A REPORT ON

**STUDENT MANAGEMENT SYSTEM**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT

OF

DBMS (THIRD YEAR ENGINEERING)

##### SUBMITTED BY

STUDENT NAME:

7316 Atul Verma

7304 Abhisek Kumar Singh

7358 Umang Kumar

****

## DEPARTMENT OF COMPUTER ENGINEERING

## ARMY INSTITUTE OF TECHNOLOGY

DIGHI HILLS, ALANDI ROAD, PUNE 411015

## SAVITRIBAI PHULE PUNE UNIVERSITY

## 2022 - 2023

****

**CERTIFICATE**

This is to certify that the project report entitled

“**STUDENT MANAGEMENT SYSTEM**”

Submitted by

STUDENT NAME: Roll No:

is a Bonafede student at this institute and the work has been carried out by him under the supervision of Prof. Sharayu Lokhande and it is approved for the partial fulfillment of the requirement of, third year course on DBMS Mini Project of Savitribai Phule Pune University~~.~~

Prof. Sharayu Lokhande Prof. Dr S.R. Dhore

DBMS HOD

Department of Computer Engineering Department of Computer Engineering

Place: Pune

Date: 10 Nov 2022

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to my professors for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards seniors for their kind co-operation and encouragement which helped me in completion of this project.

My thanks and appreciations also go to my other colleague in developing the project and people who have willingly helped me out with their abilities.

I would like to express my special thanks of gratitude to my teacher as well as our principal who gave me the golden opportunity to do this wonderful project on the topic, which also helped me in doing a lot of Research and I came to know about so many new things.

**ABSTRACT**

Our focus is designing a unique Student Management System that will improve Data management in Institutes experience for both Students and the Administration authorities. The whole system will run on a single local network. The system is written in Python and its distributions. After that they will be able to various tasks that are designed for them. Moderators and database administrators can easily manage a whole class/institute without any difficulty. Student Management System is software which is helpful for students as well as the school authorities. In the current system all the activities are done manually. It is very time consuming and costly. Our Student Management System deals with the various activities related to the students.

The admin can do CRUD (Create a student record, Read a student’s data, Update a student’s bio, and finally Delete a students data). While the students have the rights to check their data and request an update if necessary. The design is simple, and the user won’t find it difficult to understand, use and navigate, due to Python’s tkinter module which creates an interactive and user-friendly GUI environment.

**TABLE OF CONTENTS**

# CHAPTER TITLE PAGE NO.

|  |  |  |
| --- | --- | --- |
| Sr. No. | Title of Chapter | Page No. |
| 01 | Introduction | 6 |
| 02 | Features provided to various users | 7 |
| 03 | Literature Survey | 8-10 |
| 04 | Analysis and Requirement Specification | 11-12 |
| 05 | Design | 13 |
| 06 | Proposed System | 14 |
| 07 | Implementation | 15-19 |
| 08 | Snapshots | 20-23 |
| 09 | Conclusion | 24 |

**INTRODUCTION**

A database management system (DBMS) refers to the technology for creating and managing databases. Basically, DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. This project aims at computerizing the manual process of management system. Along with the Py mongo program to analyze the program. The project consists of seven forms(entity) namely Planner who will plan for the management. The form Planner will have number of staffs. As well as each staff will maintains the guests and staffs are belongs to department.

**Features provided to various users**

The services of a management system can include:

•Adding the information

•Delete the entry

•Update the entry

•Update the entry

**LITERATURE SURVEY**

**2.1 Introduction to Database Management System:**

DBMS stands for Database Management System. We can break it like this DBMS=Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store and access those data in an easy and effective manner. Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data. According to the principles of database systems, the data is stored in such a way that it acquires a lot less space as the redundant data (duplicate data) has been removed before storage. Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database system ensures that data is retrieved as quickly as possible.

