International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Student Management system

Mrs. Selina Khoirom[1], Laimujam Raj Singh[2], Sougaijam Nealson Singh[3], Laishram Rabi Singh[4], Keisham Linthoingambi[5]

[1]Assistant Professor, Dept. of Computer Science & Engineering, Manipur Institute of Technology, Manipur-795004, India
[2-5]B.E. Student, Dept. of Computer Science & Engineering, Manipur Institute of Technology, Manipur-795004, India

---

ABSTRACT: Our project Student Management system includes 'computerized the process. Our software has the facility to give a unique id for every student and stores the details of every number. The data can be retrieved easily. The interface is very User-friendly. The data are well protected for personal use and makes the data processing very fast.

***Key Words*: (Student, Management, System, Python Language, User friendly)**

1. INTRODUCTION

This project is aimed to automate the student management system. This project is developed mainly to administrate the student records. The purpose of the project entitled as to computerize the Front Office Management of student records in colleges, schools and coaching's, to develop software which is user friendly, simple, fast and cost-effective. Traditionally, it was done manually .The main function of the system is to register and store student details, retrieve and these details as and when required, and also to manipulate these details meaningfully.

2. SCOPE

The proposed software product is the student Management system. The system will be used in **any** School, College and coaching institute to get the information from the student and then storing that data for future usage.

The current system in use is a paper-based system. It is too slow and cannot provide update lists of students within a reasonable timeframe. The intentions of the system are to reduce over-time pay and increase the productivity. Requirements statements in this document are both functional and non-functional.

3. SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Pentium IV Processor or higher
- 2 GB RAM and above
- 40 GB hard disk
- Mouse/Keyboard

3.2 SOFTWARE REQUIREMENTS

- OS-Windows 7/8/10
- Python Interpreter
- Pycharm IDE

4. TECHNICAL DESCRIPTION

4.1 FRONT END DESCRIPTION

LANGUAGE: Python

Python is widely used general-purpose, high level programming language. It was initially design by Guido van Rossum in 1991 and developed by Python software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer line of code.

Python is a programming language that let you work quickly and integrate system more efficiently.

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

Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkits and is Python's de facto standard GUI. Tkinter is included in standard Linux, Microsoft Window and Mac OS X installs of Python.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.

5. SOFTWARE ANALYSIS AND DESIGN

5.1 SOFTWARE ANALYSIS

5.1.1 SOFTWARE DEVELOPMENT LIFE CYCLE

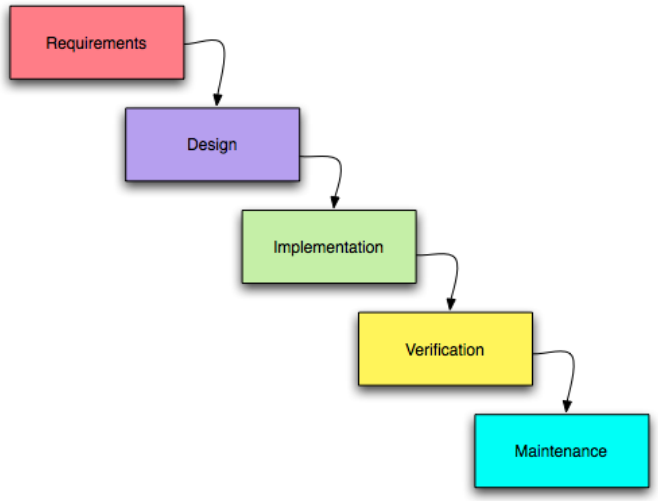

Fig: SDLC

5.1.2 DESCRIPTION OF USED MODEL

The Waterfall Model

The Waterfall Model is a sequential software development process, in which progress is seen as flowing steadily downward (like a waterfall) through the phases of Conception, Initiation, Analysis, design (Validation), Construction, Testing and Maintenance.

