

# **PLACEMENT NOTIFIER**

*Submitted for Mini Project Report*

by

**M.SAKETH - 19BQ1A05B7**

**K.SAITEJA - 19BQ1A0577**

**K.SHRAVANI - 19BQ1A0587**

**K.SUMASRI - 19BQ1A0575**

Under the guidance of

**M. NAGA SRI HARSHA**



## **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

(B.Tech Program is Accredited by NBA)

## **VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

NAMBUR(V), PEDAKAKANI(M), GUNTUR-522 508

Tel no: 0863-2118036, url:[www.vvitguntur.com](http://www.vvitguntur.com)

June 2021



**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE  
Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified  
Nambur, Pedakakani (M), Guntur (Dt) - 522508

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**  
B.Tech Program is Accredited by NBA

## **CERTIFICATE**

This is to certify that this Socially Relevant Project Report is the bonafide work of **M.SAKETH, K.SAITEJA, K.SHRAVANI, K.SUMASRI** bearing **Reg.No. 19BQ1A05B7, 19BQ1A0577, 19BQ1A0587, 19BQ1A0575** who had carried out the project entitled “**PLACEMENT NOTIFIER**” under our supervision.

**Project Guide**

M. Naga Sri Harsha

**Head of the Department**

Dr. V. Rama Chandran

---

**Submitted for Viva voce Examination held on \_\_\_\_\_**

**Internal Examiner**

**External Examiner**

## TABLE OF CONTENTS

CH no.	TITLE	PAGE NO
	ABSTRACT	
	Contents	
1.	Introduction	1-2
2.	Concepts & Methods	3
	2.1 Problem Description	3
	2.2 Proposed Solution	3
	2.3 System Requirements	3
3.	Implementation	4-19
	3.1 Tools used	4-5
	3.2 Pseudo Code/Algorithms	6-13
	3.3 Screenshots	14-19
4.	Summary (or) Conclusion	20
	BIBLIOGRAPHY	21

## **ABSTRACT**

It is a website which is useful for both student and faculty where they can sign up and login into their respective accounts and are provided with certain features that allows them to see latest updates related to placement drives and also has many features such as adding students who got placed in specific companies which you can see in this project.

