angle_in_radians = angle_in_degrees * math.pi / 180
        #line_length=100
        #center_x=250
        #center_y=250
        #end_x=center_x + line_length * math.cos(angle_in_radians)
        #end_y=center_y - line_length * math.sin(angle_in_radians)

        clock.save("clock_new.png")
        #=============hour line image==============
        #           x1,y1,x2,y2
        origin=200,200
        draw.line((origin,200+50*sin(radians(hr)),200-
50*cos(radians(hr))),fill="black",width=4)
        clock.save("clock_new.png")
        #=============minute line image==============
        draw.line((origin,200+80*sin(radians(min_)),200-
80*cos(radians(min_))),fill="green",width=3)
        clock.save("clock_new.png")
        #=============second line image==============
        draw.line((origin,200+100*sin(radians(sec_)),200-
100*cos(radians(sec_))),fill="red",width=4)
        draw.ellipse((194,194,205,205),fill="red")
        clock.save("clock_new.png")


def add_course(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=CourseClass(self.new_win)


    def add_student(self):
```

```
        self.new_win=Toplevel(self.root)
        self.new_obj=studentClass(self.new_win)


    def add_result(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=resultClass(self.new_win)

    def add_report(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=reportClass(self.new_win)



    def logout(self):
        op=messagebox.askyesno("Confirm","Do you really want to
logout?",parent=self.root)
        if op==True:
            self.root.destroy()
            os.system("python login.py")

