**AN INTERNSHIP REPORT ON**

**“STUDENT MANAGEMENT SYSTEM”**

***Submitted to the Dept. of Information Technology, SNIST***

***in partial fulfilment of the academic requirements for the award of***

**B.Tech (Information Technology)**

**under JNTUH**

**By**

**T. HRUTHIK REDDY (17311A1263)**

**L. ABHINAY (17311A12A7)**

**V. VENKATESH (17311A12A8)**

**Through Goal Street**

**Under the guidance of**

Dr.Ch.Mukunda



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SREENIDHI INSTITUTION OF SCIENCE AND TECHNOLOGY**

**(An Autonomous Institution)**

**(AFFILIATED TO JNTUH, HYDERABAD)**

**Yamnampet , Ghatkesar, R.R. Dist , Hyd-501301**

**2020.**



**CERTIFICATE**

This is to certify that the Internship-Project report on **“STUDENT MANAGEMENT SYSTEM"** is bonafide work done and submitted by (T.HRUTHIK REDDY-17311A1263,L.ABHINAY-17311A12A7,V VENKATESH-17311A12A8**) ,** in partial fulfilment of the requirement for the award of Degree of **B.Tech degree in Information Technology, SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY , Affiliated to Jawaharlal Nehru Technological University, Hyderabad** is a record of bonafide work carried out by him, under our guidance and supervision from to **May 2020 – August 2020.**

The results presented in this dissertation have been verified and are found to be satisfactory. The results embodied in this dissertation have not been submitted to any other university for the award of any other degree or diploma.

**Project Coordinator Head of the Department**

Dr.Ch.Mukunda. Dr.V.V.S.S.Balaram.

Department of IT.

SNIST.

**ACKNOWLEDGMENT**

Project development is not an easy task. It requires co-operation and help of various people. Apart from my efforts ,the success of any project depends largely on the encouragement and guidelines of may others. We are really thankful and sincerely want to inspire my feeling of gratitude towards the one who helped in the completion of the project.

First my grateful thank to Dr.V.V.S.S.Balaram (HOD) and Dr.Ch.Mukunda for the golden oppurtunity. Today after completing my project work, I am immensly satisfied.There were many times during the span of making the project when the clock beats me to learn out of energy at that moment Faceprep helped me such situation. He also helped me to have a theoretical knowledge and practical knowledge about the Data Science.

T. HRUTHIK REDDY (17311A1263)

L. ABHINAY (17311A12A7)

V. VENKATESH (17311A12A8)

**CONTENTS**

ACKNOWLEDGEMENT -------------------------------------------------------- 3

ABSTRACT ------------------------------------------------------------------------ 5

**CHAPTER 1**

INTRODUCTION ----------------------------------------------------------------- 6-15

PYTHON DESCRIPTION ----------------------------------------------------------- 6

DATABASE ------------------------------------------------------------------------- 14

**CHAPTER 2**

MODULES IMPORTED ---------------------------------------------------- 16-19

**CHAPTER 3**

USE CASE DIAGRAM**---------------------------------------------------------20**

ADDED FEILDS ----------------------------------------------------------------21

**CHAPTER 4**

TECHNOLOGY OVERVIEW --------------------------------------------------- 22-23

**CHAPTER 5**

SOURCE CODE ------------------------------------------------------------------24-55

**CHAPTER 6**

FUTURE ENHANCEMENT ---------------------------------------------------- 56

**CHAPTER 7**

CONCLUSION --------------------------------------------------------------------- 57

**CHAPTER 8**

SOLUTION ----------------------------------------------------------------------- 17-18

**STUDENT MANAGEMENT SYSTEM**

**ABSTRACT**

Student Information Management System can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project.

Name of the Project: Student Information Management System

Students form a main part of any institution that concerns with. But the institutions find it difficult to keep details of so many students of the organization just in one stretch. It will involve a lot of pen paper work. Sometimes there will be some huge heap of files bundled up and kept together in some corner of the office. If you want any information regarding the particular student then it can be obtained by just entering the roll number or the name of the student to be searched. This student management system will make the work of storing the data in an organized way.

The student management system application will help in managing the student’s reports, results and exams will become easier with one such system. It will also help in saving time and effort. The user interface must be user friendly and easy to understand. The information of the particular student will be obtained in just one mouse click

**CHAPTER 1**

**INTRODUCTION**

**PYTHON INTRODUCTION:**

**Python** is a programming language, which means it’a a language both people and computers can understand. Python was developed by a Dutch software engineer named Guido van Rossum, who created the language to solve some problems he saw in computer languages of the time.

**Python** is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

**You Can Use Python for Pretty Much Anything**

One significant advantage of learning Python is that it’s a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

* Data science
* Scientific and mathematical computing
* Web development
* Computer graphics
* Basic game development
* Mapping and geography (GIS software)

**Python Is Widely Used in Data Science**

Python’s ecosystem is growing over the years and it’s more and more capable of the statistical analysis.

It’s the best compromise between scale and sophistication (in terms od data processing).

Python emphasizes productivity and readability.

Python is used by programmers that want to delve into data analysis or apply statistical techniques (and by devs that turn to data science)

There are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as Mat Lab. The most popular libraries and tools for data science are:

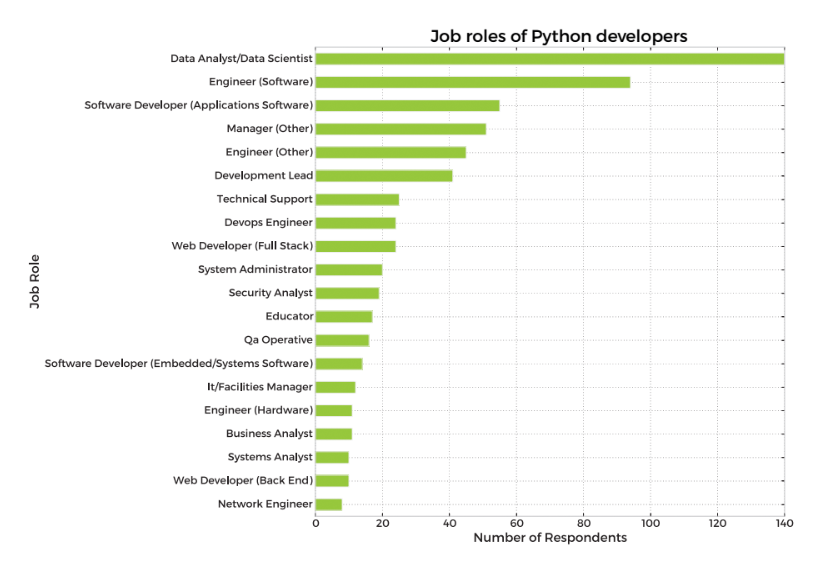
**Pandas**: a library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

**NumPy**: the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

**SciPy**: a library used by scientists, analysts, and engineers doing scientific computing and technical computing.

Being a free, cross-platform, general-purpose and high-level programming language, Python has been widely adopted by the scientific community. Scientists value Python for its precise and efficient syntax, relatively flat learning curve and the fact that it integrates well with other languages (e.g. C/C++).

