**CHAPTER 1: INTRODUCTION**

**1.1 INTRODUCTION OF STUDENT REGISTRATION SYSTEM**

The Student Registering System in Python is a fully functional desktop application coded in a Python programming language. The project contains all the needed function for registering student record. This project was made so that it could make it easier to register student record. This will eventually help you manage all the student records.

A student registration system in Python is a software application designed to manage the process of registering students for classes at a school or university. The system may also include administrative functionality for staff to manage student records, course offerings, and schedules.

A student registration system typically includes a database to store student and course data. The database might include information such as student names, contact information, father name, and mother name. The system uses this data to manage the registration process, ensuring that students do not register for conflicting courses or courses they are not eligible for.

Python is a popular programming language for building student registration systems due to its simplicity, ease of use, and wide range of libraries and frameworks. Python frameworks such as Tkinter or Django can be used to build the web interface, while database management systems such as Excel or SQLite can be used to store student and course data.

Overall, a student registration system in Python provides an efficient and effective way to manage the complex process of registering students for classes, ensuring that courses are scheduled correctly and students are able to enroll in the classes they need.

**1.2 Problem Definition**

Here are some common problems that may arise in a student registration system in Python:

* **System performance**: As the number of students and courses increases, the system may experience performance issues such as slow page load times and database queries taking too long to execute.
* **Data validation:** The registration system must ensure that data entered by users is valid and meets the required format. If data validation is not properly implemented, it can lead to errors and incorrect data being stored in the system.
* **Security vulnerabilities:** The registration system must protect sensitive information such as student records and personal data from unauthorized access. Failure to implement proper security measures can lead to data breaches and compromise the integrity of the system.
* **User interface:** The user interface of the registration system must be user-friendly and intuitive. Poorly designed user interfaces can lead to confusion and errors in data entry.
* **Availability:** The registration system must be available for use during registration periods. Any unexpected downtime or system failures can cause delays and frustration for students.
* **System maintenance:** The registration system must be regularly maintained and updated to ensure that it remains functional and secure. Failure to perform regular maintenance can lead to system failures and security vulnerabilities.

Overall, these are some of the common problems that can arise in a student registration system in Python. These issues can be mitigated by proper planning, testing, and implementation of the system.

**1.3 OBJECTIVE**

The main objective of our project is that we can easily register new student

* Other objectives are:
* Simplify the registration process
* Provide administrative support
* Ensure eligibility requirements

By achieving these objectives, a student registration system in Python can help schools and universities manage the complex process of course registration more effectively, ensuring that students are able to register for the courses they need and that staff members can manage the process more efficiently.

## 1.4 BACKGROUND

The current system is a manual operated system where new student are registered manually in a new student register where admission number student name and other details are recorded. After the student is assigned his/her class another record is written down. Also the person in charge of admission has to count the number of student manually. All this work become more tedious and also wastage of time and also high cost of operation The person who will be in charge of the system will be able to log into the system and able to register new students and also be able to track student information. The school uses manual system in the process of administration and all its data is stored on paper and maintained in files. The administration of this school collects information from the student: student personal details like their names, admission, county of origin, medical details and parent’s details are also kept e.g. name, phone number, address and marital status. Before admission of the student, he/she must have paid 50% of school fee.

## 1.5 CURRENT SYSTEM

The manual system used in running School involves manual data entry to the databases. Incase of a transaction the registrar enters data and perform the necessary computations required . The registrar manually generates reports by gathering the required data from the database file and then typing them.

Secondly all these tasks are supposed to be done in the system manually for example;-

* Information in the school is manually collected when a student or a teacher join the school.
* Information from every class is also collected manually for example form 1 and form 4 pay different amount of money than form 2 and 3.
* Once a student makes payment, the accountant records the student in an accounts book stating the balance remaining and later keeps the book in a shelve or cabinets.
* Every student is supposed to fill personal details on a registration sheet and then the secretary files those records.
* Also the school library is manual whereby if one wants to borrow a book, he/she will have to pick the book from the shelves and then fill in a book borrowing from both personal details as well as details of the book. The form is then given to a librarian to manually check whether the details are correct. For the student’s details the student is required to give admission number and for the book details the librarian checks the filled in details against the book itself.
* The timetable is prepared manually whereby each lesson takes 45 minutes and the lesson is allocated to one teacher.

**1.6 SCOPE**

The scope of a student registration system in Python is broad and can cover many aspects of the student registration process.

Without an Online Student Registration System, managing and maintaining the details of the student is a tedious job for any organization.

Student registration system will store all the details of the students including their background information, educational qualifications, personal details and all the information related to their resume.

* **Student profiles:** The system should maintain profiles for each student, including their personal information, academic history, and current course schedule.
* **Administrative functionality:** The system may include administrative functionality for staff members, such as managing course offerings, viewing course rosters, and managing student records.

**1.7 LIMITATION**

Like any software system, a student registration system in Python has its limitations. Some of the common limitations of a student registration system in Python are:

* **Security concerns:** A student registration system in Python may contain sensitive information, such as student records and personal data, that must be protected from unauthorized access or disclosure. Failure to implement proper security measures can lead to data breaches and compromise the integrity of the system.
* **Cost:** Developing and maintaining a student registration system in Python can be expensive, particularly for smaller schools or universities with limited budgets. Cost considerations may limit the scope and functionality of the system.
* **Limited scalability:** A student registration system in Python may be limited in its scalability, particularly for large schools or universities with many students and programs. As the number of users and courses increases, the system may experience performance issues or require additional resources to function properly.

Overall, a student registration system in Python can provide significant benefits in terms of managing the course registration process. However, it is important to be aware of the limitations of the system and address them proactively to ensure that the system is reliable, secure, and scalable.

**1.8 MODULE**

**Login Module:** Login module will help in authentication of user accounts. User who have valid login id and password can only login into respective accounts.

**Update/Delete Module:** Suppose there are hundreds of students and from this we have to update/search a particular student and we know the name of the student, in manual system it is a tedious task though we know the name of the student, but using this module we can easily update/delete student by specifying the name of the student in the search criteria.

**1.9 PURPOSE**

The purpose of this SRS document is to allow the registration of students in particular course. It is intended to be complete specifications of what functionality the admission provides. It will also facilitate keeping all the records of students, such as their id,name,address DOB, etc. So all the information about a student will be available in a few seconds.