    def exit_(self):
        op=messagebox.askyesno("Confirm","Do you really want to
Exit?",parent=self.root)
        if op==True:
            self.root.destroy()
if __name__=="__main__":
    root=Tk()
    obj=RMS(root)
    root.mainloop()
```

Course Window Code:

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
```

```
class CourseClass:
    def __init__(self,root):
        self.root=root
        self.root.title("FitBuddy Android App")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        #===title===
        title=Label(self.root,text=" Manage Course Details",font=("goudy old
style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=
35)
        #=======Variables========
        self.var_subject=StringVar()
        self.var_faculty=StringVar()
        self.var_year=StringVar()


        #======Widgets==========
        lbl_subjectName=Label(self.root,text="Subject Name",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=60)
        lbl_faculty=Label(self.root,text="Faculty",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=100)
        lbl_courseyear=Label(self.root,text="Course Year",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=140)
        lbl_description=Label(self.root,text="Description",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=180)


        #=======Entry Fields============


#self.txt_subjectName=Entry(self.root,textvariable=self.var_subject,font=("go
udy old style",15,'bold'),bg='lightyellow')


self.txt_courseName=ttk.Combobox(self.root,textvariable=self.var_subject,valu
es=("Select","Artificial Intelligence","Cloud Computing","Rural
Development","VFHS","Data Structure","DBMS","Operating System","Python","Web
Designing","Compiler Design","Theory of Automata","Machine
Learning","Constition of India", "Power BI", "Digital and Social Media
Marketing", "SQL"),font=("goudy old style" ,15,'bold'),justify=CENTER)
        self.txt_courseName.place(x=150,y=60,width=200)
```

```
        self.txt_courseName.current(0)

        #self.txt_subjectName.place(x=150,y=60,width=200)


txt_faculty=Entry(self.root,textvariable=self.var_faculty,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=100,width=200)


txt_courseyear=Entry(self.root,textvariable=self.var_year,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)

        self.txt_description=Text(self.root,font=("goudy old
style",15,'bold'),bg='lightyellow')

        self.txt_description.place(x=150,y=180,width=500,height=130)


        #========Buttons===========

        self.btn_add=Button(self.root,text='Save',font=("goudy old
style",15,"bold"),bg="#2196f3",fg="white",cursor="hand2",command=self.add)

        self.btn_add.place(x=150,y=400,width=110,height=40)

        self.btn_update=Button(self.root,text='Update',font=("goudy old
style",15,"bold"),bg="#4caf50",fg="white",cursor="hand2",command=self.update)

        self.btn_update.place(x=270,y=400,width=110,height=40)

        self.btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,"bold"),bg="#f44336",fg="white",cursor="hand2",command=self.delete)

        self.btn_delete.place(x=390,y=400,width=110,height=40)

        self.btn_clear=Button(self.root,text='Clear',font=("goudy old
style",15,"bold"),bg="#607d8b",fg="white",cursor="hand2",command=self.clear)

        self.btn_clear.place(x=510,y=400,width=110,height=40)

        #=========Search Panel========

        self.var_search=StringVar()

        lbl_self_subjectName=Label(self.root,text="Subject Name",font=("goudy
old style",15,'bold'),bg='lightyellow').place(x=720,y=60)


txt_search_subjectName=Entry(self.root,textvariable=self.var_search,font=("go
udy old style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)

        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search)
.place(x=1070,y=60,width=120,height=28)

        #========Content=========

        self.C_Frame=Frame(self.root,bd=2,relief=RIDGE)

        self.C_Frame.place(x=720,y=100,width=470,height=340)
```

```
        scrolly=Scrollbar(self.C_Frame,orient=VERTICAL)
        scrollx=Scrollbar(self.C_Frame,orient=HORIZONTAL)

self.CourseTable=ttk.Treeview(self.C_Frame,columns=("cid","name","faculty","y
ear","description"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)

        scrollx.pack(side=BOTTOM,fill=X)
        scrolly.pack(side=RIGHT,fill=Y)
        scrollx.config(command=self.CourseTable.xview)
        scrolly.config(command=self.CourseTable.yview)

        self.CourseTable.heading("cid",text="Course ID")
        self.CourseTable.heading("name",text="Name")
        self.CourseTable.heading("faculty",text="Faculty")
        self.CourseTable.heading("year",text="Year")
        self.CourseTable.heading("description",text="Description")
        self.CourseTable["show"]='headings'
        self.CourseTable.column("cid",width=100)
        self.CourseTable.column("name",width=100)
        self.CourseTable.column("faculty",width=100)
        self.CourseTable.column("year",width=100)
        self.CourseTable.column("description",width=150)
        self.CourseTable.pack(fill=BOTH,expand=1)
        self.CourseTable.bind("<ButtonRelease-1>",self.get_data)
        self.show()

#============================================================
    def clear(self):
        self.show()
        self.var_subject.set("")
        self.var_faculty.set("")
        self.var_year.set("")
        self.var_search.set("")
        self.txt_description.delete('1.0',END)
```

```
        self.txt_courseName.config(state=NORMAL)


    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_subject.get()=="":
                messagebox.showerror("Error","Subject Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where
name=?",(self.var_subject.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please select course from
the list first",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to
delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from course where
name=?",(self.var_subject.get(),))
                        con.commit()
                        messagebox.showinfo("Delete","Subject deleted
Successfully",parent=self.root)
                        self.clear()


        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")



    def get_data(self,ev):
        self.txt_courseName.config(state='readonly')
        r=self.CourseTable.focus()
        content=self.CourseTable.item(r)
        row=content["values"]
```

```
        #print(row)
        self.var_subject.set(row[1])
        self.var_faculty.set(row[2])
        self.var_year.set(row[3])
        #self.var_.set(row[4])
        self.txt_description.delete('1.0',END)
        self.txt_description.insert(END,row[4])

    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_subject.get()=="":
                messagebox.showerror("Error","Subject Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where
name=?",(self.var_subject.get(),))
                row=cur.fetchone()
                if row!=None:
                    messagebox.showerror("Error","Subject Name already
present",parent=self.root)
                else:
                    cur.execute("insert into
course(name,faculty,year,description) values(?,?,?,?)",(
                        self.var_subject.get(),
                        self.var_faculty.get(),
                        self.var_year.get(),
                        self.txt_description.get("1.0",END)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Course Added
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")
```

```
    def update(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_subject.get()=="":
                messagebox.showerror("Error","Subject Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where
name=?",(self.var_subject.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Select Subject from
list",parent=self.root)
                else:
                    cur.execute("update course set
faculty=?,year=?,description=? where name=?",(
                        self.var_faculty.get(),
                        self.var_year.get(),
                        self.txt_description.get("1.0",END),
                        self.var_subject.get()
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Course Update
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")




    def show(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
```

```
        try:
            cur.execute("select * from course")
            rows=cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute(f"select * from course where name LIKE
'%{self.var_search.get()}%'")
            rows=cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


if __name__=="__main__":
    root=Tk()
    obj= CourseClass(root)
    root.mainloop()
```

Student Window Code:

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
class_studentClass:
    def __init__(self,root):
```

```
        self.root=root
        self.root.title("FitBuddy Android App")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        #===title===
        title=Label(self.root,text=" Manage Student Details",font=("goudy old
style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=
35)
        #=======Variables========
        self.var_roll=StringVar()
        self.var_name=StringVar()
        self.var_email=StringVar()
        self.var_gender=StringVar()
        self.var_dob=StringVar()
        self.var_contact=StringVar()
        self.var_course=StringVar()
        self.var_a_date=StringVar()
        self.var_state=StringVar()
        self.var_city=StringVar()
        self.var_pin=StringVar()