As a result of this popularity there are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as Mat Lab.



**Here’s our list of the most popular Python scientific libraries and tools**

**Astropy**

The Astropy Project is a collection of packages designed for use in astronomy. The core astropy package contains functionality aimed at professional astronomers and astrophysicists, but may be useful to anyone developing astronomy software.

**Biopython**

Biopython is a collection of non-commercial Python tools for computational biology and bioinformatics. It contains classes to represent biological sequences and sequence annotations, and it is able to read and write to a variety of file formats.

**Cubes**

Cubes is a light-weight Python framework and set of tools for the development of reporting and analytical applications, Online Analytical Processing (OLAP), multidimensional analysis and browsing of aggregated data.

**DEAP**

DEAP is an evolutionary computation framework for rapid prototyping and testing of ideas. It incorporates the data structures and tools required to implement most common evolutionary computation techniques such as genetic algorithm, genetic programming, evolution strategies, particle swarm optimization, differential evolution and estimation of distribution algorithm.

**SCOOP**

SCOOP is a Python module for distributing concurrent parallel tasks on various environments, from heterogeneous grids of workstations to supercomputers.

**PsychoPy**

PsychoPy is a package for the generation of experiments for neuroscience and experimental psychology. PsychoPy is designed to allow the presentation of stimuli and collection of data for a wide range of neuroscience, psychology and psychophysics experiments.

**Pandas**

Pandas is a library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

**Mlpy**

Mlpy is a machine learning library built on top of NumPy/SciPy, the GNU Scientific Libraries. Mlpy provides a wide range of machine learning methods for supervised and unsupervised problems and it is aimed at finding a reasonable compromise between modularity, maintainability, reproducibility, usability and efficiency.

**matplotlib**

Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib allows you to generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, and more.

**NumPy**

NumPy is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

**NetworkX**

NetworkX is a library for studying graphs which helps you create, manipulate, and study the structure, dynamics, and functions of complex networks.

**TomoPy**

TomoPy is an open-sourced Python toolbox to perform tomographic data processing and image reconstruction tasks. TomoPy provides a collaborative framework for the analysis of synchrotron tomographic data with the goal to unify the effort of different facilities and beamlines performing similar tasks.

**Theano**

Theano is a numerical computation Python library. Theano allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

**SymPy**

SymPy is a library for symbolic computation and includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. It provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web.

**SciPy**

SciPy is a library used by scientists, analysts, and engineers doing scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

**Scikit-learn**

Scikit-learn is a machine learning library. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**Scikit-image**

Scikit-image is a image processing library. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

**Python Saves Time**

Even the classic “Hello, world” program illustrates this point:

print("Hello, world")

For comparison, this is what the same program looks like in Java:

public class HelloWorld {

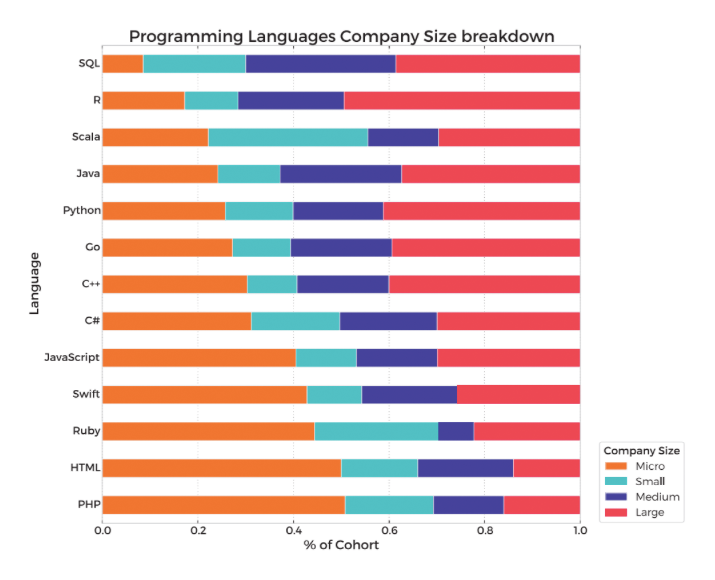
public static void main(String[] args) {

System.out.println("Hello, world");