Overall, it will make Student Online Registration System an easier job for the administrator and the student of any organization.

The main purpose of this SRS document is to illustrate the requirements of the project Online Student Registration System and is intended to help any organization to maintain and manage its student’s personal data.

**1.10 OVERVIEW**

The purpose this documents is to present a detailed description of the SRS. It will explain the purpose and features of the software, the interfaces of the software, what the software will do, the constraints under which it must operates and how the software will react to external stimuli. This document is intended for both the end users and the developers of the software.

Specific design and implementation details will be specified in a future document. The student registration system has to handle records for many students and maintenance was difficult. Though it has used an information system, it was totally manual. Hence there is a need to upgrade the system with a computer based information system which is Online Student Registration System.

**1.11 SOFTWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| **Operating System** | Window-7 and later versions (32bit, 64 bit) |
| **Language** | Python |
| **Platform** | Python IDLE 3.7 (min) |
| **Database** | Microsoft Excel |

**HARDWARE REQUIREMENTS:**

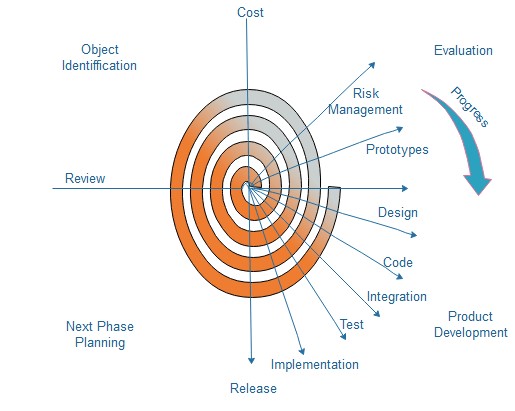
|  |  |
| --- | --- |
| **Processor** | Intel Core i7-12700K (min) 32bit or 64 bit |
| **Hard-Disk** | 256GB (min) |
| **RAM** | 1GB (min) |

**INPUT/OUTPUT REQUIREMENTS:**

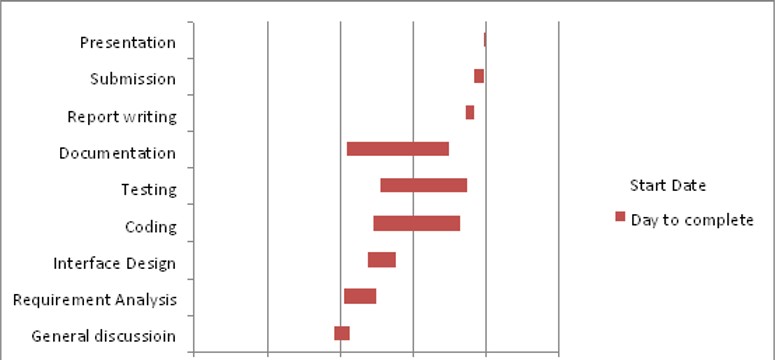
|  |  |
| --- | --- |
| **Input** | Mouse (any) |
| **Input** | Keyboard (any) |
| **Output** | Monitor (any) |
| **Output** | Printer (any) |

**1.12 PROCESS MODEL**

* **SPIRAL MODEL**



* **GNATT CHART**



15/07

29/07

12/08

26/08

2/09

9/09

**CHAPTER 2: OVERALL DESCRIPTION**

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. At last, the constraints and assumptions for the system will be presented.

In Online Student Registration System, the student is facilitated by the online system for registering students, add subjects and fee structure Our Student Online Registration System deals with the various activities related to the students.

**2.1 FEASIBILITY ANALYSIS**

Whatever we think need not be feasible .It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development ofa system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

**2.2 ECONOMICAL FEASIBILITY**

Development of this application is highly economically feasible .The organization needed not spend much money for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources .Even after the development, the organization will not be in condition to invest more in the organization .Therefore, the system is economically feasible.

**2.3 INTERFACE**

In computing, an interface is a shared boundary across which three separate components of computer system exchange information.

**User interfaces:** The application will have a user friendly and menu based interface.

### 2.4 FACT FINDING

### 2.4.1 INTRODUCTION

In order to collect information on which to base the analysis and to ascertain whether or not the information on the system meets the user’s current needs, various techniques will be used to collect information about the existing management system.

The following methods and techniques will be used to carry out fact finding on the current system in the organization.

### 2.4.2 INTERVIEWS

This method was used to collect information from the Administration (head teacher, mare teachers and employees) through face to face interaction concerning the operation of the present system. The interview covered the following areas:

1. The method currently used to admit students and to generate the students results data and the current backup storage.
2. The problems of the current system and how it operates.
3. The required output of the information that need to be kept.

### 2.4.3 OBSERVATION

An observation since it involves an analyst getting involved in some of activities of the interview it will be carried out on the operation of the current system and formulate questions and conclusions on the basis of the observation. The observation will be carried out during the operation hours.

**2.4.4 QUESTIONNAIRES**

Questionnaires will also be used and the responses got from them was that very few employees were computer literate and they were at ease with the new system since it was going to make work easier to the very busy departments

**CHAPTER 3: REQUIREMENTS AND SPECIFICATIONS**

**3.1 Software Requirement :-**

* Windows 10 Operating System
* Microsoft Visual Studio code
* Python 3.9.13 64 bit
* Microsoft Excel Database

**3.2 Hardware Requirement :-**

* Minimum 4GB RAM
* Minimum 1 TB hard disk
* Printer

**CHAPTER 4: TOOL DESCRIPTION**

**4.1 INTRODUCTION TO PYTHON**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

**4.1.1 IMPORTANCE OF PYTHON PROGRAMMING**

Python is a high-level, interpreted, interactive, and object-oriented scripting language. Python was designed to be highly readable which uses English keywords frequently whereas other languages use punctuation and it has fewer syntactical constructions than other languages.

It is used in :

1. Software Development

2. Web Development

3. System Scripting

4. Mathematics

**4.1.2 USER INTERFACE PYTHON**

### Python has a lot of [GUI frameworks](http://wiki.python.org/moin/GuiProgramming), but [Tkinter](https://wiki.python.org/moin/TkInter) is the only framework that’s built into the Python standard library. Tkinter has several strengths. It’s **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they’re run.