        #======Widgets==========
        #=========column1========
        lbl_roll=Label(self.root,text="Roll No.",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=60)
        lbl_Name=Label(self.root,text="Name",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=100)
        lbl_Email=Label(self.root,text="Email",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=140)
        lbl_gender=Label(self.root,text="Gender",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=180)


        lbl_state=Label(self.root,text="State",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=220)
        txt_state=Entry(self.root,textvariable=self.var_state,font=("goudy
```

```
old style",15,'bold'),bg='lightyellow').place(x=150,y=220,width=150)


        lbl_city=Label(self.root,text="City",font=("goudy old
style",15,'bold'),bg='white').place(x=310,y=220)

        txt_city=Entry(self.root,textvariable=self.var_city,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=380,y=220,width=100)


        lbl_pin=Label(self.root,text="Pin",font=("goudy old
style",15,'bold'),bg='white').place(x=500,y=220)

        txt_pin=Entry(self.root,textvariable=self.var_pin,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=560,y=220,width=120)


        lbl_address=Label(self.root,text="Address",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=260)


        #=======Entry Fields============

        self.txt_roll=Entry(self.root,textvariable=self.var_roll,font=("goudy
old style",15,'bold'),bg='lightyellow')

        self.txt_roll.place(x=150,y=60,width=200)

        txt_name=Entry(self.root,textvariable=self.var_name,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=100,width=200)

        txt_email=Entry(self.root,textvariable=self.var_email,font=("goudy
old style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)


self.txt_gender=ttk.Combobox(self.root,textvariable=self.var_gender,values=("
Select","Male","Female","Other"),font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)

        self.txt_gender.place(x=150,y=180,width=200)

        self.txt_gender.current(0)


         #============column2============

        lbl_dob=Label(self.root,text="D.O.B",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=60)

        lbl_contact=Label(self.root,text="Contact",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=100)

        lbl_addmission=Label(self.root,text="Admission",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=140)

        lbl_course=Label(self.root,text="Course",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=180)
```

```
        #=======Entry Fields============

        self.course_list=[]

        #function_call to update the list

        self.fetch_course()

        txt_dob=Entry(self.root,textvariable=self.var_dob,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=60,width=200)


txt_contact=Entry(self.root,textvariable=self.var_contact,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=100,width=200)


txt_addmission=Entry(self.root,textvariable=self.var_a_date,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=140,width=200)


self.txt_course=ttk.Combobox(self.root,textvariable=self.var_course,values=se
lf.course_list,font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)

        self.txt_course.place(x=480,y=180,width=200)

        self.txt_course.set("Select")


        #===========Text Address============


        self.txt_address=Text(self.root,font=("goudy old
style",15,'bold'),bg='lightyellow')

        self.txt_address.place(x=150,y=260,width=540,height=100)


        #========Buttons===========

        self.btn_add=Button(self.root,text='Save',font=("goudy old
style",15,"bold"),bg="#2196f3",fg="white",cursor="hand2",command=self.add)

        self.btn_add.place(x=150,y=400,width=110,height=40)

        self.btn_update=Button(self.root,text='Update',font=("goudy old
style",15,"bold"),bg="#4caf50",fg="white",cursor="hand2",command=self.update)

        self.btn_update.place(x=270,y=400,width=110,height=40)

        self.btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,"bold"),bg="#f44336",fg="white",cursor="hand2",command=self.delete)

        self.btn_delete.place(x=390,y=400,width=110,height=40)

        self.btn_clear=Button(self.root,text='Clear',font=("goudy old
style",15,"bold"),bg="#607d8b",fg="white",cursor="hand2",command=self.clear)

        self.btn_clear.place(x=510,y=400,width=110,height=40)
```

```
        #=========Search Panel========

        self.var_search=StringVar()

        lbl_self_roll=Label(self.root,text="Roll No.",font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=720,y=60)


txt_search_roll=Entry(self.root,textvariable=self.var_search,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)

        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search)
.place(x=1070,y=60,width=120,height=28)

        #========Content=========

        self.C_Frame=Frame(self.root,bd=2,relief=RIDGE)

        self.C_Frame.place(x=720,y=100,width=470,height=340)


        scrolly=Scrollbar(self.C_Frame,orient=VERTICAL)

        scrollx=Scrollbar(self.C_Frame,orient=HORIZONTAL)


self.CourseTable=ttk.Treeview(self.C_Frame,columns=("roll","name","email","ge
nder","dob","contact","admission","course","state","city","pin","address"),xs
crollcommand=scrollx.set,yscrollcommand=scrolly.set)


        scrollx.pack(side=BOTTOM,fill=X)

        scrolly.pack(side=RIGHT,fill=Y)

        scrollx.config(command=self.CourseTable.xview)

        scrolly.config(command=self.CourseTable.yview)


        self.CourseTable.heading("roll",text="Roll No.")