}

}

**All the Big Names Use Python**

****

**Python Keywords and Identifier**

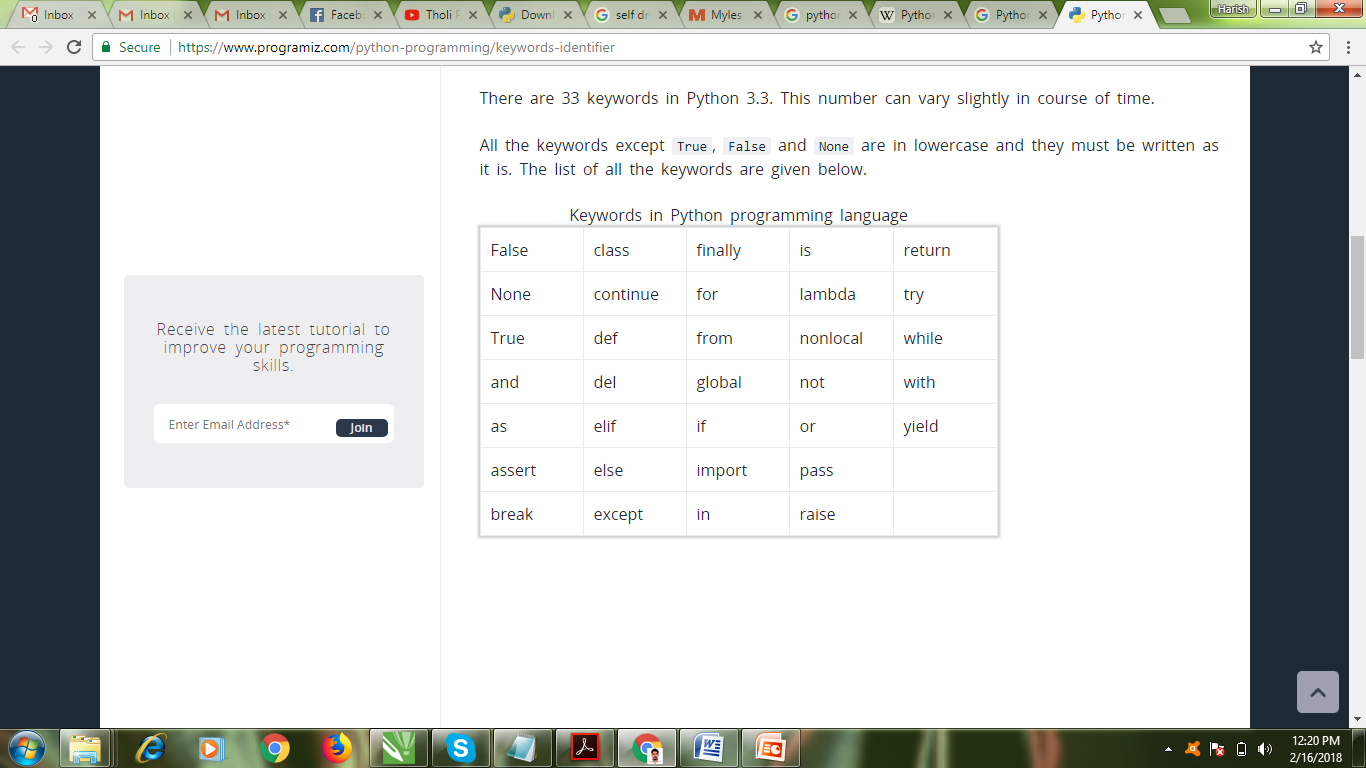
Keywords are the reserved words in Python.

We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.

In Python, keywords are case sensitive.

There are 33 keywords in Python 3.3. This number can vary slightly in course of time.

All the keywords except True, False and None are in lowercase and they must be written as it is. The list of all the keywords is given below



Identifier is the name given to entities like class, functions, variables etc. in Python. It helps differentiating one entity from another.

**Rules for writing identifiers**

Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (\_). Names like myClass, var\_1 and print\_this\_to\_screen, all are valid example.

An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.

Keywords cannot be used as identifiers.

>>> global = 1

File "<interactive input>", line 1

global = 1

^

SyntaxError: invalid syntax

We cannot use special symbols like !, @, #, $, % etc. in our identifier.

>>> a@ = 0

File "<interactive input>", line 1

a@ = 0

^

SyntaxError: invalid syntax

Identifier can be of any length.

**Python**

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.



Python Logo

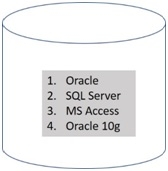
**DATABASE:**

A collected information which is in an organized form for easier access, management, and various updating is known as a database.

Before going into a further discussion of databases, we must have a prior knowledge of exactly what is a DATA? Data can be defined as a collection of facts and records on which we can apply reasoning or can-do discussion or some calculation. The data is always easily available and is in plenty. It can be used for processing some useful information from it. Also, it can be in redundant, can be irrelevant. Data can exist in form of graphics, reports, tables, text, etc. that represents every kind of information, that allows easy retrieval, updating, analysis, and output of data by systematically organized or structured repository of indexed information.

**Containers having a huge amount of data are known as databases,** for example, a public library stores books. Databases are computer structures that save, organize, protect, and deliver data.

Any system that manages databases is called a **database management system**, or DBM. The typical diagram representation for a database is a cylinder.



Inside a database, the data is recorded in a table which is a collection of rows, columns, and it is indexed so that to find relevant information becomes an easier task. As new information is added, data gets updated, expanded and deleted. The various processes of databases create and update themselves, querying the data they contain and running applications against it.

The are several different types of database models have been developed so far, for example, *flat, hierarchical, network* and *relational*. These models describe the operations that can be performed on them as well as the structure of the conforming databases. Normally there is a database schema which describes the exact model, entity types, and relationships among those entities.

**Flat Databases** have the following characteristics −

* simple
* long and dominant
* useful for very small scale and simple applications.

A **Relational Database** has the following characteristics −

* organizes data such that it appears to the user to be stored in a series of interrelated tables
* used for high-performance applications
* efficient
* ease of use
* ability to perform a variety of useful tasks

**CHAPTER 2**

**MODULES IMPORTED**

**Tkinter module:**

**Graphical User Interface(GUI)** is a form of user interface which allows users to interact with computers through visual indicators using items such as icons, menus, windows, etc. It has advantages over the Command Line Interface(CLI) where users interact with computers by writing commands using keyboard only and whose usage is more difficult than GUI.

**Tkinter** is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don’t need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.  
  