### Although Tkinter is considered the de facto Python GUI framework, it’s not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you’re looking for.

### However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to quickly build something that’s functional and cross-platform.

**4.1.3 FEATURES OF PYTHON**

### 1) Easy to Learn and Use

### 2) Expressive Language

### 3) Interpreted Language

### 4) Cross-platform Language

### 5) Free and Open Source

### 6) Object-Oriented Language

### 7) Extensible

### 8) Large Standard Library

### 9) GUI Programming Support

### 10) Integrated

### 11. Embeddable

### 12. Dynamic Memory Allocation

### 4.2 DATABASES

### Microsoft Excel is a spreadsheet program used to store, organize, and analyze data. It allows you to create tables of information with rows and columns, perform calculations, create charts and graphs, and more.

### Excel databases are commonly used to manage and organize large amounts of data, including customer information, inventory lists, financial records, and more. The data is usually arranged in tables, with each row representing a record and each column representing a field of information.

### Excel offers many features to make managing databases easier, including:

### Sorting and filtering: You can sort and filter data by specific criteria, making it easier to find and analyze information.

### Formulas and functions: Excel has a wide range of built-in formulas and functions that allow you to perform calculations and analysis on your data.

### Pivot tables: Pivot tables are a powerful tool for summarizing and analyzing large datasets. They allow you to quickly create summaries, charts, and reports based on your data.

### Data validation: Excel allows you to set up data validation rules, which help to ensure that data is entered correctly and consistently.

### Macros: Macros are sets of instructions that automate repetitive tasks. In Excel, macros can be used to perform complex data analysis tasks.

### Overall, Excel is a powerful tool for managing and analyzing databases of all sizes. Whether you are a small business owner, a data analyst, or a student, Excel can help you organize and analyze your data in a meaningful way.

**4.2.1 WHAT IS DATABASES**

### A database is information that is set up for easy access, management and updating. Computer databases typically store aggregations of [data](https://www.techtarget.com/searchdatamanagement/definition/data) records or [files](https://searchsqlserver.techtarget.com/definition/flat-file) that contain information, such as sales transactions, customer data, financials and product information.

### Databases are used for storing, maintaining and accessing any sort of data. They collect information on people, places or things. That information is gathered in one place so that it can be observed and analyzed. Databases can be thought of as an organized collection of information.

**4.3 INTRODUCTION TO VS CODE**

Visual Studio Code (VS Code) is a free and open-source source code editor developed by Microsoft. It's designed to be lightweight, fast, and customizable for developers of all types. VS Code supports a variety of programming languages and offers features such as syntax highlighting, code completion, debugging, and Git integration. It can be used on multiple platforms, including Windows, Linux, and macOS.

One of the unique features of VS Code is its extension system, which allows users to install and use various extensions to enhance their coding experience. There are thousands of extensions available for VS Code, including ones for specific programming languages, frameworks, and tools.

Overall, VS Code is a popular choice among developers due to its ease of use, customization options, and powerful features.

**4.3.1 OVERVIEW OF VISUAL STUDIO**

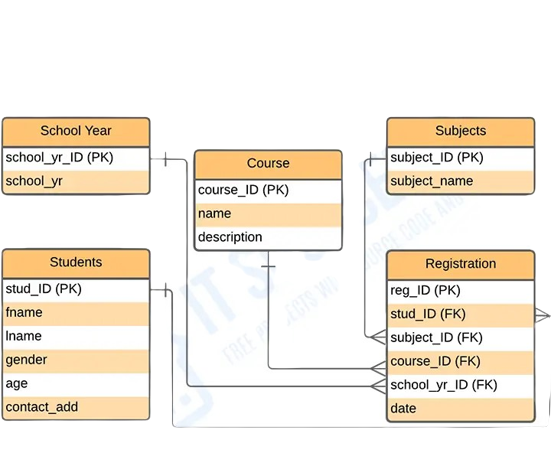
Visual Studio is an Integrated Development Environment (IDE) developed by Microsoft. It provides a complete set of tools and services for building, testing, and deploying software applications for various platforms such as Windows, Android, iOS, and web applications. Visual Studio supports a wide range of programming languages, including C++, C#, Python, Java, and many others.

Some of the key features of Visual Studio include:

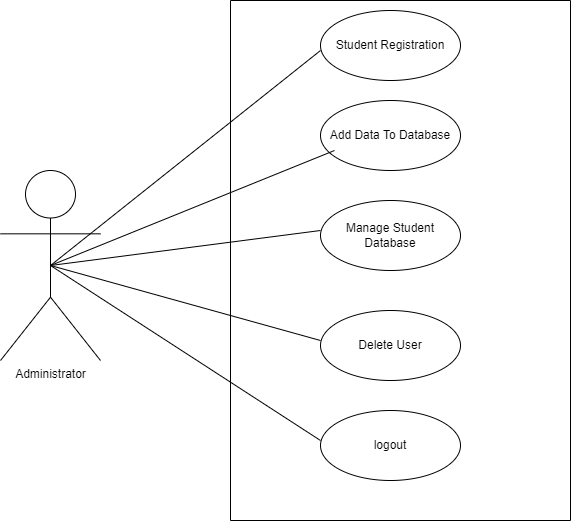
* **Code editing and debugging**: Visual Studio provides a rich code editor with features such as syntax highlighting, IntelliSense, and code refactoring. It also has advanced debugging capabilities to help developers quickly identify and fix issues.
* **Integrated testing**: Visual Studio provides various tools for testing software applications, including unit testing, performance testing, and load testing.
* **Collaboration and version control**: Visual Studio integrates with various version control systems such as Git, SVN, and Mercurial, making it easy for developers to collaborate on projects and manage code changes.
* **Cloud integration**: Visual Studio integrates with Microsoft Azure, allowing developers to easily deploy and manage applications on the cloud.
* **Cross-platform development**: Visual Studio supports building applications for multiple platforms, including Windows, Android, iOS, and web applications.