* **Applications of DBMS**

The development of computer graphics has been driven both by the needs of the user community and by the advances in hardware and software. The applications of database are many and varied; it can be divided into four major areas:

* + Hierarchical and network system
  + Flexibility with relational database
  + Object oriented application.
  + Interchanging the data on the web for e-commerce
* **Display information**

In this project, we are taken HTML web page as a front end to display the information which are stored in the backend database called MongoDB.

Professions such as engineering and architecture are concerned with design. Starting with a set of specification engineers and architects seek a cost effective and esthetic solutions that satisfies the specifications. Design is an iterative process rarely in the real world is a problem specified such that there is a unique optimal solution. Thus, the designer works iteratively.

* **User Interfaces**

Our interactions with computers have become dominated by a visual paradigm that includes windows, icons, menus, pointing device, such as a mouse. Although we are familiar with the syntax of MongoDB, advances in MongoDB have made possible other forms of advantages.

* **What is MongoDB?**
  + MongoDB is an open-source document database built on a horizontal scale-out architecture that uses a flexible schema for storing data. Founded in 2007, MongoDB has a worldwide following in the developer community.
  + Instead of storing data in tables of rows or columns like [SQL databases](https://www.mongodb.com/nosql-explained/nosql-vs-sql), each record in a MongoDB database is a document described in BSON, a binary representation of the data. Applications can then retrieve this information in a JSON format.
  + Document databases are highly flexible, allowing variations in the structure of documents and storing documents that are partially complete. One document can have others embedded in it. Fields in a document play the role of columns in a SQL database, and like columns, they can be indexed to increase search performance.
  + MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need. MongoDB’s document model is simple for developers to learn and use, while still providing all the capabilities needed to meet the most complex requirements at any scale. We provide drivers for 10+ languages, and the community has built dozens more.
    - MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
    - MongoDB is free to use. Versions released prior to October 16, 2018, are published under the AGPL. All versions released after October 16, 2018, including patch fixes for prior versions, are published under the Server-Side Public License (SSPL) v1.
* **MongoDB Command Syntax**

MongoDB Mongo shell is an interactive JavaScript interface that allows you to interact with MongoDB instances through the command line. The shell can be used for:

1. Data manipulation
2. Administrative operations such as maintenance of database instances
3. Here are the top features that Mongo shell offers:
4. Run all MongoDB queries from the Mongo shell.
5. Manipulate data and perform administration operations.
6. Mongo shell uses JavaScript and a related API to issue commands.
7. See previous commands in the mongo shell with up and down arrow keys.
8. View possible command completions using the tab button after partially entering a command.
9. Print error messages, so you know what went wrong with your commands.

* **MongoDB-related Libraries for Python**

You must install the Py Mongo driver module to make it available to your Python application. We recommend using [pip](http://pypi.python.org/pypi/pip) to install Py Mongo.

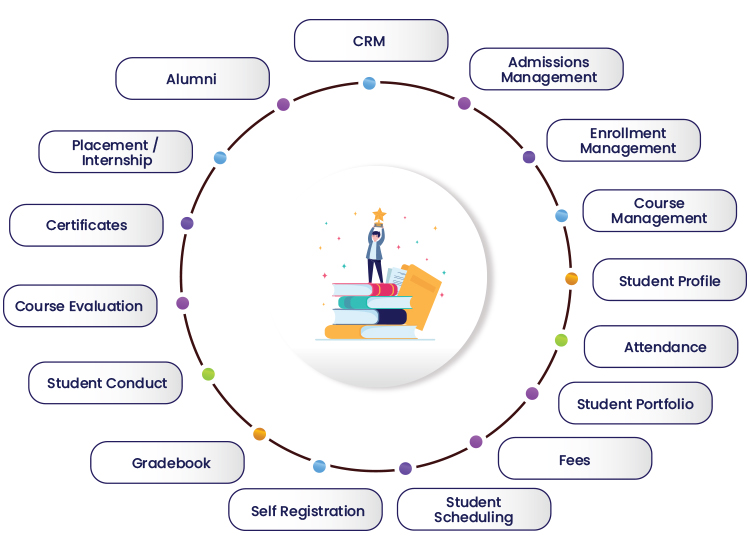
The following command demonstrates how you can install the latest version of the module using the command line:

python -m pip install pymongo

**ANALYSIS & REQUIREMENT SPECIFICATION**

**3.1 Purpose**

A Student Management System is also known as a Student Information System (SIS). These systems work to coordinate scheduling and communications between faculty regarding students. This system exists to simplify information tracking for both parents and administrative staff.



The main objective at the end of the day for any school is the dissemination of information in an environment that encourages accountability, relevance and retention of the information gathered. Such are the main objectives of a school administrator, who, as a student manager, is charged with allowing this opportunity and environment to occur on behalf of the students.

**3.2 Scope**

The scope of the project is managing consistency and storage of data by dedicated data administrator. It provides most of the features that a Database Management System should have. It is developed by using MongoDB database. It has been implemented in WINDOWS platform.

**3.3 Functional Requirements:**

Three modules are used in this project namely Admin, user, planner

* Admin: can insert, analyze the table’s
* User: can register their information
* Student: can update the profile

**3.4 Non-Functional Requirements:**

* **Hardware specification**
  + Processor: i5 Core Processor
  + Clock speed: 2.5GHz
  + Monitor: 1024 \* 768 Resolution Color
  + Keyboard: QWERTY
  + RAM: 1 GB

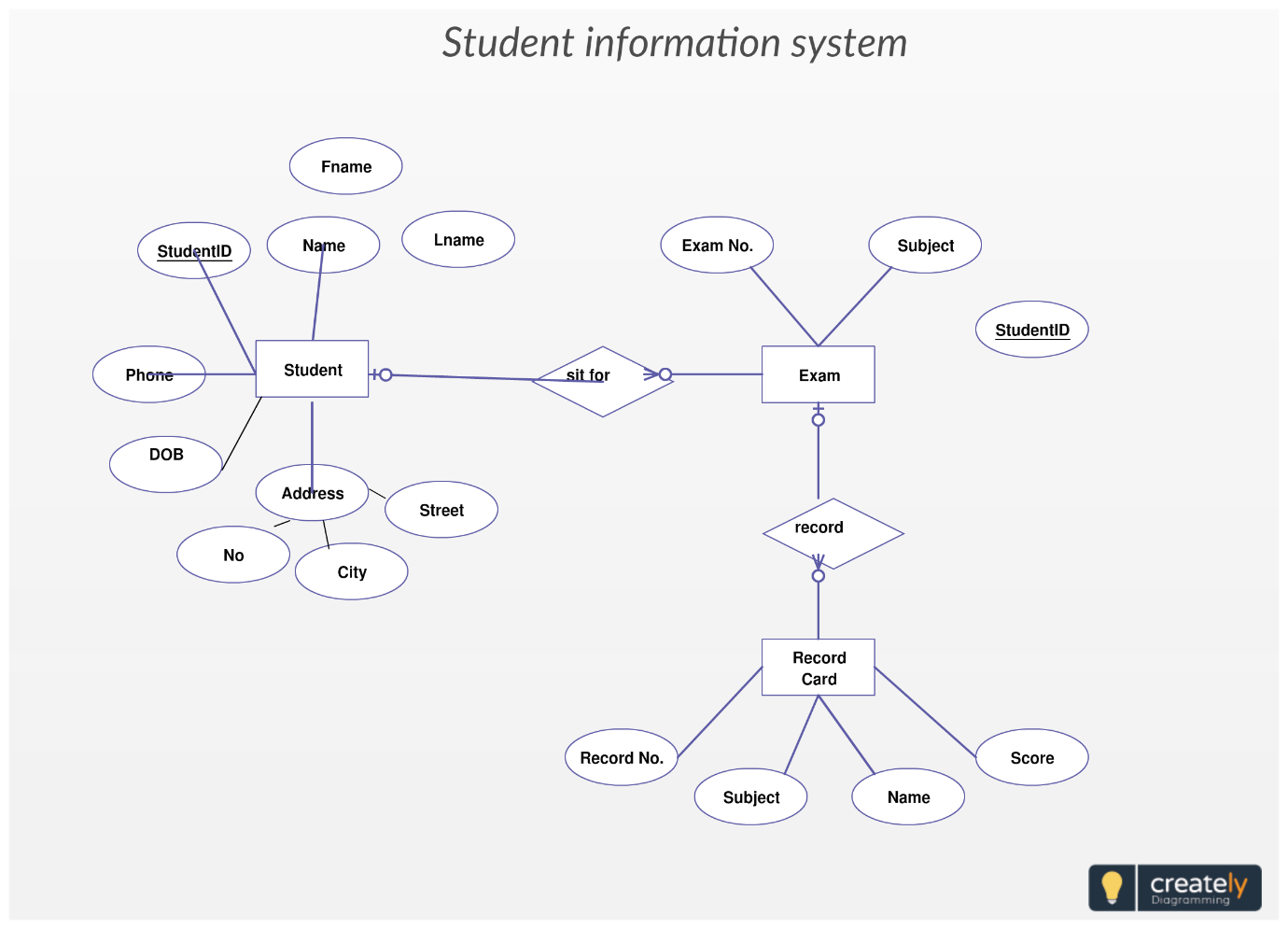
Input Output Console for interaction

* **Software specification**
  + MongoDB LibrariesPython3, (pymongo Libraries)
  + MongoDB Workbench 6.3 CE
  + PycharmPyCharm IDE
  + Operating system: Windows 11

**DESIGN OF THE PROJECT**

This project has been developed using MongoDB software which is queries oriented. Changes set the queries and the way in which it uses a system state may cause anticipated changes in the behavior of other result.