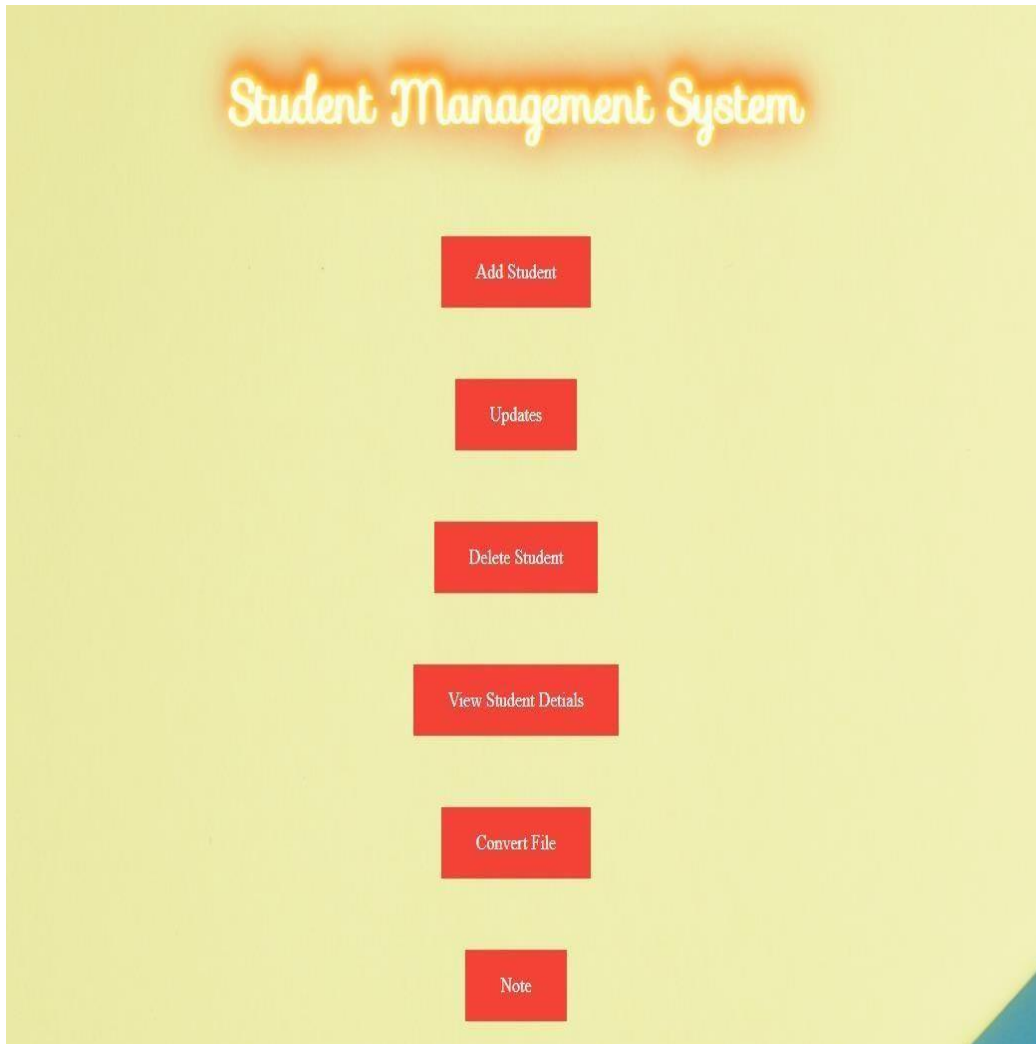
# **1. INTRODUCTION**

We've developed a web page which allows you to login into the portal and the portal contains fields as mentioned in the order given below:

- Sign Up
- Login
  - Add Student
  - Updates
  - Delete Student
  - View Student Details
  - Convert File
  - Note
- Logout

This is the file structure for our portal and this portal allows the user to add students , where each member is assigned a unique id number and will update the information regarding the company they are placed in along with minor details of the student. Also faculty can add a note on the matter what he wanted to convey to his group of students through note option. Also each student can view the live updates form different companies through Updates option.

The basic interface for our portal looks like the image shown below:



## **2. CONCEPTS AND METHODS**

### **2.1 PROBLEM DESCRIPTION:**

College management and faculty finds it difficult to filter students based on the company which they got placed and they are facing many difficulties in sending information to different batches and it is not sure the information is received to all the students.

### **2.2 PROPOSED SOLUTION:**

This web portal makes the task of college management much simpler by refining students based on the company they are placed and each guide teacher can also update the students with live placement drives that are going on by providing useful links to different websites. Also faculty can add any key points which he want to keep students updated on by providing note in the portal.

### **2.3 SYSTEM REQUIREMENTS:**

The basic requirements that are needed for this project to run on system are:

- Software Requirements:
  - ✓ OS :- Linux/Unix/MacOS/Windows(or any other)
  - ✓ Code Editor : Notepad/Visual Studio Code/Thonny
  - ✓ Languages used :- Python 3.9.2
  - ✓ Browser: Chrome/ Microsoft Edge
- Hardware Requirements:
  - ✓ RAM :- 4GB(or above)
  - ✓ Processor :- i5(or i7 recommended)

### **3.IMPLEMENTATION**

#### **3.1TOOLS USED:**

- There are some essentials frameworks that are used for developing this project. The entire project is done using the languages Html, CSS , JavaScript , Python and SQLite3.
- The details of frameworks and modules that are used for this project are as given below:
- **Flask** : Flask is a web application framework written in Python. Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.
- **Role:** Flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog or a wiki. We used flask for app routing and importing several modules required for user authentication and login.
- **SQLAlchemy:** SQLAlchemy can be used to automatically load tables from a database using something called reflection. Reflection is the process of reading the database and building the metadata based on that information.
- **Role:** We used SQLAlchemy for establishing a connection and connecting to a database.
- **OS** : The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.