To follow the waterfall model, one proceeds from one phase to the next in a sequential manner. For example, one first completes requirement specification, which after signoff are considered "set in stone." When the requirements are fully completed, one proceeds to design. The software in question is designed and a blueprint is drawn for implementers (coders) to follow – this design should be a plan for implementing the requirements given. When the design is fully completed, an implementation of that design is made by coders. Towards the later stages of this implementations phase, separate software components produced are combined to introduce new functionality reduced risk through the removal of errors.

Thus the waterfall model maintains that one should move to a phase only when its preceding phase is completed and perfected. However, there are various modified waterfall models (including Royce's final model) that may include slight or major variations upon this process. Time spent early in the software production cycle can lead to greater economy at later stages. It has been shown that a bug found in the early stages (such as requirements specifications or design) is cheaper in terms of money, effort and time, to fix the same bug found later on in the process. To take an extreme example, if a program design turns out to be impossible to implement, it is easier to fix the design at stage than to realize months later, when program components are being integrated, that all the work one so far has to be scrapped because of a broken design.

This is the central idea behind the waterfall model – time spent early on making sure that requirements and design are absolutely correct will save you much time and effort later. Thus, the thinking of those who follow waterfall process goes, one should make sure that each phase is 100% complete and absolutely correct before proceeding to the next phase of program creation. Program requirements should be set in stone before design is started (otherwise work put into a design based on incorrect requirements is wasted); the program's design should be perfect before people begin work on implementing the design (otherwise they are implementing the wrong design and their work is wasted)' etc.

A further argument for the waterfall model is that it places emphasis on documentation (such as requirements documents and design documents) as well as source code. In less designed and documented methodologies, should team members leave, much knowledge is not lost and may be difficult for a project to recover from. Should a fully working design document be present (as is the intent of Big Design Up Front and the waterfall model) new team members or even entirely new teams should be able to familiarize themselves by reading the documents.

Basic principles of the waterfall model are:

Project is divided into sequential phases, with some overlap and splash back acceptable between phases.

Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

5.2 SOFTWARE DESIGN

5.2.1 Data Flow Diagram

The DFD takes an input-process-output view of a system. That is, data objects flow into the software, are transformed by the processing elements, and resultant data objects flow out of the software. Data object are represented by circles. The DFD is represented by labeled arrows and transformations are represented in a hierarchical fashion. The first DFD represents the system as a whole. Subsequent data flow diagrams provide increasing detail with each subsequent data flow diagrams provide increasing detail with each subsequent level.

The data flow diagram enables the software engineer to develop modes of the information domain at the same time. As the DFD is refined into levels of greater detail, the analysis perform an implicit functional decomposition of the system. Also DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the applications.

6. IMPLEMENTATION

6.1 WELCOME PAGE

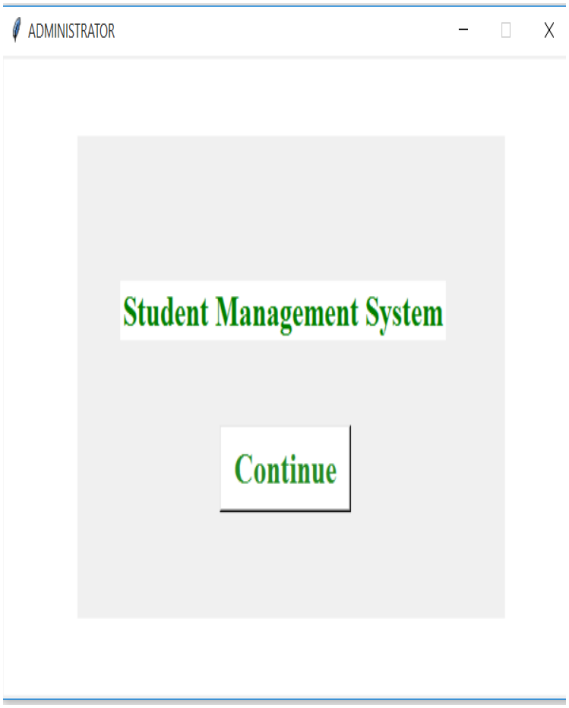

Fig 1

This is the front page, we can see this page after launched the application. By using Tkinter, we are creating Graphical User Interfaces in Python.