**4.1 ER Diagram:**

****

**PROPOSED SYSTEM**

There will be three major components:

1. Create Student Registration
2. Storing Students DataRead Student Data
3. Student information updateUpdate Student Data
4. Delete Student Record

Proposed system provides the following solution

1. It provides better and more efficient service to admin.
2. Admin can easily add new studentsstudents and manage them.
3. Faster search for user.
4. Easy updating service provided to admin and admin can modify its profile.any time.
5. All details about an admin and the studentsstudent will be easy to navigate via GUI.

Admin’s Role

1. Making correct entry about the new student.
2. Take care of which students need to be updated.

User’s Role

1. Updating a student information.
2. Update and delete the student’s information.Query their data.
3. Update their data.

**IMPLEMENTATION**

from \_\_future\_\_ import unicode\_literals

from pymongo import MongoClient

from random import randint

from tkinter import \*

import tkinter.messagebox

import tkinter.ttk

from tkinter import Button

import sys

try:

client = MongoClient(port=27017)

db = client.Assignment08

print("Connected to MongoDB")

except:

print("Database connection Error ")

print("No connection could be made because the target machine actively refused it ")

tkinter.messagebox.showerror("Error", "Connection Error")

sys.exit(1)

root = Tk()

root.geometry('400x350')

root.title("Student Management System")

def add\_STUDENTS(root, db):

def add\_query():

global root

prn = E1.get()

name = E2.get()

email = E3.get()

batch = E4.get()

mobile = E5.get()

PRN = [prn]

NAME = [name]

EMAIL = [email]

BATCH = [batch]

MOBILE = [mobile]