        self.CourseTable.heading("name",text="Name")

        self.CourseTable.heading("email",text="Email")

        self.CourseTable.heading("gender",text="Gender")

        self.CourseTable.heading("dob",text="D.O.B")

        self.CourseTable.heading("contact",text="Contact")

        self.CourseTable.heading("admission",text="Admission")

        self.CourseTable.heading("course",text="Course")

        self.CourseTable.heading("state",text="State")

        self.CourseTable.heading("city",text="City")
```

```
        self.CourseTable.heading("pin",text="PIN")
        self.CourseTable.heading("address",text="Address")
        self.CourseTable["show"]='headings'
        self.CourseTable.column("roll",width=100)
        self.CourseTable.column("name",width=100)
        self.CourseTable.column("email",width=100)
        self.CourseTable.column("gender",width=100)
        self.CourseTable.column("dob",width=100)
        self.CourseTable.column("contact",width=100)
        self.CourseTable.column("admission",width=100)
        self.CourseTable.column("course",width=100)
        self.CourseTable.column("state",width=100)
        self.CourseTable.column("city",width=100)
        self.CourseTable.column("pin",width=100)
        self.CourseTable.column("address",width=200)
        self.CourseTable.pack(fill=BOTH,expand=1)
        self.CourseTable.bind("<ButtonRelease-1>",self.get_data)
        self.show()
    #=======================================================
    def clear(self):
        self.show()
        self.var_roll.set("")
        self.var_name.set("")
        self.var_email.set("")
        self.var_gender.set("Select")
        self.var_dob.set("")
        self.var_contact.set("")
        self.var_a_date.set("")
        self.var_course.set("Select")
        self.var_state.set("")
        self.var_city.set("")
        self.var_pin.set("")
        self.txt_address.delete("1.0",END)
        self.txt_roll.config(state=NORMAL)
```

```
        self.var_search.set("")


    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from student where
roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please select student from
the list first",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to
delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from student where
roll=?",(self.var_roll.get(),))
                        con.commit()
                        messagebox.showinfo("Delete","Student deleted
Successfully",parent=self.root)
                        self.clear()


        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def get_data(self,ev):
        self.txt_roll.config(state='readonly')
        r=self.CourseTable.focus()
        content=self.CourseTable.item(r)
        row=content["values"]
        self.var_roll.set(row[0])
```

```
        self.var_name.set(row[1])
        self.var_email.set(row[2])
        self.var_gender.set(row[3])
        self.var_dob.set(row[4])
        self.var_contact.set(row[5])
        self.var_a_date.set(row[6])
        self.var_course.set(row[7])
        self.var_state.set(row[8])
        self.var_city.set(row[9])
        self.var_pin.set(row[10])
        self.txt_address.delete("1.0",END)
        self.txt_address.insert(END,row[11])


    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from student where
roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                if row!=None:
                    messagebox.showerror("Error","Roll No. already
present",parent=self.root)
                else:
                    cur.execute("insert into
student(roll,name,email,gender,dob,contact,admission,course,state,city,pin,ad
dress) values(?,?,?,?,?,?,?,?,?,?,?,?)",(
                        self.var_roll.get(),
                        self.var_name.get(),
                        self.var_email.get(),
                        self.var_gender.get(),
                        self.var_dob.get(),
```

```
                        self.var_contact.get(),
                        self.var_a_date.get(),
                        self.var_course.get(),
                        self.var_state.get(),
                        self.var_city.get(),
                        self.var_pin.get(),
                        self.txt_address.get("1.0",END)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Student Added
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")
    def update(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from student where
roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Select student from
list",parent=self.root)
                else:
                    cur.execute("update student set
name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,city=?,p
in=?,address=? where roll=?",(
                        self.var_name.get(),
                        self.var_email.get(),
                        self.var_gender.get(),
                        self.var_dob.get(),
```

```
                        self.var_contact.get(),
                        self.var_a_date.get(),
                        self.var_course.get(),
                        self.var_state.get(),
                        self.var_city.get(),
                        self.var_pin.get(),
                        self.txt_address.get("1.0",END),
                        self.var_roll.get()
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Student Update
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def show(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select * from student")
            rows=cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('',END,values=row)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def fetch_course(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
```