Here, we are creating a label widgets 'Student Management System' and a button 'Continue' by using Tkinter.

6.2 Admin page

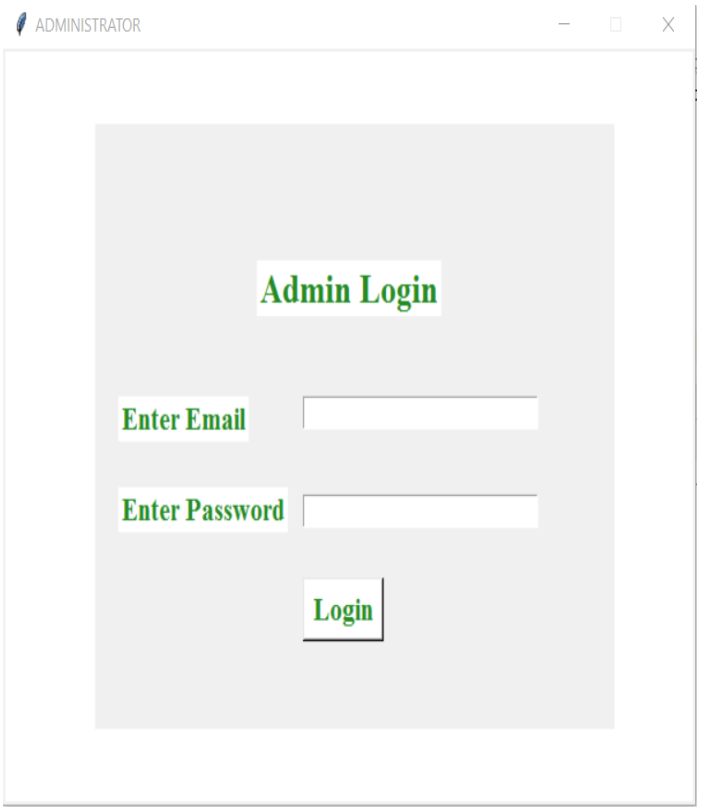

Fig 2

We have a Login page in our Student Management System.

There are three Label Widgets i.e. Admin Login, Enter Email, Enter Password and two entry Widgets and a button Widgets 'Login'.

In this page, we can login to our Student Management System by entering the email and password.

Login source code:

From tkinter import *from tkinter import message box import pymysql class LoginWindow: def __init__(self,*args): self.win = Tk()

reset the window and background color self.canvas = Canvas(self.win, width=600, height=500, bg='white') self.canvas.pack(expand=YES, fill=BOTH)

```
# show window in center of the screen
width = self.win.winfo_screenwidth()
height = self.win.winfo_screenheight()
x = int(width / 2 - 600 / 2) y = int(height / 2 - 500 / 2)
str1 = "600x500+" + str(x) + "+" + str(y)
self.win.geometry(str1)

# disable resize of the window
self.win.resizable(width=False, height=False)


# change the title of the window
self.win.title("ADMINISTRATOR") def add_frame(self):
self.frame = Frame(self.win, height=400, width=450)
self.frame.place(x=80, y=50) x, y = 70, 20 # now create a login form self.label = Label(self.frame, text="Admin Login", g="white", fg="forestgreen", font=("times new roman", 20, "bold"))

self.label.place(x=140, y=90) self.emlabel = Label(self.frame, text="Enter Email", bg="white", fg="forestgreen", font=("times new roman", 16, "bold"))
self.emlabel.place(x=20, y=180) self.email = Entry(self.frame, font='Courier 12')
self.email.place(x=180, y=180) self.pslabel = Label(self.frame, text="Enter Password", bg="white", fg="forestgreen",
font=("times new roman", 16, "bold"))
self.pslabel.place(x=20, y=240)

self.password = Entry(self.frame, show='*', font='Courier 2')
self.password.place(x=180, y=245)self.button = Button(self.frame, text="Login", bg="white", g="forestgreen",
font=("times new roman", 16, "bold"), command=self.login_admin) self.button.place(x=180, y=300)

self.win.mainloop()

def login_admin(self): data = (self.email.get(), self.password.get())
if self.email.get() == "": messagebox.showinfo("Alert!", "Enter Email First")
elif self.password.get() == "": messagebox.showinfo("Alert!", "Enter Password first") else:
con = pymysql.connect(host="localhost", user="root", password="raj8521", database="login")
cursor = con.cursor()

def user_login(tup): try:cursor.execute("SELECT * FROM `admin` WHERE email`=%s AND `password`=%s", tup) return (cursor.fetchone()) except: return False res =user_login(data) if res: messagebox.showinfo("Message", "Login Successfully")self.win.destroy() import student
student.Student()else:
messagebox.showinfo("ALert!", "Wrong username/password")

if __name__ == "__main__":
x = LoginWindow()
```