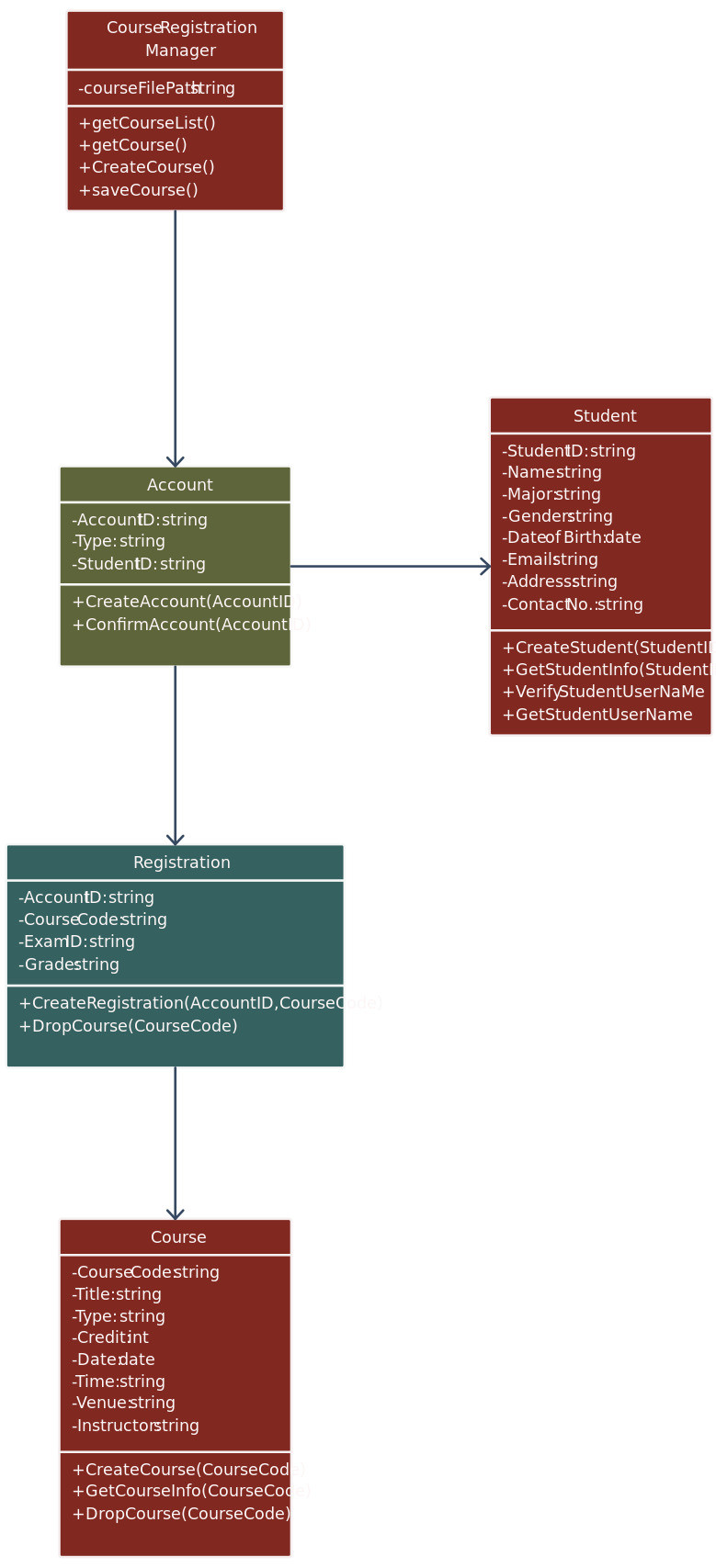
Overall, Visual Studio is a powerful IDE that provides developers with a comprehensive set of tools and services for building, testing, and deploying software applications.