- **Role:** The `*os*` and `*os. path*` modules include many functions to interact with the file system.
- **JSON :** Python has a built-in package called `json`, which can be used to work with JSON data. It's done by using the `JSON` module, which provides us with a lot of methods which among `loads()` and `load()` methods are gonna help us to read the JSON file.
- **Role:** JSON is a syntax for storing and exchanging data. JSON is text, written with JavaScript object notation. We used it for adding data and deleting data from form.
- **Werkzeug:** Werkzeug is a comprehensive WSGI web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.
- **Role:** We used it for generating password hash and checking the password hash when user provides any input.
- **SQLite3 :** SQLite3 can be integrated with Python using `sqlite3` module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249.
- **Html & Css:** Html is used for designing templates like login page , sign up , add student , delete student page etc... , Where as Css is used for adding background images , styling , font and buttons for the web page.

### **3.2 PSUEDO CODE/ ALGORITHMS:**

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    notes = db.relationship('Note')
```

The above written pseudo code corresponds to the creation of an user and storing the user details in a database.

```
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('index'))
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user=current_user)
```

The above pseudo code is for retrieving data from login page and the code for login page is separately written inside a html page called login.html.

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
```

The above shown code is for logout redirection where the above function will again redirects to login page.

```
@app.route('/index', methods=['GET', 'POST'])
def index():
    return render_template("index.html");

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

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

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

The above shown methods will return the html pages as templates whenever they are called and they are used in a sequential manner in the construction of this web portal.

```

@app.route('/sign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('email')
        first_name = request.form.get('firstName')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, first_name=first_name,
password=generate_password_hash(
            password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('login'))

    return render_template("sign_up.html", user=current_user)

```

The above shown code is signup validation and as said above the code for signup page was separately written inside a html document called signup.html.

```

def saveRecord():
    msg = "msg"
    if request.method == "POST":
        try:
            name = request.form["name"]
            email = request.form["email"]
            gender = request.form["gender"]
            contact = request.form["contact"]
            dob = request.form["dob"]
            address = request.form["address"]
            with sqlite3.connect("student_detials.db") as connection:
                cursor = connection.cursor()
                cursor.execute("INSERT into Student_Info (name, email, gender, contact,
dob,address) values (?,?,?,?,?)",(name, email, gender, contact, dob,address))
                connection.commit()
                msg = "Student detials successfully Added"
        except:
            connection.rollback()
            msg = "We can not add Student detials to the database"
        finally:
            return render_template("success_record.html",msg = msg)
            connection.close()

```

The above written code is for Saving the details of a particular student and adding them to the student information.

```

def student_info():
    connection = sqlite3.connect("student_detials.db")
    connection.row_factory = sqlite3.Row
    cursor = connection.cursor()
    cursor.execute("select * from Student_Info")
    rows = cursor.fetchall()
    return render_template("student_info.html",rows = rows)

```

The above shown code is for displaying student information which is stored in a database.

```

def deleterecord():
    id = request.form["id"]
    with sqlite3.connect("student_detials.db") as connection:

        cursor = connection.cursor()
        cursor.execute("select * from Student_Info where id=?", (id,))
        rows = cursor.fetchall()
        if not rows == []:

            cursor.execute("delete from Student_Info where id = ?",(id,))
            msg = "Student detial successfully deleted"
            return render_template("delete_record.html", msg=msg)

        else:
            msg = "can't be deleted"
            return render_template("delete_record.html", msg=msg)

```

The above shown code is for deleting a record of a particular student information from the database permanently.

```

def note():
    if request.method == 'POST':
        note = request.form.get('note')

        if len(note) < 1:
            flash('Note is too short!', category='error')
        else:
            new_note = Note(data=note, user_id=current_user.id)
            db.session.add(new_note)
            db.session.commit()
            flash('Note added!', category='success')
    return render_template("home.html", user=current_user)

```

The above shown code is for adding any particular note to the field notes and save the notes .

```
def delete_note():
    note = json.loads(request.data)
    noteId = note['noteId']
    note = Note.query.get(noteId)
    if note:
        if note.user_id == current_user.id:
            db.session.delete(note)
            db.session.commit()

    return jsonify({ })
```

The above shown code is for deleting a particular note from the notes field permanently.