Some other Python Libraries available for creating our own GUI applications are

 Kivy

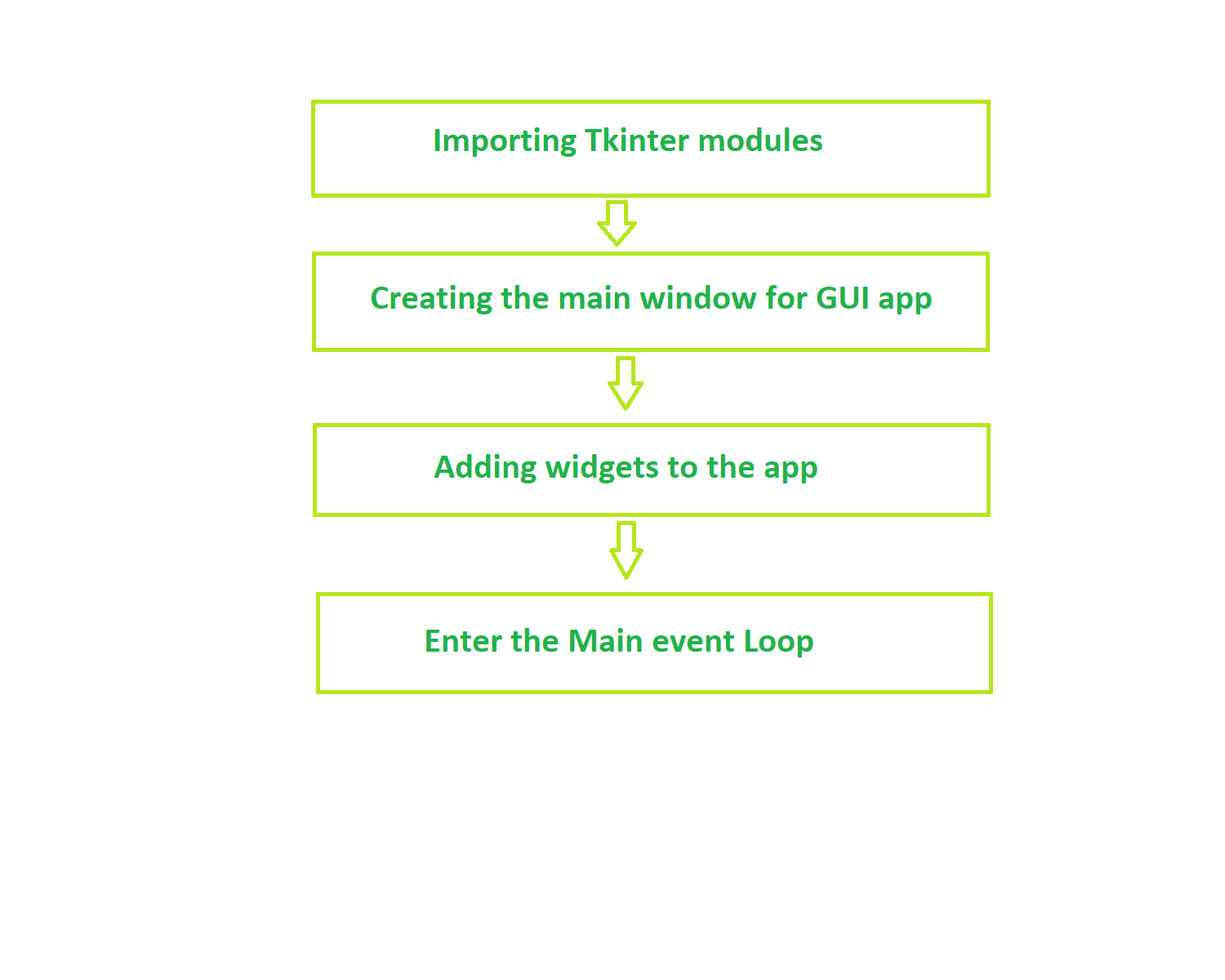
 Python Qt

 wxPython

Among all Tkinter is most widely used

**Widgets** in Tkinter are the elements of GUI application which provides various controls (such as Labels, Buttons, ComboBoxes, CheckBoxes, MenuBars, RadioButtons and many more) to users to interact with the application.

|  |  |
| --- | --- |
| **WIDGETS** | **DESCRIPTION** |
| Label | It is used to display text or image on the screen |
| Button | It is used to add buttons to your application |
| Canvas | It is used to draw pictures and others layouts like texts, graphics etc. |
| ComboBox | It contains a down arrow to select from list of available options |
| CheckButton | It displays a number of options to the user as toggle buttons from which user can select any number of options. |
| RadiButton | It is used to implement one-of-many selection as it allows only one option to be selected |
| Entry | It is used to input single line text entry from user |
| Frame | It is used as container to hold and organize the widgets |
| Message | It works same as that of label and refers to multi-line and non-editable text |
| Scale | It is used to provide a graphical slider which allows to select any value from that scale |
| Scrollbar | It is used to scroll down the contents. It provides a slide controller. |
| SpinBox | It is allows user to select from given set of values |
| Text | It allows user to edit multiline text and format the way it has to be displayed |
| Menu | It is used to create all kinds of menu used by an application |

**Fundamental structure of tkinter program**  
  


**PANDAS:**

Pandas is an open source library that is used to analyze data in Python. It takes in data, like a CSV or SQL database, and creates an object with rows and columns called a data frame. Pandas is typically imported with the alias pd.

# Ways of creating a Pandas DataFrame # Passing in a dictionary: data = {'name':['Anthony', 'Maria'], 'age':[30, 28]} df = pd.DataFrame(data) # Passing in a list of lists: data = [['Tom', 20], ['Jack', 30], ['Meera', 25]] df = pd.DataFrame(data, columns = ['Name', 'Age']) # Reading data from a csv file: df = pd.read\_csv('students.csv')

The fundamental Pandas object is called a DataFrame. It is a 2-dimensional size-mutable, potentially heterogeneous, tabular data structure.

A DataFrame can be created multiple ways. It can be created by passing in a dictionary or a list of lists to the pd.DataFrame() method, or by reading data from a CSV file.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## Key Features of Pandas

* Fast and efficient DataFrame object with default and customized indexing.
* Tools for loading data into in-memory data objects from different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of date sets.
* Label-based slicing, indexing and subsetting of large data sets.
* Columns from a data structure can be deleted or inserted.
* Group by data for aggregation and transformations.
* High performance merging and joining of data.
* Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install NumPy using popular Python package installer, **pip.**

pip install pandas

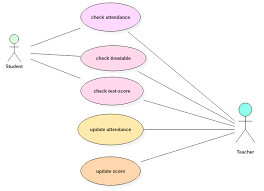
If you install Anaconda Python package, Pandas will be installed by default with the following −

## Windows

* **Anaconda** (from [https://www.continuum.io](https://www.continuum.io/)) is a free Python distribution for SciPy stack. It is also available for Linux and Mac.
* **Canopy** ([https://www.enthought.com/products/canopy/](https://www.enthought.com/products/canopy)) is available as free as well as commercial distribution with full SciPy stack for Windows, Linux and Mac.
* **Python** (x,y) is a free Python distribution with SciPy stack and Spyder IDE for Windows OS. (Downloadable from <http://python-xy.github.io/>)

**CHAPTER 3**

**USECASE DIAGRAM**



A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Due to their simplistic nature, use case diagrams can be a good communication tool for [stakeholders](https://en.wikipedia.org/wiki/Project_stakeholder). The drawings attempt to mimic the real world and provide a view for the [stakeholder](https://en.wikipedia.org/wiki/Project_stakeholder) to understand how the system is going to be designed. Siau and Lee conducted research to determine if there was a valid situation for use case diagrams at all or if they were unnecessary. What was found was that the use case diagrams conveyed the intent of the system in a more simplified manner to [stakeholders](https://en.wikipedia.org/wiki/Project_stakeholder) and that they were "interpreted more completely than class diagrams"

**ADD STUDENT:**

We can add the student by filling the form with the appropriate details and that is stored in the database and that is displayed by side frame.

