VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018, Karnataka

Python Application Programming

A mini project report on

Department Library Management

Submitted By:

Divya Bhat N **(1GA18CS055)**

Sharan S **(1GA18CS148)**

Under the Guidance of

Ms. Sushmitha S

Assistant Professor

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Global Academy of Technology

Rajarajeshwarinagar, Bengaluru – 560098

2020-2021

Global Academy of Technology

Department of Computer Science and Engineering

Certificate

This is to certify that the project entitled **"Department Library Management"** is a bonafide work carried out by **Divya Bhat N (1GA18CS055), Sharan S (1GA18CS148)** as a part of assignment in Python Applications Programming in Computer Science and Engineering during the year 2020-2021.

Max. marks	Marks obtained	Faculty Name and Signature
10		

ABSTRACT

A Library plays the most vital part in an Educational Institution. A library is a main element that thrives students to learn more about a particular subject. Digitalization is the next sensation. To encourage/improve digitalization, this project stands as an initiation. Bidding adieu to the process of data storing of library history in registers, it is time to implement a digitalised version of the same.

The website to be produced is the Digitalized Department Library Management. Here, there are 3 users. They are The Admin, The Teachers and The Students. The Admin has access to all the data and can make necessary changes or add in entries of new book arrivals, register students and also teachers along with the removal of the same. The Teachers and Students can also create their own personal accounts and view available books online and then borrow it later from the Department. The Student Information stored includes their Name, USN, Email ID, Date of Birth and a track of all the recently borrowed books. The Teacher Information stored includes their Name, Employee Code, Email ID, Date of Birth and a track of all the recently borrowed books. All the data and history of the person is stored using their unique member ID such as the USN or Employee Code. Finally, if the person wishes to quit, they can De-register by simply deactivating their account.

TABLE OF CONTENTS

ORGANIZATION OF THE REPORT

The report is divided into various chapters and is organized as follows:

Chapter 1: Introduction

This chapter includes brief introduction to Python Programming Language and its applications.

Chapter 2: System requirements

This chapter includes details of hardware and software requirements necessary for the execution of the project.

Chapter 3: Implementation and Results

This chapter includes the program code of the project and the results of successful runs of the code.

Conclusion

This section includes the conclusion about the project.

References

This section includes the bibliographical references used for the development of the project.

CHAPTER 1

INTRODUCTION

1.1. Python Programming language

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale project. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project. Guido van Rossum has since then withdrawn his nomination for the 2020 Steering council.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

1.2. Applications of Python programming language

1) Web Development

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on Instagram. Python provides many useful frameworks, and these are given below:

- Django and Pyramid framework (Used for heavy applications)
- Flask and Bottle (Micro-framework)
- Plone and Django CMS (Advance Content management)

2) Software Development

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

- SCons is used to build control.
- Buildbot and Apache Gumps are used for automated continuous compilation and testing.
- Round or Trac for bug tracking and project management.

3) Desktop GUI Applications

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface. Some popular GUI libraries are given below:

- Tkinter or Tk
- wxWidgetM
- Kivy (used for writing multitouch applications)
- PyQt or Pyside

4) Audio or Video-based Applications

Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are TimPlayer, cplay, etc. The few multimedia libraries are given below:

- Gstreamer
- Pyglet
- QT Phonon

5) Scientific and Numeric

In this era of Artificial intelligence where the machine can perform the tasks cloning the actions of humans, Python language is the most suitable language for Artificial intelligence or machine learning.

Implementation of machine learning algorithms require complex mathematical calculation. Python has various libraries for scientific execution such as Numpy, Scipy, Scikit-learn, etc. Few popular frameworks of machine libraries are given below:

- SciPy
- Scikit-learn
- NumPy
- Pandas
- Matplotlib

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

1. Operating System (Windows-XP or Higher)
2. Efficient Search Engine
3. SQL Server Edition

2.2 Hardware Requirements

1. Pentium III or better processor
2. 256 MB RAM
3. 10GB hard disk
4. 10 – 50 Mbps of Stable Network

CHAPTER 3

IMPLEMENTATION AND RESULTS

3.1 About Project

The Project demonstrates working of a digitalized management of the department library. The project has been completed with the complete implementation of the frontend and backend along with the connection of the front and back ends. HTML, CSS, Bootstrap and JavaScript have been implemented for the designing of the webpages. The database has been implemented using MySQL. The linking of the front and back ends is established using “Flask” a Python framework for Web Development.