```
            cur.execute("select name from course")
            rows=cur.fetchall()
            if len(rows)>0:
                for row in rows:
                    self.course_list.append(row[0])
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select * from student where
roll=?",(self.var_search.get(),))
            row=cur.fetchone()
            if row!=None:
                self.CourseTable.delete(*self.CourseTable.get_children())
                self.CourseTable.insert('',END,values=row)
            else:
                messagebox.showerror("Error","No record
found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


if __name__=="__main__":
    root=Tk()
    obj= studentClass(root)
    root.mainloop()
```

Result Window Code:

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
class resultClass:
```

```
    def __init__(self,root):
        self.root=root
        self.root.title("FitBuddy Android App")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        #===title===
        title=Label(self.root,text=" Add Student Results ",font=("goudy old
style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height
=50)
        #========widgets==========
        #==========variables=======
        self.var_roll=StringVar()
        self.var_name=StringVar()
        self.var_course=StringVar()
        self.var_marks=StringVar()
        self.var_full_marks=StringVar()
        self.roll_list=[]
        self.fetch_roll()


        lbl_select=Label(self.root,text="Select Student",font=("goudy old
style",20,'bold'),bg='white').place(x=50,y=100)
        lbl_name=Label(self.root,text="Name",font=("goudy old
style",20,'bold'),bg='white').place(x=50,y=160)
        lbl_course=Label(self.root,text="Course ",font=("goudy old
style",20,'bold'),bg='white').place(x=50,y=220)
        lbl_marks=Label(self.root,text="Marks Obtained",font=("goudy old
style",20,'bold'),bg='white').place(x=50,y=280)
        lbl_full_marks=Label(self.root,text="Full Marks",font=("goudy old
style",20,'bold'),bg='white').place(x=50,y=340)



self.txt_student=ttk.Combobox(self.root,textvariable=self.var_roll,values=sel
f.roll_list,font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)
        self.txt_student.place(x=280,y=100,width=200)
        self.txt_student.set("Select")
```

```
        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search)
.place(x=500,y=100,width=100,height=28)

        self.course_list=[]

        #function_call to update the list

        self.fetch_course()

        txt_name=Entry(self.root,textvariable=self.var_name,font=("goudy old
style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=160,width=
340)


self.txt_course=ttk.Combobox(self.root,textvariable=self.var_course,values=(s
elf.course_list),font=("goudy old style"
,15,'bold'),state='readonly',justify=CENTER)

        self.txt_course.place(x=280,y=225,width=210)

        self.txt_course.current(0)


        txt_marks=Entry(self.root,textvariable=self.var_marks,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=280,y=280,width=320)


txt_full_marks=Entry(self.root,textvariable=self.var_full_marks,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=280,y=340,width=320)


        #==========button=========

        btn_add=Button(self.root,text='Submit',font=("times new
roman",15),bg="lightgreen",activebackground="lightgreen",cursor="hand2",comma
nd=self.add).place(x=300,y=420,width=120,height=35)

        btn_clear=Button(self.root,text='Clear',font=("times new
roman",15),bg="lightgray",activebackground="lightgray",cursor="hand2",command
=self.clear).place(x=430,y=420,width=120,height=35)


        #========image========

        self.stu_img=Image.open("image/RESULT.png")

        self.stu_img=self.stu_img.resize((500,300),Image.ANTIALIAS)

        self.stu_img=ImageTk.PhotoImage(self.stu_img)


        self.lbl_stu=Label(self.root,image=self.stu_img).place(x=650,y=100)


        #================================
```

```
    def fetch_roll(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select roll from student")
            rows=cur.fetchall()
            if len(rows)>0:
                for row in rows:
                    self.roll_list.append(row[0])
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select name,course from student where
roll=?",(self.var_roll.get(),))
            row=cur.fetchone()
            if row!=None:
                self.var_name.set(row[0])
                self.var_course.set(row[1])
            else:
                messagebox.showerror("Error","No record
found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_name.get()=="":
                messagebox.showerror("Error","Please first search student
record",parent=self.root)
```

```
            else:
                cur.execute("select * from result where roll=? and
course=?",(self.var_roll.get(),self.var_course.get()))
                row=cur.fetchone()
                if row!=None:
                    messagebox.showerror("Error","Result already
present",parent=self.root)
                else:

per=(int(self.var_marks.get())*100)/int(self.var_full_marks.get())
                    cur.execute("insert into
result(roll,name,course,marks_ob,full_marks,per) values(?,?,?,?,?,?)",(
                        self.var_roll.get(),
                        self.var_name.get(),
                        self.var_course.get(),
                        self.var_marks.get(),
                        self.var_full_marks.get(),
                        str(per)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Result Added
Successfully",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")



    def clear(self):
        self.var_roll.set("Select")
        self.var_name.set("")
        self.var_course.set("")
        self.var_marks.set("")
        self.var_full_marks.set("")

    def fetch_course(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
```

```
        try:
            cur.execute("select name from course")
            rows=cur.fetchall()

            if len(rows)>0:
                for row in rows:
                    self.course_list.append(row[0])

            #print(v)


        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


if __name__=="__main__":
    root=Tk()
    obj= resultClass(root)
    root.mainloop()
```