**CHAPTER 5: SYSTEM DESIGN**

**5.1 ER DIAGRAM**

**5.2 USE CASE DIAGRAM**

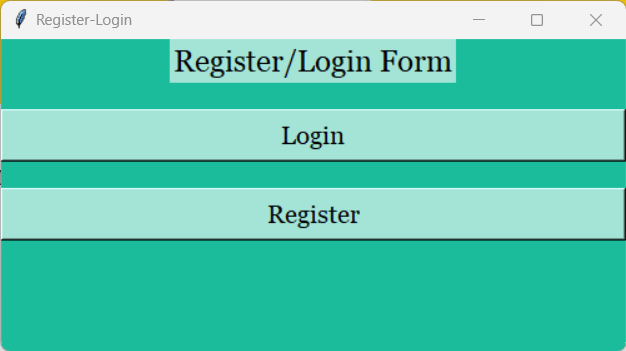


**5.3 CLASS DIAGRAM**

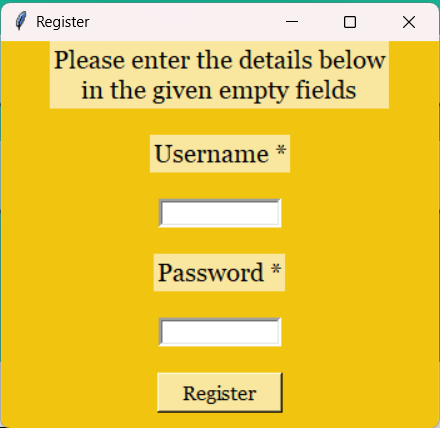
**CHAPTER 6: RESULT**

**6.1 PROJECT SNAPSHOT**

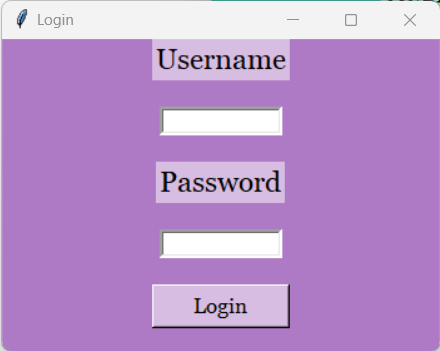
* **ADMIN REGISTRATION AND LOGIN**

****

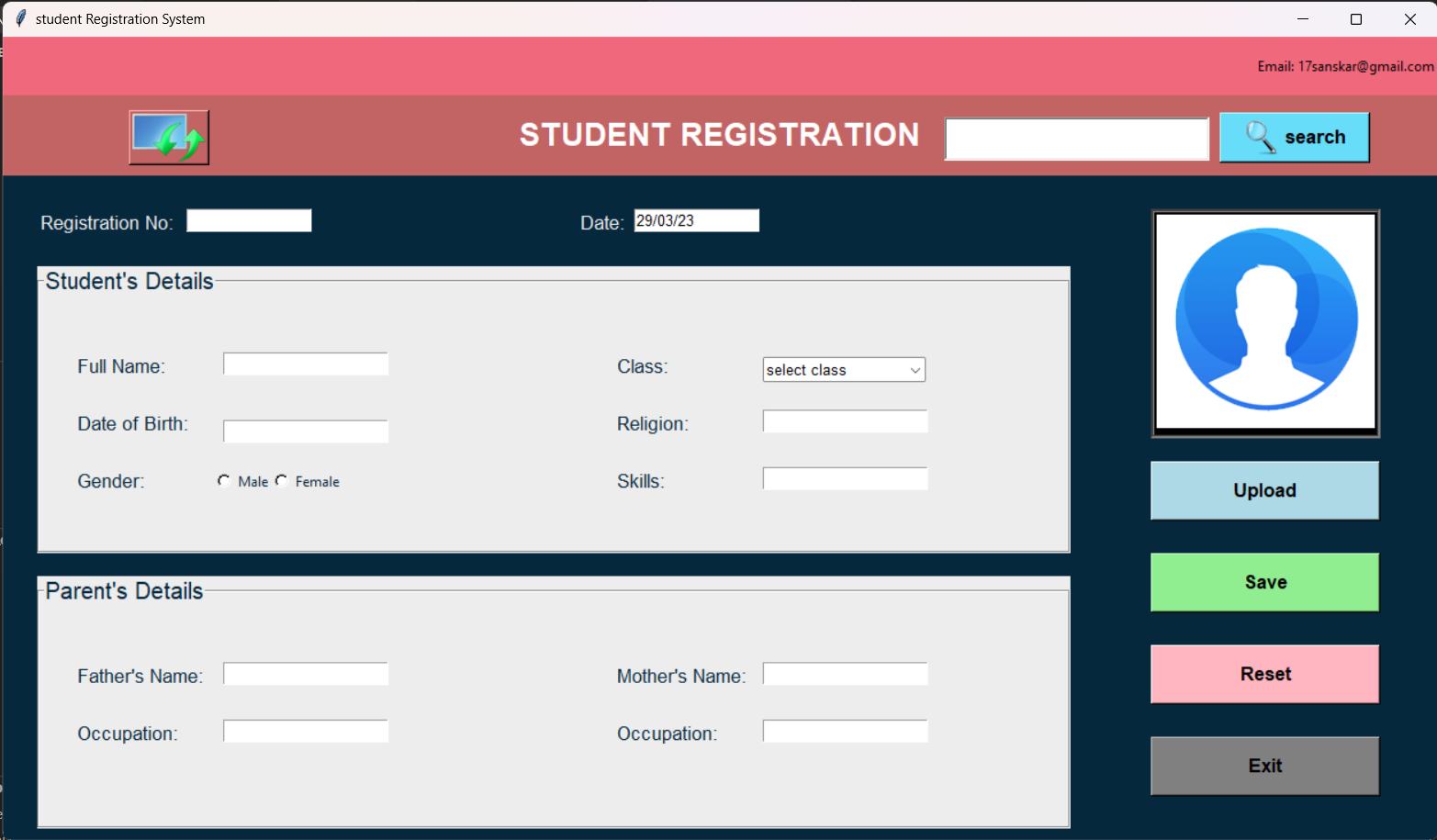
* **ADMIN REGISTRATION**

****

* **ADMIN LOGIN**

****

* **ADMIN DASHBOARD**

****

**CHAPTER 7: SYSTEM IMPLEMENTATION**

* **REGISTRATION AND LOGIN**

from tkinter import \*

from tkinter import messagebox

import os

import p1

def register\_user():

username\_info=username.get()

password\_info=password.get()

file=open(username\_info,"w")

file.write(username\_info+"\n")

file.write(password\_info)

file.close()

u\_entry.delete(0, END)

p\_entry.delete(0, END)

messagebox.showinfo("info","Registration Successful")

def login\_verify():

username1=username\_verify.get()

password1=password\_verify.get()

u\_entry1.delete(0, END)

p\_entry1.delete(0, END)

list\_of\_files=os.listdir()

if username1 in list\_of\_files:

file1=open(username1,"r")

verify=file1.read().splitlines()

if password1 in verify:

messagebox.showinfo("info","Login Successfull!!")

else:

messagebox.showinfo("info","Password has not been recognised")

else:

messagebox.showinfo("info","User not found")

def register():

global username,password

global u\_entry,p\_entry,root1

root1=Toplevel(root)

username=StringVar()

password=StringVar()

root1.title("Register")

root1.configure(bg="#F1C40F")

w1=350

h1=310

root1.geometry("%dx%d+%d+%d" %(w1,h1,x,y))

Label(root1,text="Please enter the details below\nin the given empty fields",font=("Georgia",15),bg="#F9E79F").pack()

Label(root1,text="",bg="#F1C40F").pack()

Label(root1,text="Username \*",font=("Georgia",15),bg="#F9E79F").pack()

Label(root1,text="",bg="#F1C40F").pack()

u\_entry=Entry(root1,textvariable=username,width="15",borderwidth=3)

u\_entry.pack()

Label(root1,text="",bg="#F1C40F").pack()

Label(root1,text="Password \*",font=("Georgia",15),bg="#F9E79F").pack()

Label(root1,text="",bg="#F1C40F").pack()

p\_entry=Entry(root1,textvariable=password,width="15",borderwidth=3)

p\_entry.pack()

Label(root1,text="",bg="#F1C40F").pack()

button=Button(root1,text="Register",width=10,command=lambda:register\_user(),bg="#F9E79F",font=("Georgia",12)).pack()

root1.mainloop()

def login():

global root2,u\_entry1,p\_entry1

global username\_verify,password\_verify

username\_verify=StringVar()

password\_verify=StringVar()

root2=Toplevel(root)

w2=350

h2=250

root2.geometry("%dx%d+%d+%d" %(w2,h2,x,y))

root2.configure(bg="#AF7AC5")

root2.title("Login")