**SEARCH STUDENT**:

We can search the student by entering the values and that is displayed in the side frame.

**DELETE STUDENT:**

We can select the student in displayed box and delete the particular student and the student is deleted and not displayed in the side frame.

**UPDATE STUDENT:**

We can update the student information in the database and as well as display on the screen the updated one.

**SHOW ALL**:

This displays the student information in the side frame when we click on it.

**EXPORT DATA:**

With this we can export the data to a particular folder and check them and can enter a particular date or time and import those data

**EXIT:**

This will exit the window that is front of you

**CHAPTER 4**

**TECHNOLOGY OVERVIEW**

The technology selected for implementing Student Information Management System is PYTHON/MYSQL .The development was done in a ‘windows’ environment using PYcharm IDE and MYSQL database.

**MYSQL**

MySQL is a relational database management system (RDBMS)[1] that runs as a server providing multi-user access to a number of databases. MySQL is a popular choice of database for use in web applications and is an open source product. The process of setting up a MySQL database varies from host to host, however we will end up with a database name, a user name and a password. Before using our database, we must create a table. A table is a section of the database for storing related information. In a table we will set up the different fields which will be used in that table. Creating a table in phpMyAdmin is simple, we just type the name, select the number of fields and click the ‘go’ button. we will then be taken to a setup screen where you must create the fields for the database.Another way of creating databases and tables in phpMyAdmin is by executing simple SQL statements.We have used this method in order to create our database and tables.

The Apache HTTP Server is a web server software notable for playing a key role in the initial growth of the World Wide Web. In 2009 it became the first web server software to surpass the 100 million web site milestone. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Since April 1996 Apache has been the most popular HTTP server software in use. As of November 2010 Apache served over 59.36% of all websites and over 66.56% of the first one million busiest websites.

XAMPP is a small and light Apache distribution containing the most common web development technologies in a single package. Its contents, small size, and portability make it the ideal tool for students developing and testing applications in PHP and MySQL. XAMPP is available as a free download in two specific packages: full and lite. While the full package download provides a wide array of development tools, XAMPP Lite contains the necessary technologies that meet the Ontario Skills Competition standards.The light version is a small package containing Apache HTTP Server, PHP, MySQL, phpMyAdmin, Openssl, and SQLite. Obtaining and Installing XAMPP As previously mentioned, XAMPP is a free package available for download and use for various web development tasks. All XAMPP packages and add-ons are distributed through the Apache Friends website at the address: http://www.apachefriends.org/. Once on the website, navigate and find the Windows version of XAMPP and download the self-extracting ZIP archive. After downloading the archive, run and extract its contents into the root path of a hard disk or USB drive. For example, the extract path for a local Windows installation would simply be C:\. If extracted properly we will notice a new xampp directory in the root of your installation disk. In order to test that everything has been installed correctly, first start the Apache HTTP Server by navigating to the xampp directory and clicking on the apache\_start.bat batch file. Next we will test if the server is running correctly by opening an internet browser and typing http://localhost/ into the address bar. If configured correctly, we will be presented with a screen similar to that of the one below. XAMPP splash screen. In order to stop all Apache processes we do not close the running terminal application, but instead run another batch file in the xampplite directory called apache\_stop.bat.

**CHAPTER 5**

**SOURCE CODE:**

######################frame1 button funcs###############

def addstudent():

def submitadd():

id = idval.get()

name = nameval.get()

mobile = mobileval.get()

email = emailval.get()

address = addressval.get()

gender = genderval.get()

dob = dobval.get()

addeddate = time.strftime("%d/%m/%Y")

addedtime = time.strftime("%H:%M:%S")

try:

strr = 'insert into studentdata2 values(%s,%s,%s,%s,%s,%s,%s,%s,%s)'

mycursor.execute(strr,(id,name,mobile,email,address,gender,dob,addeddate,addedtime))

con.commit()

res = messagebox.askyesnocancel('Notification','Id {} Name {} Added sucessfully.. and want to clean the form'.format(id,name),parent=addroot)

if(res == True):

idval.set('')

nameval.set('')

mobileval.set('')

emailval.set('')

addressval.set('')

genderval.set('')

dobval.set('')

except:

messagebox.showerror('Notifications','Id already exits try another id...',parent=addroot)

strr = 'select \* from studentdata2'

mycursor.execute(strr)

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0],i[1],i[2],i[3],i[4],i[5],i[6],i[7],i[8],]

studenttable.insert('',END,values=vv)

addroot = Toplevel(master=DataEntryFrame)

addroot.grab\_set()

addroot.geometry('470x470+220+200')

addroot.title('Student Managemnet System')

addroot.config(bg='crimson')

addroot.iconbitmap('mana.ico')

addroot.resizable(False,False)

#----------------------------- Add Student Labels------------

idlabel = Label(addroot,text='Enter Id:',bg='gold2',font=('times',20,'bold'),relief=GROOVE,borderwidth=3,width=12,anchor='w')

idlabel.place(x=10,y=10)

namelabel = Label(addroot, text='Enter Name:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

namelabel.place(x=10, y=70)

mobilelabel = Label(addroot, text='Enter Mobileno:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

mobilelabel.place(x=10, y=130)

emaillabel = Label(addroot, text='Enter Email:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

emaillabel.place(x=10, y=190)

addresslabel = Label(addroot, text='Enter Address:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

addresslabel.place(x=10, y=250)

genderlabel = Label(addroot, text='Enter Gender:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

genderlabel.place(x=10, y=310)

doblabel = Label(addroot, text='D.O.B:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

doblabel.place(x=10, y=370)