In the project, The Admin has access to all details of all the users. Separate templates have been implemented for teacher and student users. The database comprises of 4 tables. One exclusively for storing student user information, one for storing teacher user information, one for storing details of books and one for storing the book borrowing information. The Teacher and Student Users can register online and create an account for themselves with functionalities such as viewing their details, updating their details, viewing available books, and viewing their book borrowing history. The Admin, has functionalities to add and remove student users, teacher users and also new book arrivals. The Admin also has access to the consolidated history of book borrowing. The history can be viewed in 3 formats, one exclusively providing borrowing info of all students, one providing borrowing info of all teachers and one complete view of book borrowing history without any constraints.

The project is mainly based on Web Development which is an all time in the corporate world, especially with the nation moving towards digitalization. The project was initiated to overcome the tedious tasks of maintain records through huge and heavy registers. The project has currently been implemented using only basic necessity to reduce human effort which in further time will be upgraded to give out much sophisticated versions.

3.2Project code

app.py

```
from flask import Flask, flash, request, url_for, redirect, render_template, session,
 g
from flask_bootstrap import Bootstrap
from utils.ins_val_db import stu, teach, stu_val, teach_val, S_logval, T_logval, book
_val, update
from utils.del_val_db import stu_del, teach_del, book_del, teach_del_acc, stu_del_acc
from utils.display import stu_display, teach_display, book_display, stu_history, teac
h_history, book_history
from utils.display import stu_det, teach_det, avail_books, p_teach_history, p_stu_his
tory
from utils.writing import stu_write, teach_write, book_write, stu_history_csv, teach_
history_csv, book_history_csv
from utils.borrow import admin_stu_borrow, admin_teach_borrow, returnbook
import urllib.request, time
import json

app = Flask(__name__)
app.secret_key = "Random"

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/student_login",methods=["GET","POST"])
def student_login():
    if request.method == "POST":
        req = request.form
        status = S_logval(req)
        if(status != "Success"):
            return redirect("/student_login")
        else:
            user = stu(req)
            session["user"] = user[0][0]
            return redirect("/studentprofile")
    return render_template("student_login.html")

@app.route("/teacher_login",methods=["GET","POST"])
def teacher_login():
    if request.method == "POST":
        req = request.form
        status = T_logval(req)
        if(status != "Success"):
            return redirect("/teacher_login")
        else:
            user = teach(req)
```

```
            session["user"] = user[0][0]
            return redirect("/teacherprofile")
    return render_template("teacher_login.html")

@app.route("/adminlogin",methods=["GET","POST"])
def adminlogin():
    if request.method == "POST":
        req = request.form
        if req["email"]=="ADMIN" and req["password"]=="ADMIN":
            session["user"] = "ADMIN"
            return redirect("/admin")
        else:
            return redirect("/adminlogin")
    return render_template("adminlogin.html")

@app.route("/student_signup",methods=["GET","POST"])
def student_signup():
    status = None
    if request.method == "POST":
        req = request.form
        status = stu_val(req)
        if(status != "Success"):
            return redirect("/student_signup")
        else:
            return redirect("/student_login")
    return render_template("student_signup.html")

@app.route("/teacher_signup",methods=["GET","POST"])
def teacher_signup():
    status = None
    if request.method == "POST":
        req = request.form
        status = teach_val(req)
        if(status != "Success"):
            return redirect("/teacher_signup")
        else:
            return redirect("/teacher_login")
    return render_template("teacher_signup.html")

@app.route("/admin")
def admin():
    if "user" in session:
        user = session["user"]
    else:
        return redirect("/adminlogin")
    return render_template("adminprofile.html")

@app.route("/studentprofile")
```

```
def studentprofile():
    if "user" in session:
        user = session["user"]
    else:
        return redirect("/student_login")
    return render_template("studentprofile.html")

@app.route("/teacherprofile")
def teacherprofile():
    if "user" in session:
        user = session["user"]
    else:
        return redirect("/teacher_login")
    return render_template("teacherprofile.html")

@app.route("/admin/addbook",methods=["GET","POST"])