View Report Window Code:

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
class reportClass:
    def __init__(self,root):
        self.root=root
        self.root.title("FitBuddy Android App")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        #===title===
        title=Label(self.root,text=" View Student Results ",font=("goudy old
style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height
```

```
=50)


        #========Search=======

        self.var_search=StringVar()

        self.var_id=""

        lbl_search=Label(self.root,text="Search By Roll No.",font=("goudy old
style",20,'bold'),bg='white').place(x=280,y=100)

        txt_search=Entry(self.root,textvariable=self.var_search,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=520,y=100,width=150)

        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search)
.place(x=680,y=100,width=100,height=35)

        btn_clear=Button(self.root,text='Clear',font=("goudy old
style",15,"bold"),bg="Gray",fg="white",cursor="hand2",command=self.clear).pla
ce(x=800,y=100,width=100,height=35)



        #===========result_labels

        lbl_roll=Label(self.root,text="Roll No.",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=150,y=230,width=150,
height=50)

        lbl_name=Label(self.root,text="Name",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=300,y=230,width=150,
height=50)

        lbl_course=Label(self.root,text="Course ",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=450,y=230,width=150,
height=50)

        lbl_marks=Label(self.root,text="Marks Obtained",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=600,y=230,width=150,
height=50)

        lbl_full=Label(self.root,text="Total Marks",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=750,y=230,width=150,
height=50)

        lbl_per=Label(self.root,text="Percentage",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=900,y=230,width=150,
height=50)


        self.roll=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)

        self.roll.place(x=150,y=280,width=150,height=50)
```

```
        self.name=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.name.place(x=300,y=280,width=150,height=50)
        self.course=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.course.place(x=450,y=280,width=150,height=50)
        self.marks=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.marks.place(x=600,y=280,width=150,height=50)
        self.full=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.full.place(x=750,y=280,width=150,height=50)
        self.per=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.per.place(x=900,y=280,width=150,height=50)



        #======button delete====
        btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,"bold"),bg="red",fg="white",cursor="hand2",command=self.delete).pla
ce(x=500,y=350,width=150,height=35)
       #========================================
    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_search.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from result where
roll=?",(self.var_search.get(),))
                row=cur.fetchone()
                if row!=None:
                    self.var_id=row[0]
                    self.roll.config(text=row[1])
                    self.name.config(text=row[2])
```

```
                    self.course.config(text=row[3])
                    self.marks.config(text=row[4])
                    self.full.config(text=row[5])
                    self.per.config(text=row[6])
                else:
                    messagebox.showerror("Error","No record
found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def clear(self):
        self.var_id=""
        self.roll.config(text="")
        self.name.config(text="")
        self.course.config(text="")
        self.marks.config(text="")
        self.full.config(text="")
        self.per.config(text="")
        self.var_search.set("")


    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_id=="":
                messagebox.showerror("Error","Search Student result
first",parent=self.root)
            else:
                cur.execute("select * from result where
rid=?",(self.var_id,))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Student
```

```
Result",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to
delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from result where
rid=?",(self.var_id,))
                        con.commit()
                        messagebox.showinfo("Delete","Result deleted
Successfully",parent=self.root)
                        self.clear()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")



if __name__=="__main__":
    root=Tk()
    obj= reportClass(root)
    root.mainloop()
```

SQlite3 Code:

```
import sqlite3
from openpyxl import Workbook

def create_db():
    con = sqlite3.connect(database="rms.db")
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS course(cid INTEGER PRIMARY KEY
AUTOINCREMENT,name text,faculty text,year text,description text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS student(roll INTEGER PRIMARY KEY
AUTOINCREMENT,name text,email text,gender text,dob text,contact text,admission
text,course text,state text,city text,pin text,address text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS result(rid INTEGER PRIMARY KEY
AUTOINCREMENT,roll text,name text,course text,marks_ob text,full_marks
text,per text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS employee(eid INTEGER PRIMARY KEY
AUTOINCREMENT,f_name text,l_name text,contact text,email text,question
text,answer text,password text)")
    con.commit()
```

```
    # Retrieve data from the tables
    cur.execute("SELECT * FROM course")
    course_data = cur.fetchall()

    cur.execute("SELECT * FROM student")
    student_data = cur.fetchall()

    cur.execute("SELECT * FROM result")
    result_data = cur.fetchall()

    cur.execute("SELECT * FROM employee")
    employee_data = cur.fetchall()

    con.close()