6.3 Manage Students and search student page

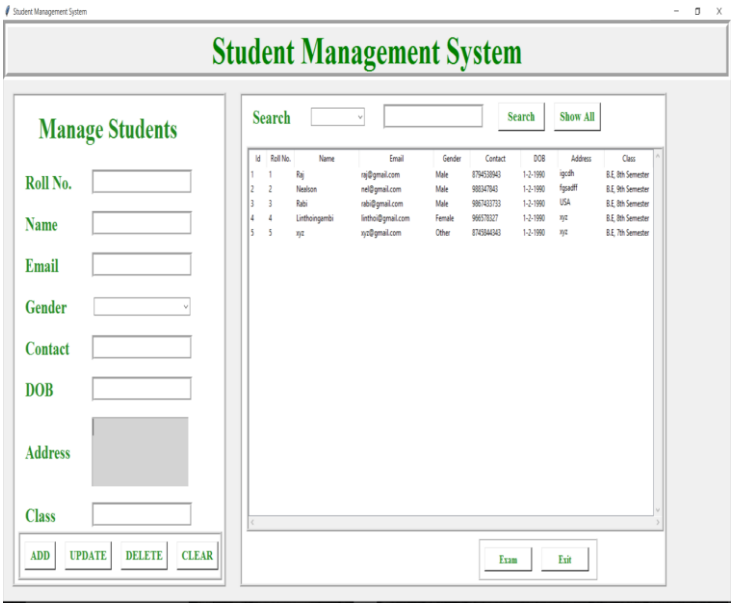

Fig 3

After logging in, we can get our Student Management System page.

Here, we have multiple number of label widgets i.e. Student Management System, Manage Students, Roll No., Name, Email, Gender, Contact, DOB, Address, Class, Search.

And we also have 8 button widgets, 2 combo box widgets and 8 entry widgets.

Manage Student source code:

```
def add_students(self): if(self.Roll_No_var.get()=="" or self.name_var.get()==""): messagebox.showerror("Error")

else:
con =pymysql.connect(host="localhost", user="root", password="raj8521", database="student") cur = con.cursor()
cur.execute("insert into students values(%s,%s,%s,%s,%s,%s,%s,%s,%s)",
(
self.id_var.set(1),
self.Roll_No_var.get(),
self.name_var.get(),
self.email_var.get(),
self.gender_var.get(),
self.contact_var.get(),
self.dob_var.get(),
self.txt_address.get('1.0', END),
self.class_var.get()
```

```
))
con.commit()
self.fetch_data()
self.clear()
con.close()
messagebox.showinfo("Success")
```