#----------------------------- Add Student Enrty-------------------

idval = StringVar()

nameval = StringVar()

mobileval = StringVar()

emailval = StringVar()

addressval = StringVar()

genderval = StringVar()

dobval = StringVar()

identry = Entry(addroot,font=('roman',15,'bold'),bd=5,textvariable=idval)

identry.place(x=250,y=10)

nameentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=nameval)

nameentry.place(x=250, y=70)

mobileentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=mobileval)

mobileentry.place(x=250, y=130)

emailentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=emailval)

emailentry.place(x=250, y=190)

addressentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=addressval)

addressentry.place(x=250, y=250)

genderentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=genderval)

genderentry.place(x=250, y=310)

dobentry = Entry(addroot, font=('roman', 15, 'bold'), bd=5, textvariable=dobval)

dobentry.place(x=250, y=370)

#---------------------- Add Button--------------

submitbtn = Button(addroot,text='Submit',font=('roman',15,'bold'),width=20,bd=5,activebackground='blue',activeforeground='White',bg='red',

command=submitadd)

submitbtn.place(x=150,y=420)

addroot.mainloop()

def searchstudent():

def search():

id = idval.get()

name = nameval.get()

mobile = mobileval.get()

email = emailval.get()

address = addressval.get()

gender = genderval.get()

dob = dobval.get()

addeddate = time.strftime("%d/%m/%Y")

if(id != ''):

strr = 'select \* from studentdata2 where id=%s'

mycursor.execute(strr,(id))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(name != ''):

strr = 'select \* from studentdata2 where name=%s'

mycursor.execute(strr,(name))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(mobile != ''):

strr = 'select \* from studentdata2 where mobile=%s'

mycursor.execute(strr,(mobile))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(email != ''):

strr = 'select \* from studentdata2 where email=%s'

mycursor.execute(strr,(email))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(address != ''):

strr = 'select \* from studentdata2 where address=%s'

mycursor.execute(strr,(address))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(gender != ''):

strr = 'select \* from studentdata2 where gender=%s'

mycursor.execute(strr,(gender))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(dob != ''):

strr = 'select \* from studentdata2 where dob=%s'

mycursor.execute(strr,(dob))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

elif(addeddate != ''):

strr = 'select \* from studentdata2 where addeddate=%s'

mycursor.execute(strr,(addeddate))

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

searchroot = Toplevel(master=DataEntryFrame)

searchroot.grab\_set()

searchroot.geometry('470x540+220+200')

searchroot.title('Student Managemnet System')

searchroot.config(bg='crimson')

searchroot.iconbitmap('mana.ico')

searchroot.resizable(False, False)

# ----------------------------- Search Student Labels------------

idlabel = Label(searchroot, text='Enter Id:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

idlabel.place(x=10, y=10)

namelabel = Label(searchroot, text='Enter Name:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

namelabel.place(x=10, y=70)

mobilelabel = Label(searchroot, text='Enter Mobileno:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

mobilelabel.place(x=10, y=130)

emaillabel = Label(searchroot, text='Enter Email:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

emaillabel.place(x=10, y=190)

addresslabel = Label(searchroot, text='Enter Address:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

addresslabel.place(x=10, y=250)

genderlabel = Label(searchroot, text='Enter Gender:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

genderlabel.place(x=10, y=310)

doblabel = Label(searchroot, text='D.O.B:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

doblabel.place(x=10, y=370)

datelabel = Label(searchroot, text='Enter Date:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

datelabel.place(x=10, y=430)

# ----------------------------- Add Student Enrty-------------------

idval = StringVar()

nameval = StringVar()

mobileval = StringVar()

emailval = StringVar()

addressval = StringVar()

genderval = StringVar()

dobval = StringVar()

dateval = StringVar()

identry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=idval)

identry.place(x=250, y=10)

nameentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=nameval)

nameentry.place(x=250, y=70)

mobileentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=mobileval)

mobileentry.place(x=250, y=130)

emailentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=emailval)

emailentry.place(x=250, y=190)

addressentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=addressval)

addressentry.place(x=250, y=250)

genderentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=genderval)

genderentry.place(x=250, y=310)

dobentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=dobval)

dobentry.place(x=250, y=370)

dateentry = Entry(searchroot, font=('roman', 15, 'bold'), bd=5, textvariable=dateval)

dateentry.place(x=250, y=430)

# ---------------------- Add Button--------------

submitbtn = Button(searchroot, text='Submit', font=('roman', 15, 'bold'), width=20, bd=5, activebackground='blue',

activeforeground='White', bg='red',

command=search)

submitbtn.place(x=150, y=480)

searchroot.mainloop()

def deletestudent():

cc = studenttable.focus()

content = studenttable.item(cc)

pp = content['values'][0]

strr = 'delete from studentdata2 where id=%s'

mycursor.execute(strr,(pp))

con.commit()

messagebox.showinfo('Notifications','Id {} deleted sucessfully....'.format(pp))

strr = 'select \* from studentdata2'

mycursor.execute(strr)

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

def updatestudent():

def update():

id = idval.get()

name = nameval.get()

mobile = mobileval.get()

email = emailval.get()

address = addressval.get()

gender = genderval.get()

dob = dobval.get()

date = dateval.get()

time = timeval.get()

strr = 'update studentdata2 set name=%s,mobile=%s,email=%s,address=%s,gender=%s,dob=%s,date=%s,' \

'time=%s where id=%s '

mycursor.execute(strr,(name,mobile,email,address,gender,dob,date,time,id))

con.commit()

messagebox.showinfo('Notifications', 'Id {} updated sucessfully....'.format(id),parent=updateroot)

strr = 'select \* from studentdata2'

mycursor.execute(strr)

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

updateroot = Toplevel(master=DataEntryFrame)

updateroot.grab\_set()

updateroot.geometry('470x610+220+150')

updateroot.title('Student Managemnet System')

updateroot.config(bg='crimson')

updateroot.iconbitmap('mana.ico')

updateroot.resizable(False, False)