def addbook():
    status = None
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = book_val(req)
            if(status != "Success"):
                return redirect("/admin/addbook")
            else:
                return redirect("/admin")
    else:
        return redirect("/adminlogin")
    return render_template("addbook.html")

@app.route("/admin/addstudent",methods=["GET","POST"])
def addstudent():
    status = None
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = stu_val(req)
            if(status != "Success"):
                return redirect("/admin/addstudent")
            else:
                return redirect("/admin")
    else:
        return redirect("/adminlogin")
    return render_template("addstudent.html")

@app.route("/admin/removebook",methods=["GET","POST"])
```

```
def removebook():
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = book_del(req)
            if(status == "Success"):
                return redirect("/admin")
            else:
                return redirect("/admin/removebook")
    else:
        return redirect("/adminlogin")
    return render_template("removebook.html")

@app.route("/admin/removestudent",methods=["GET","POST"])
def removestudent():
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = stu_del(req)
            if(status == "Success"):
                return redirect("/admin")
            else:
                return redirect("/admin/removestudent")
    else:
        return redirect("/adminlogin")
    return render_template("removestudent.html")

@app.route("/admin/viewbooks",methods=["GET","POST"])
def viewbooks():
    if "user" in session:
        user = session["user"]
        data = list()
        data = book_display()
        if not data:
            msg = "No Books"
        else:
            msg = "The Books data has been written on the Book.csv file in the Downlo
ads folder"
            book_write(data)
        flash(msg)
    else:
        return redirect("/adminlogin")
    return render_template("viewbooks.html")

@app.route("/admin/viewstudents",methods=["GET","POST"])
def viewstudents():
```

```
    if "user" in session:
        user = session["user"]
        data = list()
        data = stu_display()
        if not data:
            msg = "No Student Users"
        else:
            stu_write(data)
            msg = "The Student Users data has been written on the Student_Users.csv f
ile in the Downloads folder"
        flash(msg)
    else:
        return redirect("/adminlogin")
    return render_template("viewstudents.html")

@app.route("/admin/bookhistory")
def bookhistory():
    if "user" in session:
        user = session["user"]
        data = list()
        data = book_history()
        if not data:
            msg = "No Book History"
        else:
            msg = "The Books History data has been written on the Books_History.csv f
ile in the Downloads folder"
            book_history_csv(data)
        flash(msg)
    else:
        return redirect("/adminlogin")
    return render_template("bookhistory.html")

@app.route("/admin/studenthistory",methods=["GET","POST"])
def studenthistory():
    if "user" in session:
        user = session["user"]
        data = list()
        data = stu_history()
        if not data:
            msg = "No Student History"
        else:
            msg = "The Student History data has been written on the Students_History.
csv file in the Downloads folder"
            stu_history_csv(data)
        flash(msg)
    else:
        return redirect("/adminlogin")
    return render_template("studenthistory.html",data=data)
```

```
@app.route("/admin/addteacher",methods=["GET","POST"])
def addteacher():
    status = None
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = teach_val(req)
            if(status != "Success"):
                return redirect("/admin/addteacher")
            else:
                return redirect("/admin")
    else:
        return redirect("/adminlogin")
    return render_template("addteacher.html")

@app.route("/admin/removeteacher",methods=["GET","POST"])
def removeteacher():
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = teach_del(req)
            if(status == "Success"):
                return redirect("/admin")
            else:
                return redirect("/admin/removeteacher")
    else:
        return redirect("/adminlogin")
    return render_template("removeteacher.html")

@app.route("/admin/viewteachers",methods=["GET","POST"])
def viewteachers():
    if "user" in session:
        user = session["user"]
        data = list()
        data = teach_display()
        if not data:
            msg = "No Teacher Users"
        else:
            msg = "The Teacher Users data has been written on the Teacher_Users.csv f
ile in the Downloads folder"
            teach_write(data)
        flash(msg)
    else:
        return redirect("/adminlogin")
    return render_template("viewteachers.html")
```