Assignment08 = {

'PRN': PRN[randint(0, (len(PRN) - 1))],

'NAME': NAME[randint(0, (len(NAME) - 1))],

'EMAIL': EMAIL[randint(0, (len(EMAIL) - 1))],

'BATCH': BATCH[randint(0, (len(BATCH) - 1))],

'MOBILE': MOBILE[randint(0, (len(MOBILE) - 1))]}

if (len(prn) == 0):

tkinter.messagebox.showwarning("WARNING", "All fields are compulsory(Except: Mobile number)")

return

if (len(name) == 0):

tkinter.messagebox.showwarning("WARNING", "All fields are compulsory(Except: Mobile number)")

return

if (len(email) == 0):

tkinter.messagebox.showwarning("WARNING", "All fields are compulsory(Except: Mobile number)")

return

if (len(batch) == 0):

tkinter.messagebox.showwarning("WARNING", "All fields are compulsory(Except: Mobile number)")

return

if len(mobile) == 0 and db.students.count\_documents({'PRN': prn}, limit=1) == 0:

result = db.students.insert\_one({'PRN': prn, 'NAME': name, 'EMAIL': email, 'BATCH': batch})

elif len(mobile) != 0 and db.students.count\_documents({'PRN': prn}, limit=1) == 0:

result = db.students.insert\_one(Assignment08)

else:

tkinter.messagebox.showwarning("ERROR", "STUDENT Already Exists")

return

newwin.destroy()

tkinter.messagebox.showinfo("Add Student", "Student Added")

newwin = Toplevel(root)

newwin.geometry('400x400')

newwin.title("Add STUDENTS")

L1 = Label(newwin, text="PRN")

L1.place(x=10, y=50)

E1 = Entry(newwin, bd=7)

E1.place(x=100, y=50)

L2 = Label(newwin, text="NAME")

L2.place(x=10, y=100)

E2 = Entry(newwin, bd=7)

E2.place(x=100, y=100)

L3 = Label(newwin, text="EMAIL")

L3.place(x=10, y=150)

E3 = Entry(newwin, bd=7)

E3.place(x=100, y=150)

L4 = Label(newwin, text="BATCH")

L4.place(x=10, y=200)

E4 = Entry(newwin, bd=7)

E4.place(x=100, y=200)

L5 = Label(newwin, text="MOBILE")

L5.place(x=10, y=250)

E5 = Entry(newwin, bd=7)

E5.place(x=100, y=250)

sub = Button(newwin, text="Submit", command=add\_query)

sub.place(x=120, y=350)

def del\_data(root, db):

def delete():

global root

prn = E1.get()

if (len(prn) == 0):

tkinter.messagebox.showwarning("WARNING", "Enter a Valid PRN")

return

if db.students.count\_documents({'PRN': prn}, limit=1) == 0:

tkinter.messagebox.showwarning("ERROR", "STUDENT Does Not Exist")

return

else:

db.students.delete\_one({'PRN': prn})

newwin.destroy()

tkinter.messagebox.showinfo("Delete Student", "Student Deleted")

newwin = Toplevel(root)

newwin.geometry('400x350')

newwin.title("Delete STUDENT")

L1 = Label(newwin, text="PRN")

L1.place(x=10, y=50)

E1 = Entry(newwin, bd=5)

E1.place(x=100, y=50)

sub = Button(newwin, text="Delete Entry", command=delete)

sub.place(x=120, y=200)

def update\_data(root, db):

def UPDD():

global root

prn = E6.get()

name = E7.get()

email = E8.get()

batch = E9.get()

mobile = E10.get()

if (len(prn) == 0):

tkinter.messagebox.showwarning("WARNING", "Enter a Valid PRN")

return

if db.students.count\_documents({'PRN': prn}, limit=1) == 0:

tkinter.messagebox.showwarning("ERROR", "STUDENT Does Not Exist")

return

if (len(name) != 0):

db.students.update\_one({"PRN": prn}, {"$set": {'NAME': name}})

if (len(email) != 0):

db.students.update\_one({"PRN": prn}, {"$set": {'EMAIL': email}})

if (len(batch) != 0):

db.students.update\_one({"PRN": prn}, {"$set": {'BATCH': batch}})

if (len(mobile) != 0):

db.students.update\_one({"PRN": prn}, {"$set": {'MOBILE': mobile}})

newwin.destroy()

tkinter.messagebox.showinfo("Update Student", "Student Updated")

newwin = Toplevel(root)

newwin.geometry('400x400')

newwin.title("Update STUDENTS")