```
class Note(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    data = db.Column(db.String(10000))
    date = db.Column(db.DateTime(timezone=True), default=func.now())
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
```

The following code shown above is for creating a class named note to store the notes written by user in a field called note.

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'thisisasecretkey'
db = SQLAlchemy(app)
```

The following code shown above is for creating a flask app and creating a database to store it.

```

connection.execute("create table Student_Info (id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT NOT NULL, email TEXT UNIQUE NOT NULL,
gender TEXT NOT NULL, contact TEXT UNIQUE NOT NULL, dob TEXT NOT NULL,
address TEXT NOT NULL)")
print("Table created successfully")
connection.close()

```

The above shown code shows the creation of a student database which is used to store the information of different fields of students such as id, gender, contact, dob and address of the student by creating a connection.

```

<thead>
  <td>ID</td>
  <td>Name</td>
  <td>Email</td>
  <td>Gender</td>
  <td>Contact</td>
  <td>D.O.B</td>
  <td>Company</td>

</thead>
{% for row in rows %}
<tr>
  <td>{{row["id"]}}</td>
  <td>{{row["name"]}}</td>
  <td>{{row["email"]}}</td>
  <td>{{row["gender"]}}</td>
  <td>{{row["contact"]}}</td>
  <td>{{row["dob"]}}</td>
  <td>{{row["address"]}}</td>

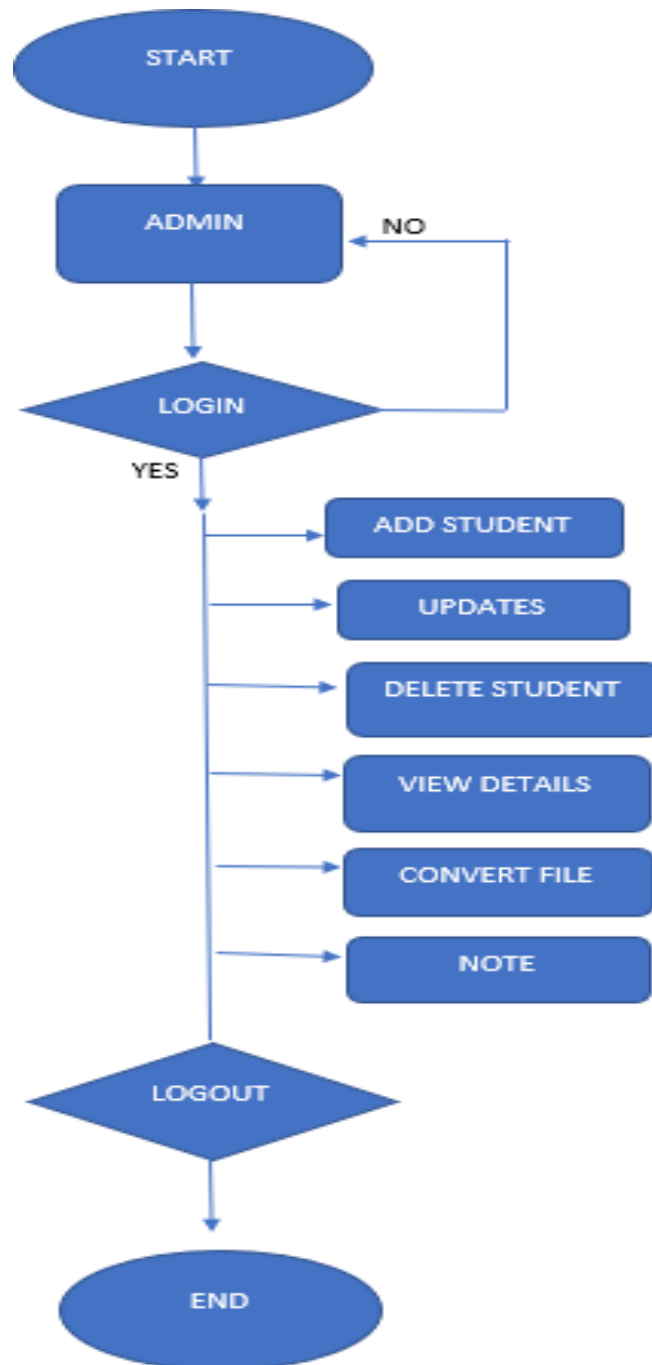
</tr>

```

The above shown html code is used for displaying the student information in a row wise