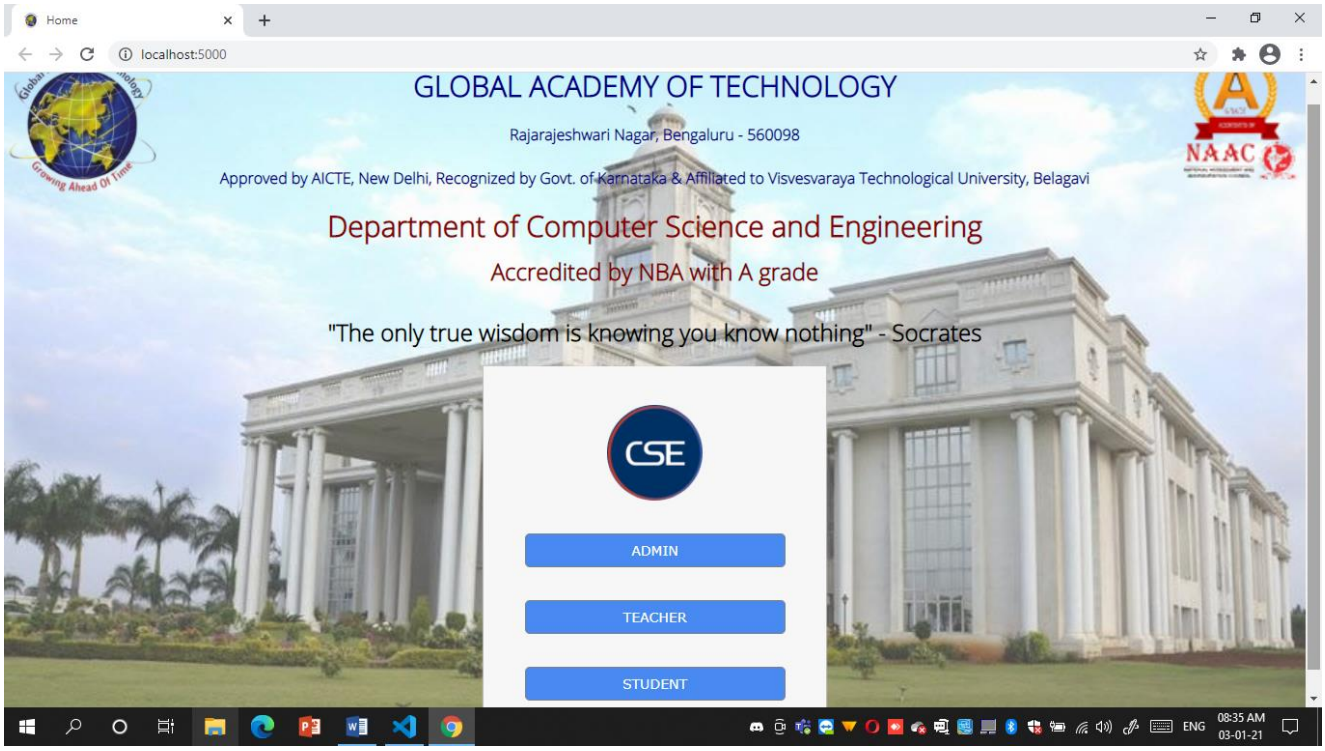
```
@app.route("/admin/teacherhistory")
def teacherhistory():
    data = list()
    data = teach_history()
    if not data:
        msg = "No Teacher History"
    else:
        msg = "The Teacher History data has been written on the Teachers_History.csv
file in the Downloads folder"
        teach_history_csv(data)
    flash(msg)
    return render_template("teacherhistory.html")

@app.route("/admin/bookreturn",methods=["GET","POST"])
def bookreturn():
    if "user" in session:
        user = session["user"]
        if request.method=="POST":
            req = request.form
            status = returnbook(req)
            if status == "Success":
                return redirect("/admin")
            else:
                return redirect("/admin/bookreturn")
    else:
        return redirect("/adminlogin")
    return render_template("bookreturn.html")

@app.route("/admin/studentborrow",methods=["GET","POST"])
def studentborrow():
    if "user" in session:
        user = session["user"]
        if request.method=="POST":
            req = request.form
            status = admin_stu_borrow(req)
            if status == "Success":
                return redirect("/admin")
            else:
                return redirect("/admin/studentborrow")
    else:
        return redirect("/adminlogin")
    return render_template("studentborrow.html")

@app.route("/admin/teacherborrow",methods=["GET","POST"])
def teacherborrow():
    if "user" in session:
        user = session["user"]
```

```
        if request.method=="POST":
            req = request.form
            status = admin_teach_borrow(req)
            if status == "Success":
                return redirect("/admin")
            else:
                return redirect("/admin/teacherborrow")
    else:
        return redirect("/adminlogin")
    return render_template("teacherborrow.html")

@app.route("/studentprofile/studetails")
def studetails():
    if "user" in session:
        user = session["user"]
        data = list()
        data = stu_det(user)
        if not data:
            data = [("NULL","NULL","NULL","NULL","NULL")]
    else:
        return redirect("/student_login")
    return render_template("studetails.html",result=data)

@app.route("/teacherprofile/teachdetails")
def teachdetails():
    if "user" in session:
        user = session["user"]
        data = list()
        data = teach_det(user)
        if not data:
            data = [("NULL","NULL","NULL","NULL","NULL")]
    else:
        return redirect("/teacher_login")
    return render_template("teachdetails.html",result=data)

@app.route("/studentprofile/stuupdate",methods=["GET","POST"])
def stuupdate():
    status = None
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = update(req,session["user"],"s")
            if status == "Success":
                return redirect("/studentprofile")
            else:
                return redirect("/studentprofile/stuupdate")
    else:
```