L6 = Label(newwin, text="PRN")

L6.place(x=10, y=50)

E6 = Entry(newwin, bd=7)

E6.place(x=100, y=50)

L7 = Label(newwin, text="NAME")

L7.place(x=10, y=100)

E7 = Entry(newwin, bd=7)

E7.place(x=100, y=100)

L8 = Label(newwin, text="EMAIL")

L8.place(x=10, y=150)

E8 = Entry(newwin, bd=7)

E8.place(x=100, y=150)

L9 = Label(newwin, text="BATCH")

L9.place(x=10, y=200)

E9 = Entry(newwin, bd=7)

E9.place(x=100, y=200)

L10 = Label(newwin, text="MOBILE")

L10.place(x=10, y=250)

E10 = Entry(newwin, bd=7)

E10.place(x=100, y=250)

sub = Button(newwin, text="Submit", command=UPDD)

sub.place(x=120, y=350)

def display(root, db):

newwin = Toplevel(root)

newwin.geometry('400x400')

newwin.title("STUDENT Details")

L1 = Label(newwin, text="PRN")

L1.grid(row=0, column=0)

L2 = Label(newwin, text="NAME")

L2.grid(row=0, column=2)

L3 = Label(newwin, text="EMAIL")

L3.grid(row=0, column=4)

L4 = Label(newwin, text="BATCH")

L4.grid(row=0, column=6)

L5 = Label(newwin, text="MOBILE")

L5.grid(row=0, column=8)

i = 1

for x in db.students.find():

y = len(x)

# print(len(x))

L1 = Label(newwin, text=x['PRN'])

L1.grid(row=i, column=0)

L2 = Label(newwin, text=x['NAME'])

L2.grid(row=i, column=2)

L3 = Label(newwin, text=x['EMAIL'])

L3.grid(row=i, column=4)

L4 = Label(newwin, text=x['BATCH'])

L4.grid(row=i, column=6)

if y == 6:

L5 = Label(newwin, text=x['MOBILE'])

L5.grid(row=i, column=8)

i += 1

add = Button(root, text='Add New STUDENTS', command=lambda: add\_STUDENTS(root, db))

delete = Button(root, text='Delete STUDENTS Entry', command=lambda: del\_data(root, db))

update = Button(root, text='Update STUDENTS Info', command=lambda: update\_data(root, db))

show = Button(root, text='Show STUDENTS Details', command=lambda: display(root, db))

add.place(x=100, y=100)

delete.place(x=100, y=150)

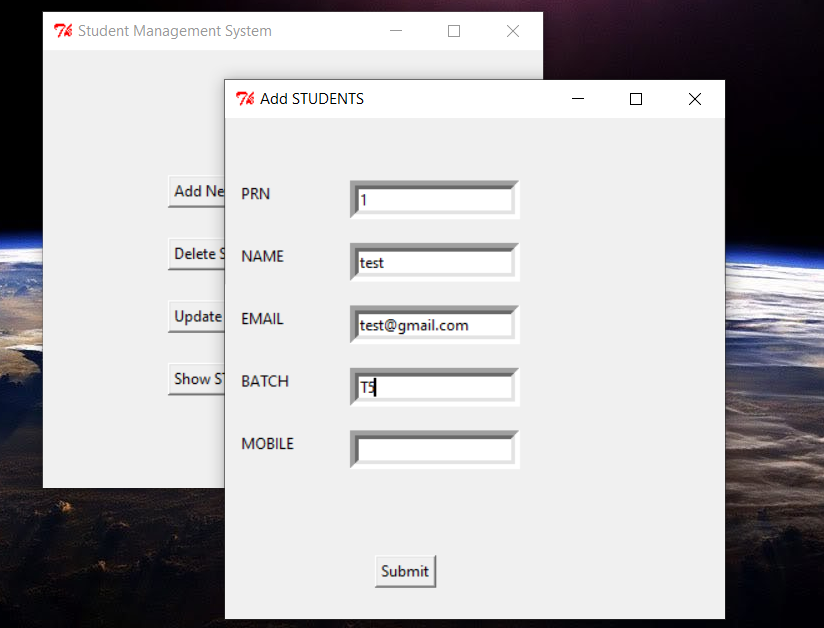
update.place(x=100, y=200)