Label(root2,text="Username",bg="#D7BDE2",font=("Georgia",17)).pack()

Label(root2,text="",bg="#AF7AC5").pack()

u\_entry1=Entry(root2,textvariable=username\_verify,width=15,borderwidth=3)

u\_entry1.pack()

Label(root2,text="",bg="#AF7AC5").pack()

Label(root2,text="Password",bg="#D7BDE2",font=("Georgia",17)).pack()

Label(root2,text="",bg="#AF7AC5").pack()

p\_entry1=Entry(root2,textvariable=password\_verify,width=15,borderwidth=3)

p\_entry1.pack()

Label(root2,text="",bg="#AF7AC5").pack()

button1=Button(root2,text="Login",width=10,command=lambda:login\_verify(),bg="#D7BDE2",font=("Georgia",13)).pack()

root2.mainloop()

def mainscreen():

global root,screen\_w,screen\_h,x,y

root=Tk()

screen\_w=root.winfo\_screenwidth()

screen\_h=root.winfo\_screenheight()

w=500

h=250

x=screen\_w/2.7

y=screen\_h/2.7

root.geometry("%dx%d+%d+%d" %(w,h,x,y))

root.title("Register-Login")

root.configure(bg="#1ABC9C")

heading=Label(root,text="Register/Login Form",font=("Georgia",18),bg="#A3E4D7").pack()

Label(root,text="",bg="#1ABC9C").pack()

login\_button=Button(root,text="Login",width="300",font=("Georgia",15),bg="#A3E4D7",command=lambda:login()).pack()

Label(root,text="",bg="#1ABC9C").pack()

register\_button=Button(root,text="Register",width="300",font=("Georgia",15),bg="#A3E4D7",command=lambda:register()).pack()

Label(root,text="",bg="#1ABC9C").pack()

root.mainloop()

mainscreen()

* **MAIN.py**

from tkinter import \*

from datetime import date

from tkinter import filedialog

from tkinter import messagebox

from PIL import Image, ImageTk

import os

from tkinter. ttk import Combobox

import openpyxl , xlrd

from openpyxl import Workbook

import pathlib

import os

import Login

background= "#06283D"

framebg= "#EDEDED"

framefg= "#06283D"

root=Tk()

root.title("student Registration System")

root.geometry("1250x700+210+100")

root.config(bg=background)

file=pathlib.Path('student\_data.xlsx')

if file.exists():

pass

else:

file=Workbook()

sheet=file. active

sheet[ 'A1']="Registration No."

sheet[ 'B1']="Name"

sheet[ 'C1']="Class"

sheet[ 'D1']="Gender"

sheet[ 'E1']="DOB"

sheet[ 'F1']="Date of Registration"

sheet[ 'G1']="Religion"

sheet[ 'H1']="skill"

sheet[ 'I1']="Father Name"

sheet[ 'J1']="Mother Name"

sheet[ 'K1']="Father's Occupation"

sheet[ 'L1']="Mother's Occupation"

file.save('student\_data.xlsx')

######Exit########

def Exit():

root.destroy()

#################show image#############################

def showimage():

global filename

global img

filename=filedialog.askopenfilename(initialdir=os.getcwd(),

title="Select image file", filetype=(("JPG File","\*. jpg"),

("PNG File","\*.png"),

("All files","\*.txt")))

img = (Image.open(filename))

resized\_image= img.resize((190,190))

photo2 = ImageTk.PhotoImage(resized\_image)

lbl.config(image=photo2)

lbl.image=photo2

def registration\_no():

file=openpyxl.load\_workbook( 'Student\_data.xlsx')

sheet=file.active

row=sheet.max\_row

max\_row\_value=sheet.cell(row=row, column=1).value

try:

Registration.set(max\_row\_value+1)

except:

Registration.set("1")

###################Clear#########################

def Clear():

global img

Name.set('')

DOB.set('')

Religion.set('')

skill.set('')

F\_Name.set('')

M\_Name.set('')

Father\_Occupation.set('')

Mother\_Occupation.set('')

Class.set("select class")

registration\_no()

saveButton.config(state = 'normal' )

img1=PhotoImage(file='Images/upload photo.png')

lbl. config(image=img1)

lbl.image=img1

img=""

################Save###############

def Save():

R1=Registration.get()

N1=Name.get()

C1=Class.get()

try:

G1=gender

except:

messagebox.showerror("error", "Select Gender!")

D2=DOB.get()

D1=Date.get()

Re1=Religion.get()

S1=skill.get()

fathername=F\_Name.get()

mothername=M\_Name.get()

F1=Father\_Occupation.get()

M1=Mother\_Occupation.get()

if N1=="" or C1=="Select Class" or D2=="" or Re1=="" or S1=="" or fathername=="" or mothername=="" or F1=="" or M1=="":

messagebox. showerror("error","Few Data is missing!")

else:

file=openpyxl.load\_workbook( 'Student\_data.xlsx')

sheet=file.active

sheet.cell(column=1,row=sheet.max\_row+1,value=R1)

sheet.cell(column=2,row=sheet.max\_row,value=N1)

sheet.cell(column=3,row=sheet.max\_row,value=C1)

sheet.cell(column=4,row=sheet.max\_row,value=G1)

sheet.cell(column=5,row=sheet.max\_row,value=D2)

sheet.cell(column=6,row=sheet.max\_row,value=D1)

sheet.cell(column=7,row=sheet.max\_row,value=Re1)

sheet.cell(column=8,row=sheet.max\_row,value=S1)

sheet.cell(column=9,row=sheet.max\_row,value=fathername)

sheet.cell(column=10,row=sheet.max\_row,value=mothername)

sheet.cell(column=11,row=sheet.max\_row,value=F1)

sheet.cell(column=12,row=sheet.max\_row,value=M1)

file. save(r'Student\_data.xlsx')

try:

img.save("Student Images/"+str(R1)+". jpg")

except:

messagebox. showinfo("info","Profile Picture is not available!!!!")

messagebox. showinfo("info","Sucessfully data entered!!!")

Clear() #clear entry box and image section

registration\_no() #it will recheck registration no. and reissue new no.