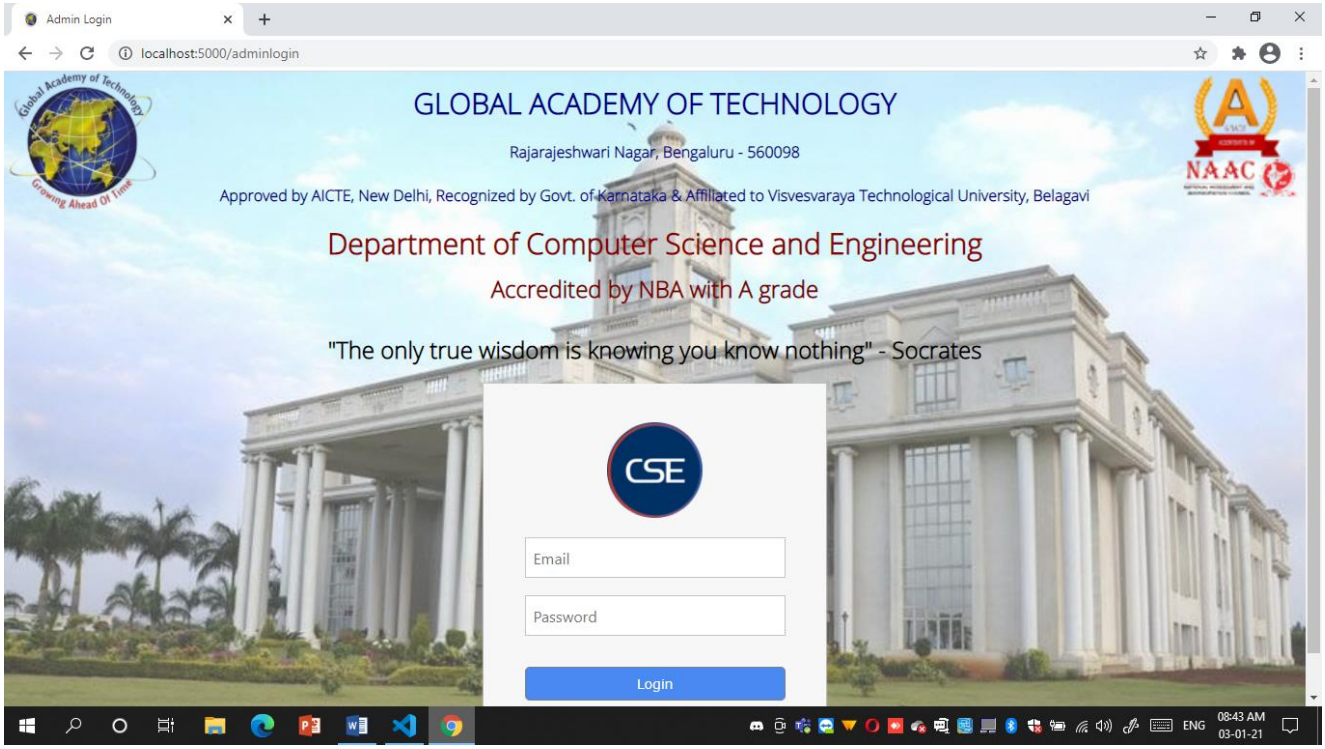
```
        return redirect("/student_login")
    return render_template("stuupdate.html")

@app.route("/teacherprofile/teachupdate",methods=["GET","POST"])
def teachupdate():
    status = None
    if "user" in session:
        user = session["user"]
        if request.method == "POST":
            req = request.form
            status = update(req,session["user"],"t")
            if status == "Success":
                return redirect("/teacherprofile")
            else:
                return redirect("/teacherprofile/teachupdate")
    else:
        return redirect("/teacher_login")
    return render_template("teachupdate.html")

@app.route("/studentprofile/stuavailbooks")
def stuavailbooks():
    if "user" in session:
        user = session["user"]
        data = list()
        data = avail_books()
        if not data:
            data = [("NULL","NULL","NULL","NULL")]
    else:
        return redirect("/student_login")
    return render_template("stuavailbooks.html",result=data)

@app.route("/teacherprofile/teachavailbooks")
def teachavailbooks():
    if "user" in session:
        user = session["user"]
        data = list()
        data = avail_books()
        if not data:
            data = [("NULL","NULL","NULL","NULL")]
    else:
        return redirect("/teacher_login")
    return render_template("teachavailbooks.html",result=data)

@app.route("/studentprofile/stuhistory")