# ----------------------------- Update Student Labels------------

idlabel = Label(updateroot, text='Update Id:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

idlabel.place(x=10, y=10)

namelabel = Label(updateroot, text='Enter Name:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

namelabel.place(x=10, y=70)

mobilelabel = Label(updateroot, text='Update Mobile:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

mobilelabel.place(x=10, y=130)

emaillabel = Label(updateroot, text='Update Email:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

emaillabel.place(x=10, y=190)

addresslabel = Label(updateroot, text='Update Address:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

addresslabel.place(x=10, y=250)

genderlabel = Label(updateroot, text='Update Gender:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

genderlabel.place(x=10, y=310)

doblabel = Label(updateroot, text=' UpdateD.O.B:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=12, anchor='w')

doblabel.place(x=10, y=370)

datelabel = Label(updateroot, text='Update Date:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

datelabel.place(x=10, y=430)

timelabel = Label(updateroot, text='Update Time:', bg='gold2', font=('times', 20, 'bold'), relief=GROOVE,

borderwidth=3,

width=12, anchor='w')

timelabel.place(x=10, y=490)

# ----------------------------- Add Student Enrty-------------------

idval = StringVar()

nameval = StringVar()

mobileval = StringVar()

emailval = StringVar()

addressval = StringVar()

genderval = StringVar()

dobval = StringVar()

dateval = StringVar()

timeval = StringVar()

identry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=idval)

identry.place(x=250, y=10)

nameentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=nameval)

nameentry.place(x=250, y=70)

mobileentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=mobileval)

mobileentry.place(x=250, y=130)

emailentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=emailval)

emailentry.place(x=250, y=190)

addressentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=addressval)

addressentry.place(x=250, y=250)

genderentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=genderval)

genderentry.place(x=250, y=310)

dobentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=dobval)

dobentry.place(x=250, y=370)

dateentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=dateval)

dateentry.place(x=250, y=430)

timeentry = Entry(updateroot, font=('roman', 15, 'bold'), bd=5, textvariable=timeval)

timeentry.place(x=250, y=490)

# ---------------------- Add Button--------------

submitbtn = Button(updateroot, text='Submit', font=('roman', 15, 'bold'), width=20, bd=5, activebackground='blue',

activeforeground='White', bg='red',

command=update)

submitbtn.place(x=150, y=540)

cc = studenttable.focus()

content = studenttable.item(cc)

pp = content['values']

if(len(pp) !=0):

idval.set(pp[0])

nameval.set(pp[1])

mobileval.set(pp[2])

emailval.set(pp[3])

addressval.set(pp[4])

genderval.set(pp[5])

dobval.set(pp[6])

dateval.set(pp[7])

timeval.set(pp[8])

updateroot.mainloop()

def showstudent():

strr = 'select \* from studentdata2'

mycursor.execute(strr)

datas = mycursor.fetchall()

studenttable.delete(\*studenttable.get\_children())

for i in datas:

vv = [i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], i[8], ]

studenttable.insert('', END, values=vv)

def exportstudent():

ff = filedialog.asksaveasfilename()

gg = studenttable.get\_children()

id,name,mobile,email,address,gender,dob,addeddate,addedtime=[],[],[],[],[],[],[],[],[]

for i in gg:

content = studenttable.item(i)

pp = content['values']

id.append(pp[0]),name.append(pp[1]),mobile.append(pp[2]),email.append(pp[3]),address.append(pp[4]),gender.append(pp[5]),

dob.append(pp[6]),addeddate.append(pp[7]),addedtime.append(pp[8])

dd = ['Id','Name','Mobile','Email','Address','Gender','D.O.B','Added Date','Added Time']

df = pandas.DataFrame(list(zip(id,name,mobile,email,address,gender,dob,addeddate,addedtime)),columns=dd)

paths = r'{}.csv'.format(ff)

df.to\_csv(paths,index=False)

messagebox.showinfo('Notifications', 'Student data is Saved {}'.format(paths))

def exitstudent():

res = messagebox.askyesnocancel('Notification','Do you want to Exit?')

if(res == True):

root.destroy()

##############################connection of database

def Connectdb():

def submitdb():

global con,mycursor

host = hostval.get()

user = userval.get()

password = passwordval.get()

try:

con = pymysql.connect(host=host,user=user,password=password)

mycursor = con.cursor()

except:

messagebox.showerror('Notification','Data is incorrect please try again')

return

try:

strr = 'create database studentmanagementsystem2'

mycursor.execute(strr)

strr = 'use studentmanagementsystem2'

mycursor.execute(strr)

strr = 'create table studentdata2(id int,name varchar(20),mobile varchar(12),email varchar(30),address varchar(100),gender varchar(50),dob varchar(50),date varchar(50),time varchar(50))'

mycursor.execute(strr)

strr = 'alter table studentdata2 modify column id int not null'

mycursor.execute(strr)

strr = 'alter table studentdata2 modify column id int primary key'

mycursor.execute(strr)

messagebox.showinfo('Notification', 'Database created and now you are connected to the database', parent=dbroot)

except:

strr = 'use studentmanagementsystem2'

mycursor.execute(strr)

messagebox.showinfo('Notification', 'Now you are connected to the database...',parent=dbroot)

dbroot.destroy()

dbroot = Toplevel()

dbroot.grab\_set()

dbroot.geometry('470x250+800+230')

dbroot.iconbitmap('mana.ico')

dbroot.resizable(False,False)

dbroot.config(bg='blue')

#------------------------connectdb labels----------#

hostlabel = Label(dbroot,text="Enter Host :",bg='gold2',font=('times',20,'bold'),relief=GROOVE,borderwidth=3,width=13,

anchor='w')

hostlabel.place(x=10,y=10)

userlabel = Label(dbroot, text="Enter User :", bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=13, anchor='w')

userlabel.place(x=10, y=70)

passwordlabel = Label(dbroot, text="Enter Password :", bg='gold2', font=('times', 20, 'bold'), relief=GROOVE, borderwidth=3,

width=13, anchor='w')

passwordlabel.place(x=10, y=130)

########################Connectdb Entry################

hostval = StringVar()

userval = StringVar()

passwordval = StringVar()

hostentry = Entry(dbroot,font=('roman',15,'bold'),bd=5,textvariable=hostval)

hostentry.place(x=240,y=10)

userentry = Entry(dbroot, font=('roman', 15, 'bold'), bd=5, textvariable=userval)

userentry.place(x=240, y=70)

passwordentry = Entry(dbroot, font=('roman', 15, 'bold'), bd=5, textvariable=passwordval)

passwordentry.place(x=240, y=130)

######################Connectdb Button################

submitbutton = Button(dbroot,text='Submit',font=('roman',15,'bold'),bg='red',bd=5,width=20,activebackground='blue',

activeforeground='White',command=submitdb)

submitbutton.place(x=150,y=190)

dbroot.mainloop()