show.place(x=100, y=250)

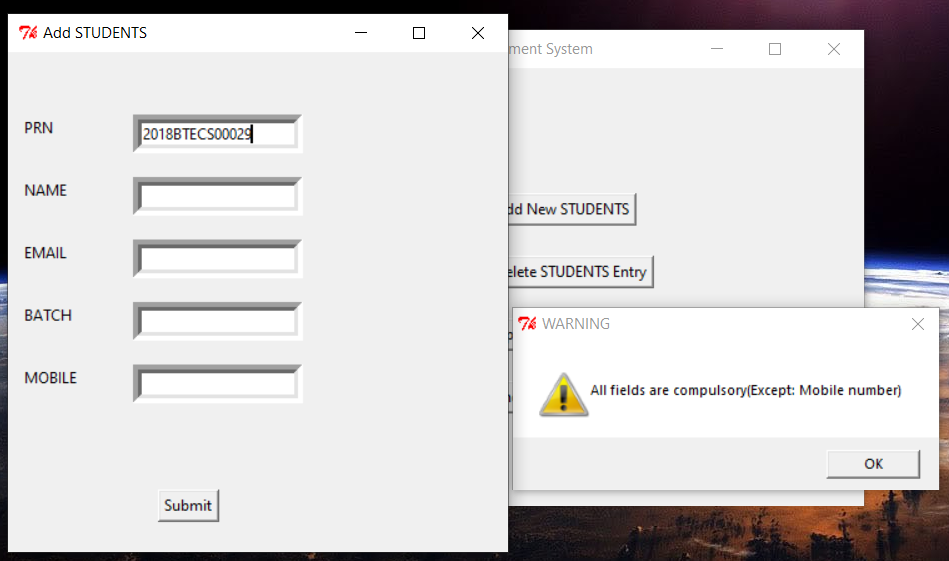
root.mainloop()