def stuhistory():
    if "user" in session:
        user = session["user"]
        data = list()
```

```
        data = p_stu_history(user)
        if not data:
            data = [("NULL","NULL","NULL","NULL","NULL")]
    else:
        return redirect("/student_login")
    return render_template("stuhistory.html",result=data)

@app.route("/teacherprofile/teachhistory")
def teachhistory():
    if "user" in session:
        user = session["user"]
        data = list()
        data = p_teach_history(user)
        if not data:
            data = [("NULL","NULL","NULL","NULL","NULL")]
    else:
        return redirect("/teacher_login")
    return render_template("teachhistory.html",result=data)

@app.route("/studentprofile/studelacc")
def studelacc():
    if "user" in session:
        user = session["user"]
        stu_del_acc(user)
    else:
        return redirect("/student_login")
    return render_template("studelacc.html")

@app.route("/teacherprofile/teachdelacc")
def teachdelacc():
    if "user" in session:
        user = session["user"]
        teach_del_acc(user)
    else:
        return redirect("/teacher_login")
    return render_template("teachdelacc.html")

@app.route("/logout")
def logout():
    session.pop("user",None)
    return render_template("logout.html")

@app.errorhandler(404)
def err_404(e):
    return render_template("err_404.html")

if __name__ == "__main__":
    app.run(debug=True,host="localhost",port=5000)
```