###############################################

def tick():

time\_string = time.strftime("%H:%M:%S")

date\_string = time.strftime("%d/%m/%Y")

clock.config(text='Date :'+date\_string+"\n"+"Time : "+time\_string)

clock.after(200,tick)

#####################################Intro Slider

import random

colors = ['red','green','blue','yellow','pink','red2','gold2']

def IntrolabelColorTick():

fg = random.choice(colors)

SliderLabel.config(fg=fg)

SliderLabel.after(20,IntrolabelColorTick)

#############################################

def IntroLabelTick():

global count,text

if(count>=len(ss)):

count = 0

text = ''

SliderLabel.config(text=text)

else:

text =text+ss[count]

SliderLabel.config(text=text)

count += 1

SliderLabel.after(150,IntroLabelTick)

###############################################################

from tkinter import \*

from tkinter import Toplevel,messagebox,filedialog

from tkinter.ttk import Treeview

from tkinter import ttk

import pandas

import pymysql

import time

root = Tk()

root.title('Student Management System')

root.config(bg='gold2')

root.geometry('1174x700+200+50')

root.iconbitmap('mana.ico')

######### Frames #######################################

#####Frame1#######

DataEntryFrame = Frame(root,bg='gold2',relief=GROOVE,borderwidth=5)

DataEntryFrame.place(x=10,y=80,width=500,height=600)

#--------------------------------Frame Lables-------------#

frontlabel = Label(DataEntryFrame,text='----------------Welcome-----------',width=30,font=('arial',22,'italic bold'),bg='gold2')

frontlabel.pack(side=TOP,expand=True)

addbtn = Button(DataEntryFrame,text='1. Add Student',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=addstudent)

addbtn.pack(side=TOP,expand=True)

searchbtn = Button(DataEntryFrame,text='2. Search Student',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=searchstudent)

searchbtn.pack(side=TOP,expand=True)

deletebtn = Button(DataEntryFrame,text='3. Delete Student',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=deletestudent)

deletebtn.pack(side=TOP,expand=True)

updatebtn = Button(DataEntryFrame,text='4. Update Student',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=updatestudent)

updatebtn.pack(side=TOP,expand=True)

showallbtn = Button(DataEntryFrame,text='5. Show All ',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=showstudent)

showallbtn.pack(side=TOP,expand=True)

exportbtn = Button(DataEntryFrame,text='6. Export Data',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=exportstudent)

exportbtn.pack(side=TOP,expand=True)

exitbtn = Button(DataEntryFrame,text='7. Exit',width=17,font=('cjiller',20,'bold'),bd=6,bg='skyblue3',activebackground='blue',relief=RIDGE,

activeforeground='White',command=exitstudent)

exitbtn.pack(side=TOP,expand=True)

####Frame2#########

ShowDataFrame = Frame(root,bg='White',relief=GROOVE,borderwidth=5)

ShowDataFrame.place(x=550,y=80,width=620,height=600)

#--------------------------------------- Showdata Frame--------------

style = ttk.Style()

style.configure('Treeview.Heading',font=('chiller',17,'italic'),foreground='blue')

style.configure('Treeview',font=('times',13,'italic'),background='cyan',foreground='black')

scroll\_x = Scrollbar(ShowDataFrame,orient=HORIZONTAL)

scroll\_y = Scrollbar(ShowDataFrame,orient=VERTICAL)

studenttable = Treeview(ShowDataFrame,columns=('Id','Name','Mobile no','Email','Address','Gender','D.O.B','Added Date','Added Time'),

yscrollcommand=scroll\_y.set,xscrollcommand=scroll\_x.set)

scroll\_x.pack(side=BOTTOM,fill=X)

scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_x.config(command=studenttable.xview)

scroll\_y.config(command=studenttable.yview)

studenttable.heading('Id',text='Id')

studenttable.heading('Name',text='Name')

studenttable.heading('Mobile no',text='Mobile no')

studenttable.heading('Email',text='Email')

studenttable.heading('Address',text='Address')

studenttable.heading('Gender',text='Gender')

studenttable.heading('D.O.B',text='D.O.B')

studenttable.heading('Added Date',text='Added Date')

studenttable.heading('Added Time',text='Added Time')

studenttable['show'] = 'headings'

studenttable.column('Id',width=100)

studenttable.column('Name',width=200)

studenttable.column('Mobile no',width=200)

studenttable.column('Email',width=300)

studenttable.column('Address',width=400)

studenttable.column('Gender',width=100)

studenttable.column('D.O.B',width=150)

studenttable.column('Added Date',width=200)

studenttable.column('Added Time',width=200)

studenttable.pack(fill=BOTH,expand=1)

###########Slider######################################

ss = 'Welcome To Student Management System'

count = 0

text = ''

SliderLabel = Label(root,text=ss,font=('chiller',30,'italic bold'),relief=RIDGE,borderwidth=5,bg='cyan')

SliderLabel.place(x=150,y=0)

IntroLabelTick()

IntrolabelColorTick()

######################Clock########################

clock =Label(root,font=('times',14,'bold'),relief=RIDGE,borderwidth=4,bg='lawn green')

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

tick()

#######################Button###########################

connectbutton = Button(root,text='Connect to Database',width=22,height=2,font=('chiller',11,'italic bold'),relief=RIDGE,borderwidth=6,bg='green2',

activebackground='blue',activeforeground='White',command=Connectdb)

connectbutton.place(x=960,y=0)

root.mainloop()

**CHAPTER 6**

**FUTURE ENHANCEMENT:**

The Student Information Management System(SIMS) can be enhanced to include some other functionality like marks , attendance management.

♣ Talent management of students based on their performance evaluation can be added.

♣ Social networking can also be added wherein students can interact with each other.

♣ Online class functionality can be added.

♣ Can evolve as an online institution.

♣ Functionality of chat and messages can be added.

♣ Online exam functionality can be added.

♣ Online resume builder functionality can also be added.

**CONCLUSION:**

The student management system is useful for all schools and colleges to maintain the data of their students

This **system** helps in maintaining the information of pupil of the organization. It can be easily accessed by the manager and kept safe for a long period of time without any changes.

In future this could be enhanced in so many ways to keep their data safe and secure