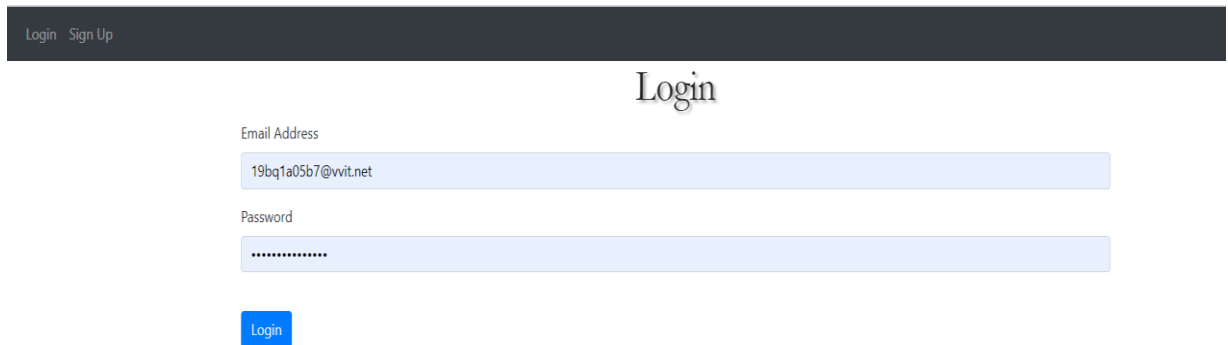
The flow of sequence of web page can be interpreted in the form of a flow chart as shown below:



### **3.3 SCREENSHOTS:**

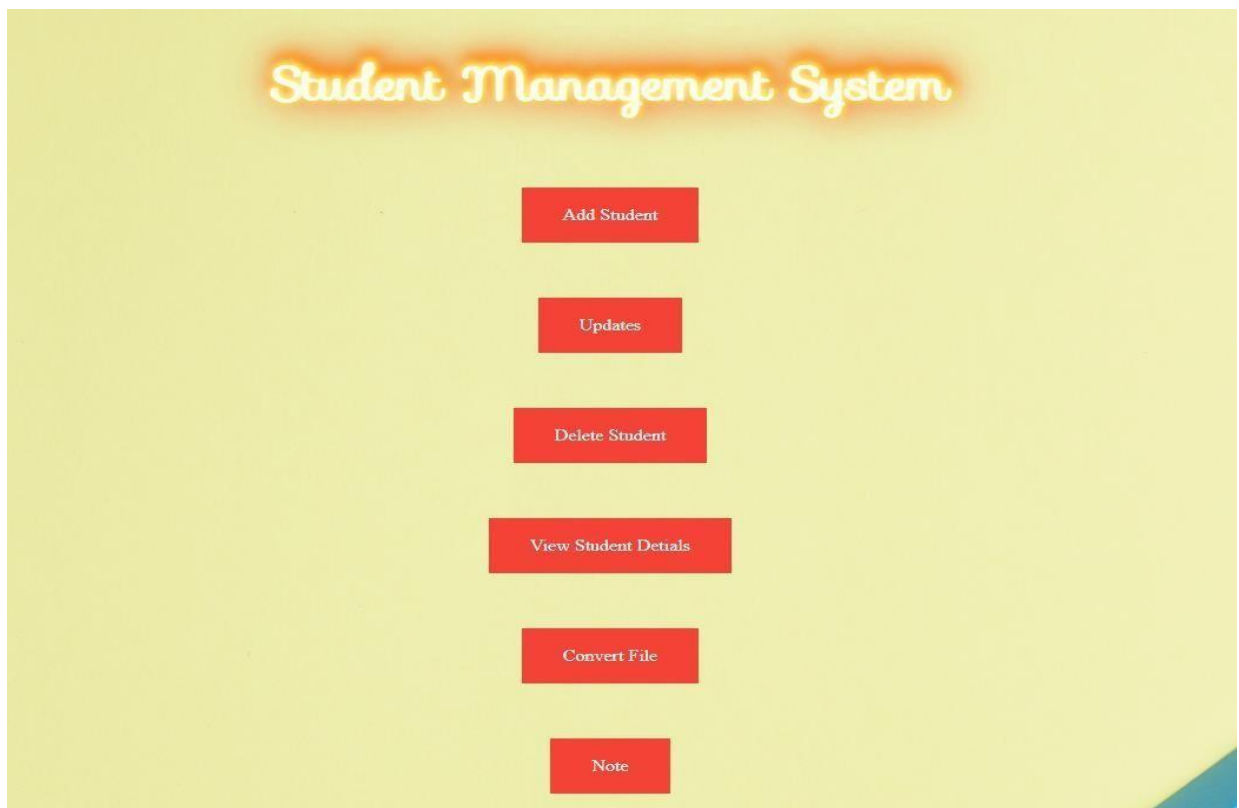
The screenshots for this project in a sequential manner are as shown below:

#### **Login page:**

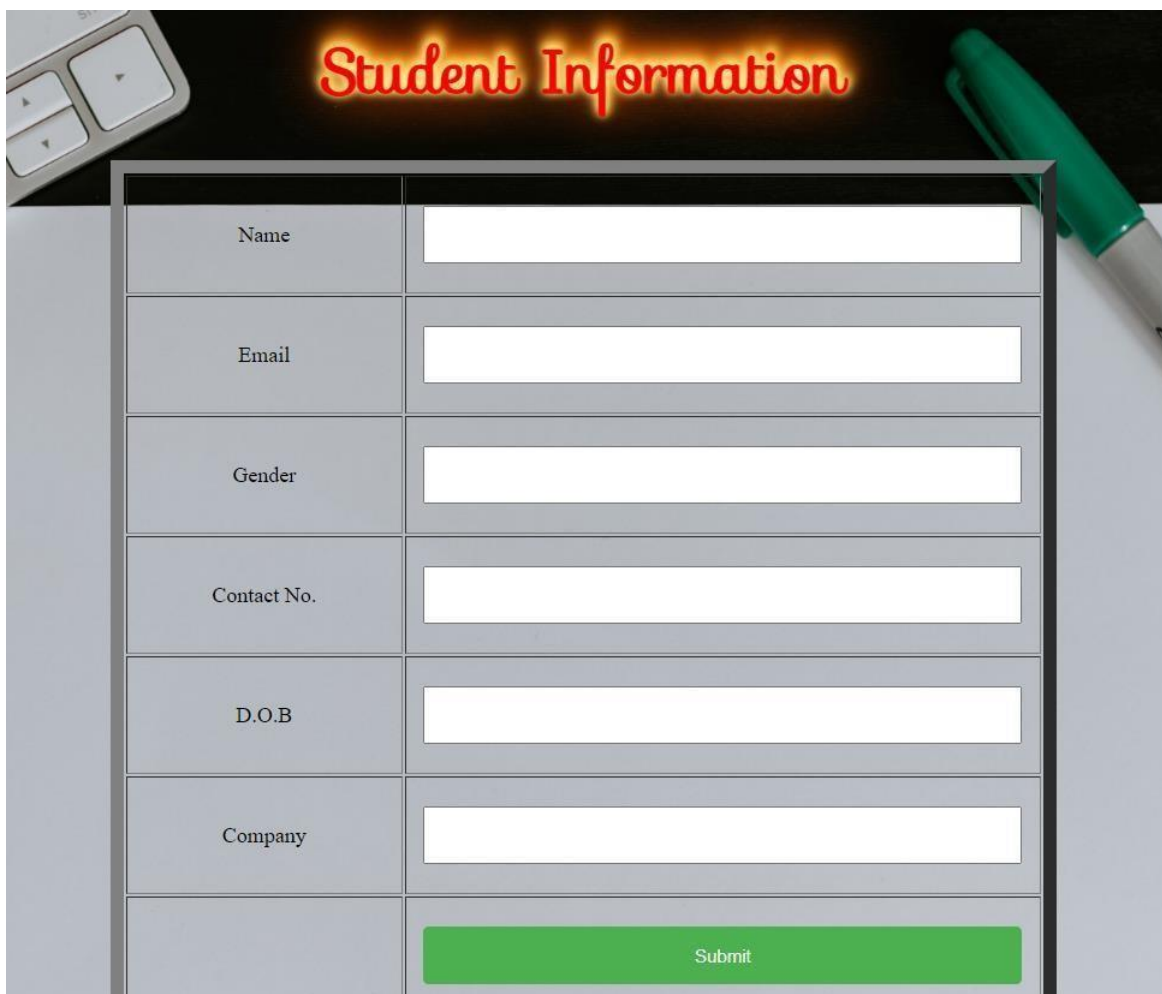


The screenshot shows a login interface. At the top, a dark grey header bar contains the links "Login" and "Sign Up". Below this, the word "Login" is displayed in a large, elegant serif font. Underneath, there are two input fields: "Email Address" with the text "19bq1a05b7@vvit.net" and "Password" with masked characters. A blue "Login" button is positioned below the password field.