####################### search ###########################

def Search():

text = Search.get() #taking input from entry box

Clear() # to clear all the data already available in entry box and other

saveButton.config(state='disable') #after clicking on search , save button will disable so that no one can click on it

file=openpyxl. load\_workbook("Student\_data.xlsx")

sheet=file.active

for row in sheet.rows:

if row[0].value == int(text):

name=row[0]

# print(str(name))

reg\_no\_position=str(name) [14:-1]

reg\_number=str(name)[15:-1]

# print (reg\_no\_position)

# print(reg\_number)

try:

print(str(name))

except:

messagebox.showerror("Invalid","Invalid registration number!!!")

#reg\_no\_position showing like A2,A3,A4,.....An

#but reg\_number just showing number after A2 like 2,3,......,n

x1=sheet.cell(row=int(reg\_number) ,column=1).value

x2=sheet.cell(row=int(reg\_number) ,column=2).value

x3=sheet.cell(row=int(reg\_number) ,column=3).value

x4=sheet.cell(row=int(reg\_number),column=4).value

x5=sheet.cell(row=int(reg\_number),column=5).value

x6=sheet.cell(row=int(reg\_number) ,column=6).value

x7=sheet.cell(row=int(reg\_number),column=7) .value

x8=sheet.cell(row=int(reg\_number) ,column=8) .value

x9=sheet.cell(row=int(reg\_number) ,column=9).value

x10=sheet.cell(row=int(reg\_number),column=10) .value

x11=sheet.cell(row=int(reg\_number) ,column=11).value

x12=sheet.cell(row=int(reg\_number) ,column=12).value

print(x1)

print(x2)

print(x3)

print(x4)

print(x5)

print(x6)

print(x7)

print(x8)

print(x9)

print(x11)

print(x12)

Registration.set(x1)

Name.set (x2)

Class.set (x3)

if x4=='Female':

R2.select()

else:

R1.select()

DOB. set(x5)

Date. set (x6)

Religion.set(x7)

skill.set(x8)

F\_Name.set (x9)

M\_Name.set(x10)

Father\_Occupation.set(x11)

Mother\_Occupation.set(x12)

img = (Image.open("Student Images/"+str(x1)+".jpg"))

resized\_image=img.resize((190,190))

photo2 = ImageTk.PhotoImage(resized\_image)

lbl. config(image=photo2)

lbl.image=photo2

###################Update#################

def Update():

R1=Registration.get()

N1=Name.get()

C1=Class.get()

selection()

G1=gender

D2=DOB.get()

D1=Date.get()

Re1=Religion.get()

S1=skill.get()

fathername=F\_Name.get()

mothername=M\_Name.get()

F1=Father\_Occupation.get()

M1=Mother\_Occupation.get()

file=openpyxl.load\_workbook("Student\_data.xlsx")

sheet=file.active

for row in sheet.rows:

if row[0].value == R1:

name=row[0]

print(str(name) )

reg\_no\_position=str(name)[14:-1]

reg\_number=str(name)[15:-1]

print(reg\_number)

sheet.cell(column=1, row=int(reg\_number),value=R1)

sheet.cell(column=2, row=int (reg\_number),value=N1)

sheet.cell(column=3, row=int(reg\_number),value=C1)

sheet.cell(column=4, row=int(reg\_number),value=G1)

sheet.cell(column=5, row=int(reg\_number),value=D2)

sheet.cell(column=6, row=int(reg\_number),value=D1)

sheet.cell(column=7, row=int(reg\_number),value=Re1)

sheet.cell(column=8, row=int(reg\_number),value=S1)

sheet.cell(column=9, row=int(reg\_number),value=fathername)

sheet.cell(column=10, row=int(reg\_number),value=mothername)

sheet.cell(column=11, row=int(reg\_number),value=F1)

sheet.cell(column=12, row=int(reg\_number),value=M1)

file.save(r'Student\_data.xlsx')

try:

img.save("Student Images/"+str(x1)+".jpg")

except:

pass

messagebox.showinfo("Update","Update Sucessfully!!")

Clear() #after update it will clear all entry boxes |

##### gander #########

def selection():

global gender

value=radio.get()

if value==1:

gender= "Male"

else:

gender="Female"

##########top frames##############

Label(root,text="Email: 17sanskar@gmail.com",width=10,height=3,bg="#f0687c",anchor='e').pack(side=TOP,fill=X)

Label(root,text="STUDENT REGISTRATION",width=10,height=2,bg="#c36464",fg='#fff',font='arial 20 bold').pack(side=TOP,fill=X)

#search box to update

Search=StringVar()

Entry(root, textvariable=Search,width=15,bd=2,font="arial 20").place(x=820,y=70)

imageicon3=PhotoImage(file="Images/search.png")

srch=Button(root,text="search",compound=LEFT,image=imageicon3,width=123,bg='#68ddfa',font="arial 13 bold",command=Search)

srch.place(x=1060,y=66)

imageicon4=PhotoImage (file= "Images/Layer 4.png")

update\_button=Button(root, image=imageicon4,bg= "#c36464",command=Update)

update\_button.place(x=110,y=64)

#Registration and Date

Label(root,text="Registration No:",font="arial 13",fg=framebg, bg=background).place(x=30,y=150)

Label(root,text="Date:",font="arial 13",fg=framebg,bg=background).place(x=500,y=150)

Registration=IntVar()

Date = StringVar()

reg\_entry = Entry(root, textvariable=Registration,width=15, font= "arial 10")

reg\_entry.place (x=160,y=150)

registration\_no()

today = date.today()

d1 = today.strftime("%d/%m/%y")

date\_entry = Entry(root, textvariable=Date, width=15, font= "arial 10")

date\_entry.place(x=550,y=150)

Date.set(d1)

#Student details

obj=LabelFrame (root, text= "Student's Details",font=20, bd=2, width=900, bg=framebg, fg=framefg, height=250, relief=GROOVE)

obj.place(x=30,y=200)

Label(obj, text= "Full Name:",font="arial 13" ,bg=framebg, fg=framefg).place(x=30,y=50)

Label(obj, text= "Date of Birth:",font="arial 13",bg=framebg, fg=framefg).place(x=30,y=100)

Label(obj, text= "Gender:",font="arial 13" ,bg=framebg, fg=framefg).place(x=30,y=150)