**SNAPSHOTS**

* **Home Page**

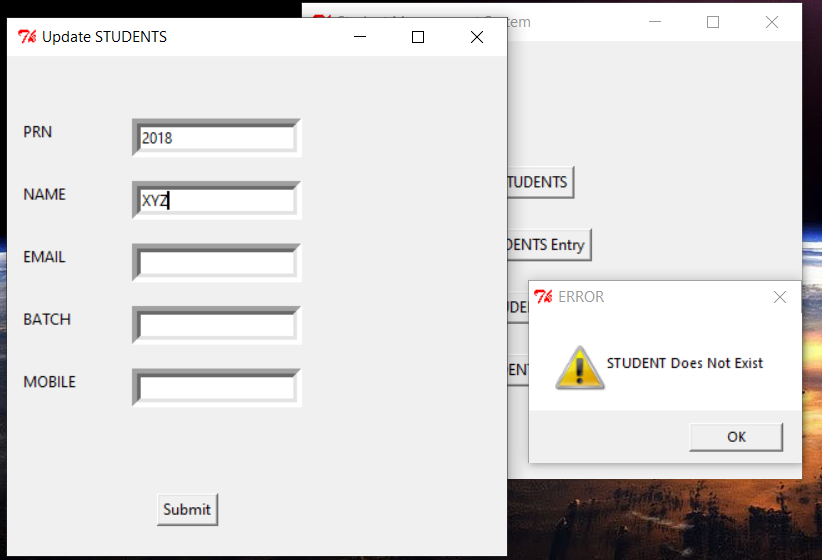
****

* **Add Students**

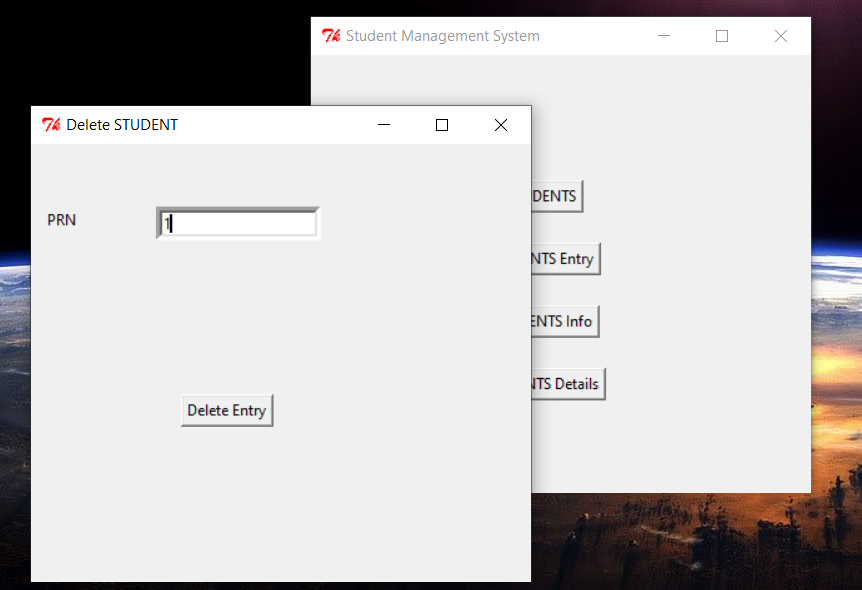
****

****

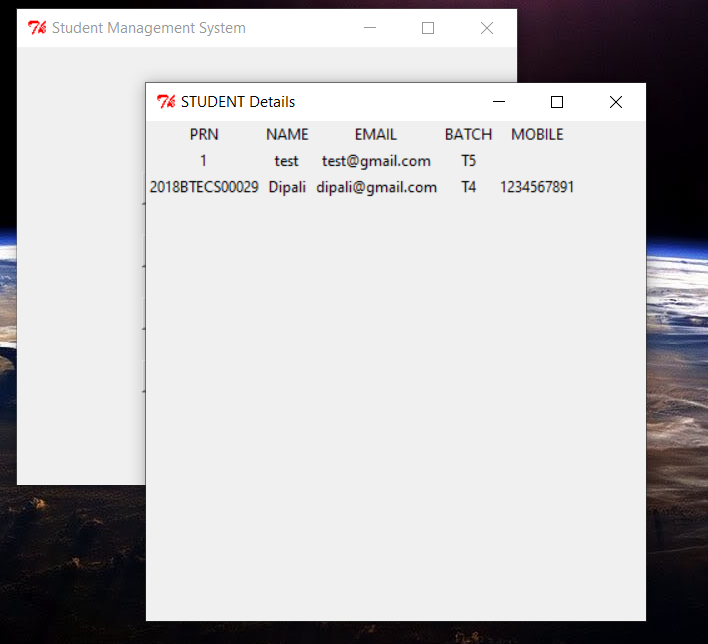
* **Search Students**

****

* **Delete Student Info**

****

* **Display Student Info**

****

**CONCLUSION**

We have implemented the student management system successfully. We used python’s tkinter as a front-end tool, MongoDB for database server and Mongo DB as backend tool.

The system is very good at saving precious time of admin as well as storing student information easily.