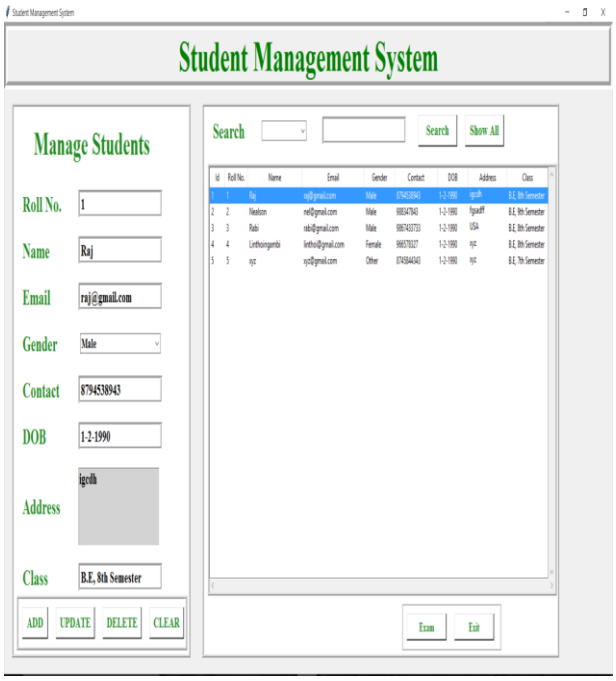

Fig4

Now, we can manage the students by simply typing the student details in the manage students. In this search label widgets, we have a combo box with 5 label widgets i.e. Id, Roll No., Name, Contact, Class.

We can select any one of our choice and search the student the student using the search button widget.

We also have Show All button widgets to show all the records of students.

We have manage exam page. In this page we manage the exam result of the student.

This page is created by using 12 label widgets, 9 entry widgets, 8 buttons and a combo box.

This combo box has 5 labels i.e. Id, Roll no., Name Class, Result.

Database

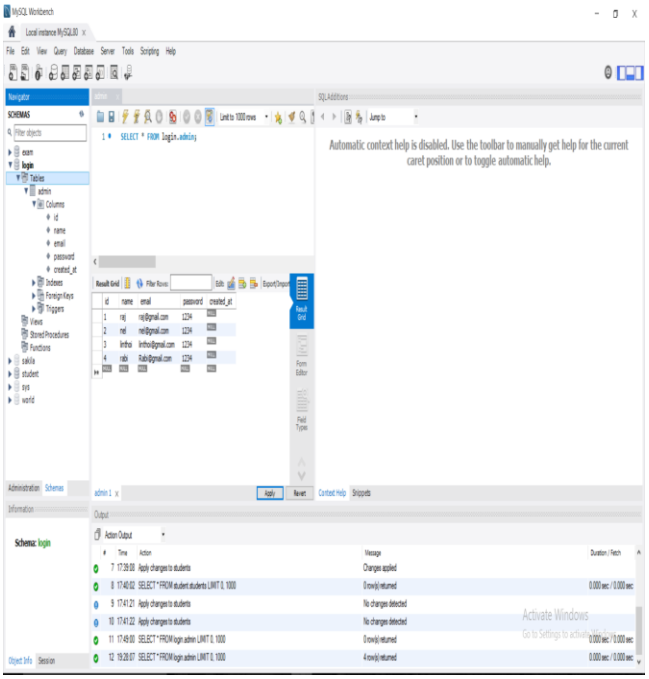

Fig 5

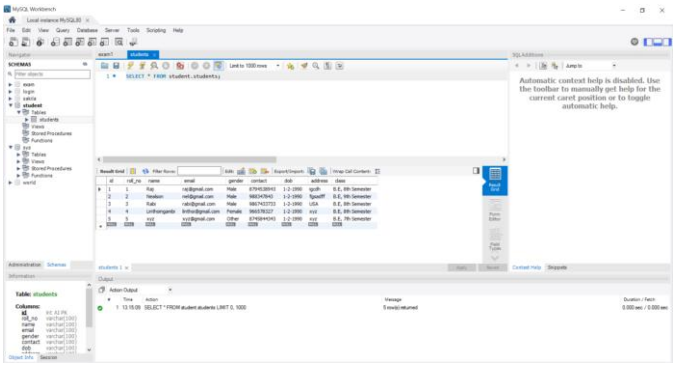

Fig 6

This is MYSQL database. It is the database that we use for creating our project 'Student Management System'.