    # Write data to Excel file
    wb = Workbook()
    course_sheet = wb.active
    course_sheet.title = "Course Data"
    course_sheet.append(("Course ID", "Name", "Faculty", "Year",
"Description"))
    for row in course_data:
        course_sheet.append(row)

    student_sheet = wb.create_sheet(title="Student Data")
    student_sheet.append(("Roll", "Name", "Email", "Gender", "DOB", "Contact",
"Admission Date", "Course", "State", "City", "PIN", "Address"))
    for row in student_data:
        student_sheet.append(row)

    result_sheet = wb.create_sheet(title="Result Data")
    result_sheet.append(("Result ID", "Roll", "Name", "Course", "Marks
Obtained", "Full Marks", "Percentage"))
    for row in result_data:
        result_sheet.append(row)

    employee_sheet = wb.create_sheet(title="Employee Data")
    employee_sheet.append(("Employee ID", "First Name", "Last Name",
"Contact", "Email", "Question", "Answer", "Password"))
    for row in employee_data:
        employee_sheet.append(row)

    wb.save("rms_data.xlsx")

create_db()
```

Chapter 6 – Testing

Testing is vital for the success of any software. No system design is ever perfect. Testing is carried out in two phases. The first phase is during software engineering, specifically during module creation. The second phase occurs after the completion of the software, and it is called system testing, which verifies that the whole set of programs hang together.

White Box Testing:

In this technique, a close examination is conducted on the logical parts of the software by testing them with cases that exercise specific sets of conditions or loops. All logical parts of the software are checked once. This technique can identify errors such as typographical errors, instances where logical expressions should be executed once but are executed more than once, and errors resulting from the use of incorrect controls and loops. When box testing is performed, all independent parts within a module are tested, logical decisions on their true and false sides are exercised, all loops and bounds within their operational limits are exercised, and internal data structures are tested to ensure their validity.

Black Box Testing:

This method allows software engineers to devise sets of input techniques that thoroughly test all functional requirements of a program. Black Box Testing examines the input, output, and external data. It verifies whether the input data is correct and whether the desired output is obtained.

Unit Testing:

Each module is considered independently in white box testing. It focuses on each unit of the software as implemented in the source code.

Integration Testing:

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. Modules are integrated by using the top down approach.

Validation Testing:

Validation testing is conducted to ensure that all functional and performance requirements are met.

Chapter 7 – Project Monitoring and Estimation

PERT Chart:

A PERT Chart is organized for events, activities, or tasks and serves as a scheduling device that visually depicts the order of tasks to be performed. It allows for the calculation of the critical path, which involves determining the time and cost associated with each path. The critical path is the path that requires the greatest amount of elapsed time.

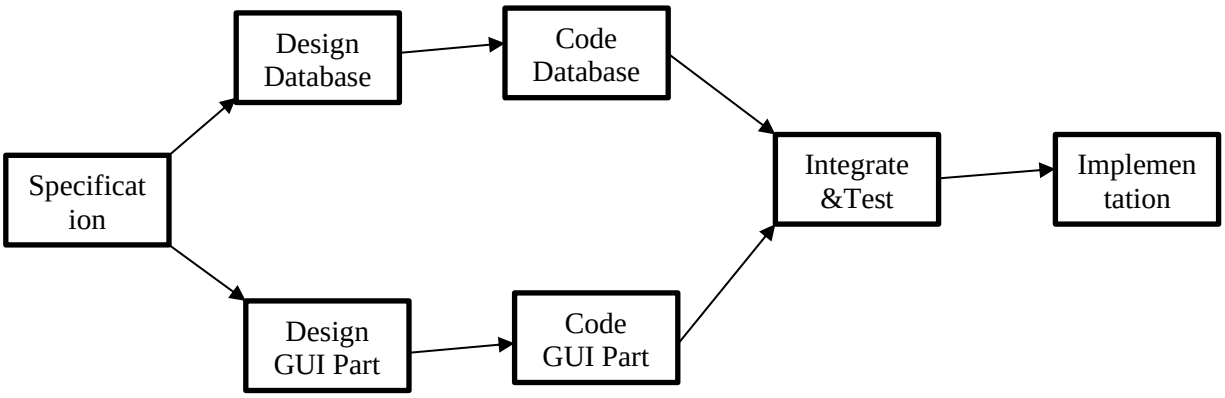

Fig. (7.1) PERT Chart Representation

GANNT Chart:

A Gantt chart, also known as a bar chart, is a project control technique primarily used for scheduling purposes. It is utilized for scheduling, budgeting, and resource planning. In a Gantt chart, each activity is represented by a bar, which is drawn against a timeline. The length of time allocated for each activity is depicted in the chart.

SDLC Activities	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY
Planning								
Analysis								
Design								
Coding								
Testing								
Implementation								

Fig. (7.2) GANNT Chart Representation

Cost Estimation of the Project:

Estimating the cost of a project involves various factors such as the scope, complexity, team size, development time, hourly rates, and other resources required. While I cannot provide you with an exact cost estimation, I can give you a general breakdown of the factors you need to consider when estimating the cost of developing a FitBuddy Android App using Python and SQLite3.

Project Scope:

Determine the specific features and functionalities required for the system, such as student registration, result recording, grade calculation, report generation, etc. The complexity and scale of these features will impact the development effort and cost.

Development Time:

Break down the development process into phases or tasks and estimate the time required for each. Consider activities like database design, user interface development, backend implementation, testing, and deployment. The more detailed and comprehensive your project plan, the more accurate your time estimation will be.