Label(obj, text= "Class:",font="arial 13" ,bg=framebg, fg=framefg).place (x=500,y=50)

Label(obj, text= "Religion:",font="arial 13" ,bg=framebg, fg=framefg).place (x=500,y=100)

Label(obj, text= "Skills:",font="arial 13" ,bg=framebg, fg=framefg).place (x=500,y=150)

Name=StringVar()

name\_entry = Entry(obj,textvariable=Name,width=20,font="arial 10")

name\_entry.place(x=160,y=50)

DOB=StringVar()

dob\_entry = Entry(obj,textvariable=DOB,width=20,font="arial 10")

dob\_entry.place(x=160, y=109)

radio= IntVar()

R1 = Radiobutton(obj,text="Male", variable=radio, value=1,bg=framebg, fg=framefg, command=selection)

R1.place(x=150, y=150)

R2 = Radiobutton(obj,text="Female", variable=radio, value=2,bg=framebg, fg=framefg, command=selection)

R2.place(x=200,y=150)

Religion=StringVar()

religion\_entry = Entry(obj,textvariable=Religion,width=20,font="arial 10")

religion\_entry.place(x=630,y=100)

skill=StringVar()

skill\_entry = Entry(obj,textvariable=skill,width=20,font="arial 10")

skill\_entry.place(x=630, y=150)

Class= Combobox(obj,values=['1','2','3','4','5','6','7','8','9','10','11','12'],font="Roboto 10",width=17,state="r")

Class.place(x=630,y=54)

Class.set("select class")

#Parents details

obj2=LabelFrame(root, text= "Parent's Details" ,font=20, bd=2, width=900, bg=framebg, fg=framefg, height=220, relief=GROOVE)

obj2.place(x=30,y=470)

Label(obj2,text="Father's Name:",font="arial 13",bg=framebg, fg=framefg) .place(x=30,y=50)

Label(obj2, text="Occupation:",font="arial 13",bg=framebg,fg=framefg) .place(x=30, y=100)

F\_Name=StringVar()

f\_entry = Entry(obj2,textvariable=F\_Name,width=20,font="arial 10")

f\_entry.place(x=160,y=50)

Father\_Occupation=StringVar()

FO\_entry = Entry(obj2,textvariable=Father\_Occupation,width=20,font="arial 10")

FO\_entry.place(x=160,y=100)

Label (obj2,text="Mother's Name:",font="arial 13",bg=framebg, fg=framefg) .place(x=500, y=50)

Label(obj2,text="Occupation:",font="arial 13",bg=framebg, fg=framefg) .place(x=500,y=100)

M\_Name=StringVar()

M\_entry = Entry(obj2,textvariable=M\_Name,width=20,font="arial 10")

M\_entry.place(x=630,y=50)

Mother\_Occupation=StringVar()

Mo\_entry = Entry(obj2,textvariable=Mother\_Occupation,width=20,font="arial 10")

Mo\_entry.place(x=630,y=100)

#######image##########

f=Frame(root, bd=3,bg="black" ,width=200, height=200, relief=GROOVE)

f.place(x=1000, y=150)

img=PhotoImage(file="Images/upload photo.png")

lbl=Label(f,bg="black", image=img)

lbl.place(x=0, y=0)

#button

Button(root, text="Upload" ,width=19, height=2, font="arial 12 bold", bg="lightblue",command=showimage).place(x=1000, y=370)

saveButton=Button(root, text="Save" ,width=19, height=2,font="arial 12 bold",bg="lightgreen",command=Save)

saveButton.place(x=1000, y=450)

Button(root, text="Reset" ,width=19, height=2,font="arial 12 bold",bg="lightpink",command=Clear).place(x=1000, y=530)

Button(root, text="Exit",width=19, height=2, font="arial 12 bold",bg="grey",command=Exit).place(x=1000, y=610)

root.mainloop()

**CHAPTER 8: CONCLUSION AND FUTURE ENHANCEMENT**

**8.1 CONCLUSION**

A student registration system in Python can greatly benefit educational institutions by streamlining their enrollment processes, improving student experiences, and increasing administrative efficiency. By using Python's libraries and frameworks, developers can easily build a user-friendly interface for students to enroll in courses, input personal information, and pay for tuition fees. Additionally, the system can provide administrators with access to important data, such as enrollment numbers and student records, which can be used to improve the overall performance of the institution. Overall, a student registration system in Python is a powerful tool for educational institutions looking to improve their enrollment processes and enhance the experience of their students.

**8.2 FUTURE SCOPE**

The future scope of a student registration system in Python is vast, as advancements in technology and education continue to evolve. Here are some potential areas where this system could be further developed and enhanced:

* **Integration with Learning Management Systems (LMS):** With the increased use of LMS platforms in education, the integration of student registration systems with these platforms can provide a seamless experience for students and faculty. This integration can enable students to access their courses and course materials directly from the registration system, reducing the need for multiple logins and interfaces.
* **Machine Learning and Data Analytics:** Machine learning algorithms and data analytics techniques can be used to analyze student data collected through the registration system. This data can be used to predict student behavior, improve course offerings, and personalize the learning experience for individual students.
* **Mobile App Development:** The development of a mobile application for the student registration system can provide students with the convenience of registering for courses, accessing schedules, and receiving notifications on their mobile devices.
* **Blockchain Integration:** Blockchain technology can be integrated with the student registration system to improve the security and transparency of the system. This integration can enable students to have control over their educational records, such as certifications and degrees, and share them with potential employers or educational institutions.

Overall, the future scope of a student registration system in Python is promising, as technology and education continue to evolve, and new developments can further enhance the functionality and effectiveness of the system.

**8.3 LESSON LEARNT**

Doing something for long time periods always gives good lesson. Some of the things that our team learnt are listed as below:

* Learnt about python, its components and ways to implement them
* Learnt to work in pressure and to be patient
* Learnt to manage the database under Excel

**CHAPTER 9: REFRENCES**

* [**https://www.python.org/**](https://www.python.org/)
* [**https://www.microsoft.com/en-in/microsoft-365/excel**](https://www.microsoft.com/en-in/microsoft-365/excel)
* [**https://www.youtube.com/watch?v=JUGEkFDeuwg**](https://www.youtube.com/watch?v=JUGEkFDeuwg)