7. TESTING

Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

Software Testing

Software testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capacity of a program or a system and determining that it meets its required result. Software is not unlike other physical processes where inputs are received and outputs are

produced. Where software differs is in the manner in which it fails. Unlike most physical systems, most of the defects in software are design errors, not manufacturing defects.

To improve quality

As computers and software are used in critical applications, the outcome of a bug can be severe. Bugs can cause huge losses.

For Verification & Validation (V&A).

It is heavily used as a tool in the V&V process. Testers can make claims based on interpretations of the testing result, which either the product works under certain situations, or it does not work.

SOFTWARE TESTING TYPE

BLACKBOX TESTING

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure .It is also termed data-driven or requirements-based testing. A testing method emphasized on executing the functions and examination of their input and output data.

WHITE BOX TESTING

Contrary to black-box testing software is viewed as a whitebox, or glass-box in white-box testing as the structure and flow of the software under test are visible to the tester.This testing is based on knowledge of the internal logic of an application's code. Testing plans are made according to the details of the software implementation, such as programming language, logic and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing or design-based testing.

UNIT TESTING

This involves testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.

System testing

Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

End-to-end testing

Similar to system testing involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

Usability testing

Under Usability Testing, User-friendliness check is done. The application flow is tested to know if a new user can understand the application easily or not, Proper help documented if a user gets stuck at any point.

Install/Uninstall testing

Tested for full, partial, or upgrade install/uninstall process on different operating systems under different hardware, software environment.

Recovery testing

Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Security testing

Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Check if system database is safe from external attacks.

Compability testing

Testing how well software performs in a particular hardware/software/operatingsystem/network environment and different combinations of above.

Comparision testing

Comparision of product strengths and weakness with previous versions or other similar products.

Alpha testing

In house virtual environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

Beta testing

Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

8. BENEFITS

8.1 BENEFITS

- Software provide easy management of student records.
- Software has very user friendly interface which is very easy to handle and understand.

- Software provides security to private data by hiding them.
- Software uses very less memory and takes less time to startup.

9. LIMITATIONS

9.1 LIMITATIONS

- Software is limited to Desktop only.
- System requires python interpreter installed on the system.
- All opinions of student management are not included in current version. Security options provide only low level security against beginner attackers.
- GUI is in English only.

10. FUTURE ENHANCEMENT

- This software can be made for all OS.
- Higher Security features can be included in this software.
- Program scheduling can also be included in this software.
- This software can be developed to use as tutorial to teach basic concepts of OS to new users.
- This software can be implemented with OS to reduce overhead of installing and running interface of each and every tool at different place.
- Automatic Shutdown through SMS can be implemented in this.

11. CONCLUSION

11.1 CONCLUSION

The project titled "Student Management System" is developed using Python Tkinter as front end and MYSQL database in back end to computerize the process of management of student records. This project covers only the basic features required.

12. REFERENCES

12.1 REFERENCES

- http://www.geeksforgeeks.org/python-gui-tkinter/
- http://www.javapoint.com/python-tkinter
- http://www.python.org/

BIOGRAPHIES

Mrs. Selina Khoirom
Assistant Professor
M. Tech.
OS, Cloud. Java

Laimujam Raj Singh
B.E 8th Semester
Computer Science & Engineering Department

Sougaijam Nealson Singh

B.E 8th Semester
Computer Science & Engineering Department

Laishram Rabi Singh
B.E 8th Semester
Computer Science & Engineering Department

Keisham Linthoingambi
B.E 8th Semester
Computer Science & Engineering Department