Team Size:

Assess the number of developers and their skill levels needed for the project. A larger team may reduce the development time, but it will also increase the cost. Consider roles like project manager, frontend developers, backend developers, database designers, and testers.

Hourly Rates:

Determine the hourly rates for each team member involved in the project. Rates can vary based on experience and location. Multiply the hourly rate by the estimated development time for each team member to calculate their cost contribution.

Third-Party Services:

If you plan to use any external services or APIs, consider their costs. For example, if you want to send SMS notifications to students or integrate with a payment gateway, you may incur additional expenses.

Infrastructure and Tools:

Take into account the cost of any infrastructure requirements such as hosting, domain registration, and server maintenance. Also, consider the cost of development tools, licenses, and libraries that you plan to use.

Contingency:

It's a good practice to include a contingency budget to accommodate any unforeseen delays or changes during the development process. A common practice is to allocate around 10-20% of the estimated cost as a contingency.

Once you have estimates for these factors, you can sum them up to get an overall cost estimation for your FitBuddy Android App project. Keep in mind that this is a high-level approach, and the actual cost may vary depending on your specific project requirements and circumstances.

Chapter 8 – Conclusion and Future Work

Conclusion:

- The FitBuddy Android App is a website that allows students and faculty members to check their academic results from anywhere at any time.
- It simplifies the process of calculating and displaying results, making it easy for users to understand their academic progress.
- Our project uses coding that is easy to use, making it accessible to users with different levels of technical knowledge.
- The system is designed to meet the needs of organizations looking to manage their projects more efficiently.
- The software planning process provides a framework for managers to make estimates and updates throughout the project, ensuring that it is completed on time, within budget, and to the required quality standards.

Future Scope:

- Add new features to the system
- Improve the user interface
- Enhance the system's security
- Optimize the system's performance
- Integrate the system with other educational software and systems, such as Learning Management Systems (LMS), Student Information Systems (SIS), and Human Resource Management Systems (HRMS)
- Explore commercialization opportunities by licensing or selling the system to educational institutions or other organizations.
- Contribute to the open-source community by releasing the system as an open-source software and allowing other developers to use, modify, and improve it.

References

- Available online at https://en.wikipedia.org/wiki/Student_information_system.

- Available online at https://www.youtube.com/Web_tech.
- Available online at https://www.python.org/.
- Available online at https://www.w3schools.com/.
- Kamthane, A. N., & Kamthane, A. A. (2017). Programming and Problem Solving with Python. Pearson Education India.
- Walia, E. S., & Gill, E. S. K. (2014). A framework for web-based FitBuddy Android App using Python. International Journal of Computer Science and Mobile Computing, 3(8), 24-33.
- Liu, Z., Wang, H., & Zan, H. (2010, October). Design and implementation of FitBuddy Android App. In 2010 International symposium on intelligence information processing and trusted computing (pp. 607-610). IEEE.

Publications

We are thrilled to announce the publication of our review paper titled "**Student Result Management Systems: A Comprehensive Review Paper**" in the prestigious International Research Journal of Modernization in Engineering Technology and Science (IRJMETS). This paper, authored by **Nazeem Ahmad and Komal Kumari**, delves into a comprehensive analysis of student result management systems, exploring their features, functionalities, and implementation strategies. Through an in-depth examination, the paper identifies key challenges in existing systems and proposes innovative solutions to address them. We express our sincere gratitude to Nazeem Ahmad and Komal Kumari for their diligent research and scholarly contributions, which significantly enriched the quality and depth of our publication. Additionally, we are pleased to share that our paper has been assigned an e-ISSN number (2582-5208) and Impact Factor (70868), further attesting to its credibility and recognition in the academic community. We believe that this publication will serve as a valuable resource for researchers, practitioners, and educators, providing insights and guidance for the advancement of student result management systems in the modern educational landscape.