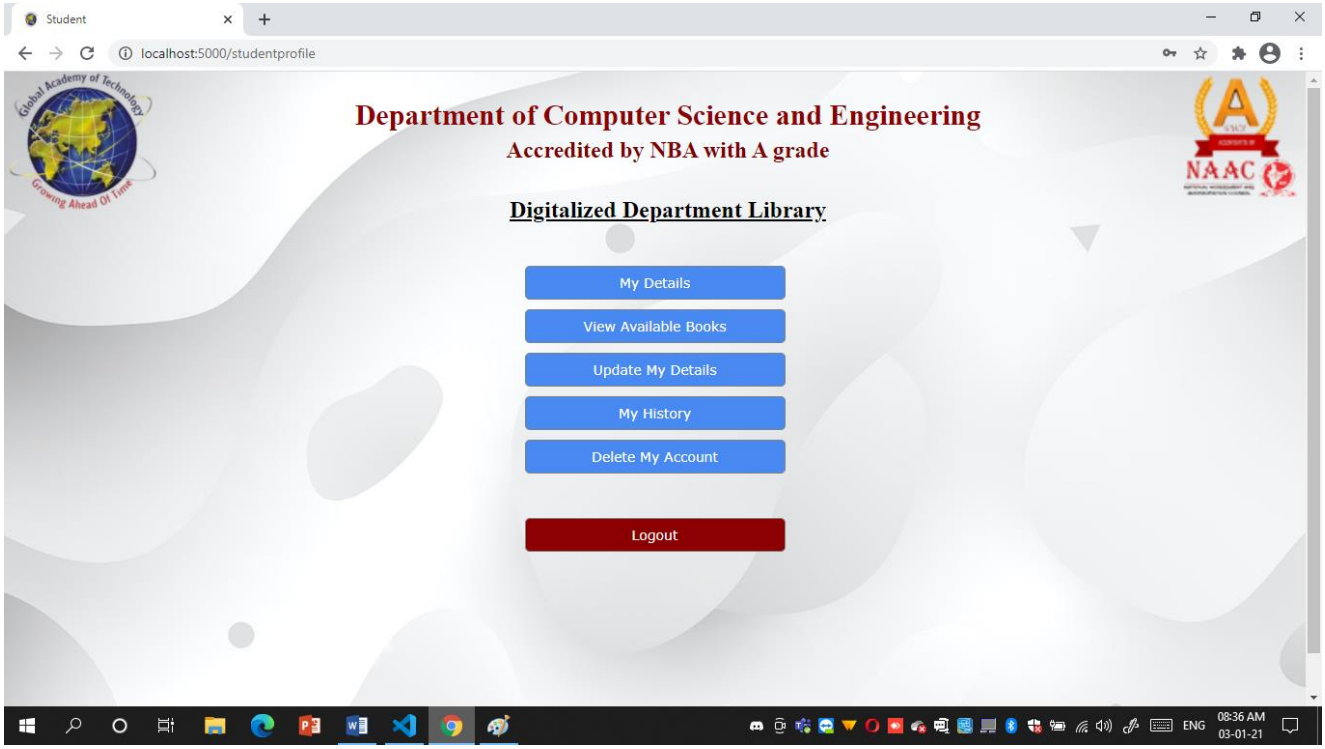
3.3 Results



Snapshot 3.3.1 : Home page



Snapshot 3.3.2 : Admin Login (Similar for Student & Teacher Users)



Snapshot 3.3.3 : Student Profile (Similar to Teacher Profile)

Snapshot 3.3.4 : Admin Profile

CONCLUSION

On working on this project, it helped us enhance our knowledge about python over a vast extent apart from the academics. It also helped us understand the need for the structuring of the storage format in the backend database. Immense amount of knowledge was gathered by either of us in order to understand the concept of web development in python using the micro framework “Flask”. The working on this project has also instilled a passion for Web Development. The micro framework Flask, the csv module, working with csv files, implementation of sessions, importing self-defined functions and organizing them into other folders and then importing and implementing them, and many more such new concepts were looked into and learnt to bring out a better working of the project. Organization of utility files helps in easy debugging of code in case any bug is encountered. Finally, not only as a part of academics but also a vast number of concepts were implemented thereby increasing a wide amount of knowledge. The ease in using python compared to other programming languages has instilled the urge to learn more about python for various other domains apart from Web Development.

Apart from Computer Programming Skills and Programming Knowledge, the project helped us learn a lot about the need of soft skills. We learnt major soft skills such as Time Management, Team Work and above all, Patience. While one worked on the structure of the backend the other worked on the frontend design (Time Management). After the necessary templates and backend queries were implemented, working together in order to enhance the functionality and taking into account one another’s solution taught us immensely about Team Work. During execution if there encountered and error, working together on the error or the bug, taught us the need for Patience.

Overall, This Project was a very good entity that helped improve ourselves not only in academics point of view but also improved our soft skills. It is thereby been our pleasure to have worked with each other along with our guide to successfully complete the project on time with all necessary functionalities.

REFERENCES

Bibliography:

[1] “Automate the Boring Stuff with Python” by Al Sweigart

[2] “Think Python: How to Think Like a Computer Scientist” by Allen B Downey

[3] “Core Python Applications Programming” by Wesley J Chun

Websites:

[1] https://realpython.com/

[2] https://pythonise.com/series/learning-flask/flask-working-with-forms

[3] https://pythonhow.com/building-a-website-with-python-flask/