#### **Home Page:**



### Add Student:



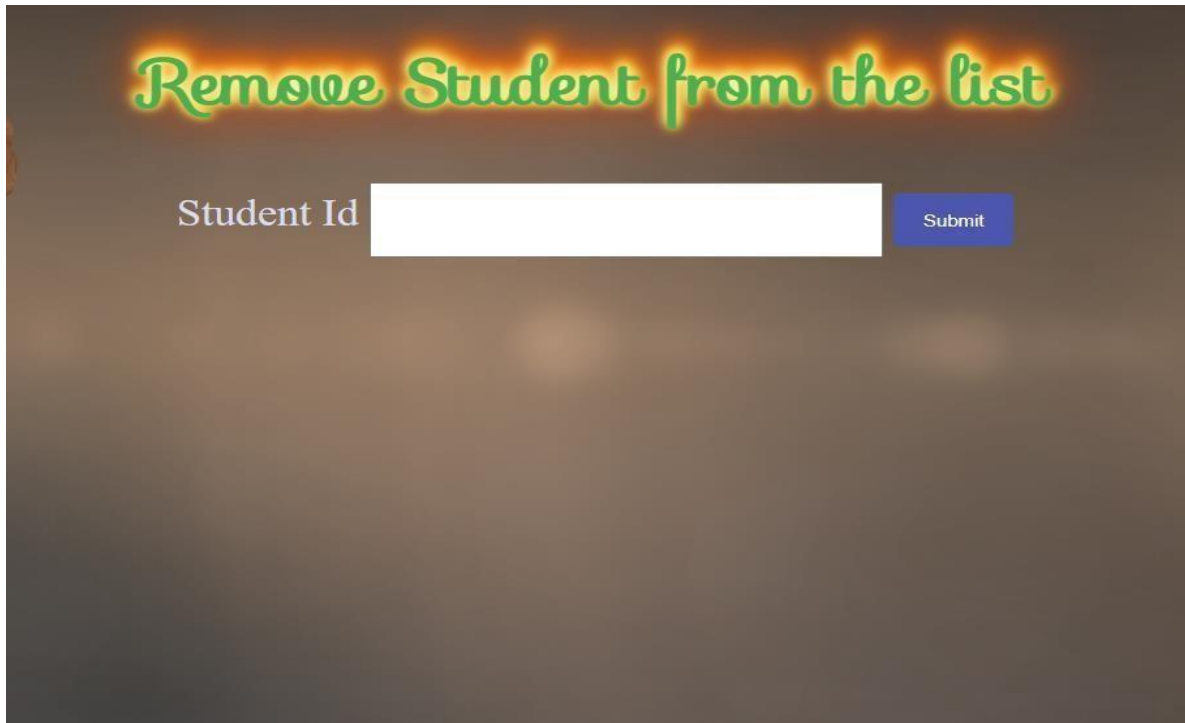
A screenshot of a web form titled "Student Information" in a glowing red font. The form is set against a background of a desk with a keyboard and a green pen. It contains several input fields for student details and a green submit button.

Name	<input type="text"/>
Email	<input type="text"/>
Gender	<input type="text"/>
Contact No.	<input type="text"/>
D.O.B	<input type="text"/>
Company	<input type="text"/>
	<input type="submit" value="Submit"/>

### Updates:



**Remove Student:**



Remove Student from the list

Student Id

**Notes:**



Notes

You can add anything here

### Convert File:

*Convert Excel to Table*

*Select Excel File*

Choose File

No file chosen

### Student Details:

*Student Details*

ID	Name	Email	Gender	Contact	D.O.B	Company
17	K Praveen	praveen@gmail.com	male	9390835145	02/03/2001	virtusa
18	K.Sainadh	kesanasainadh@gmail.com	male	9392520474	02/09/2001	CTS

[Go back to home page](#)

## **Signup Page:**

[Login](#) [Sign Up](#)

Sign Up

Email Address

Enter email

First Name

Enter first name

Password

Enter password

Password (Confirm)

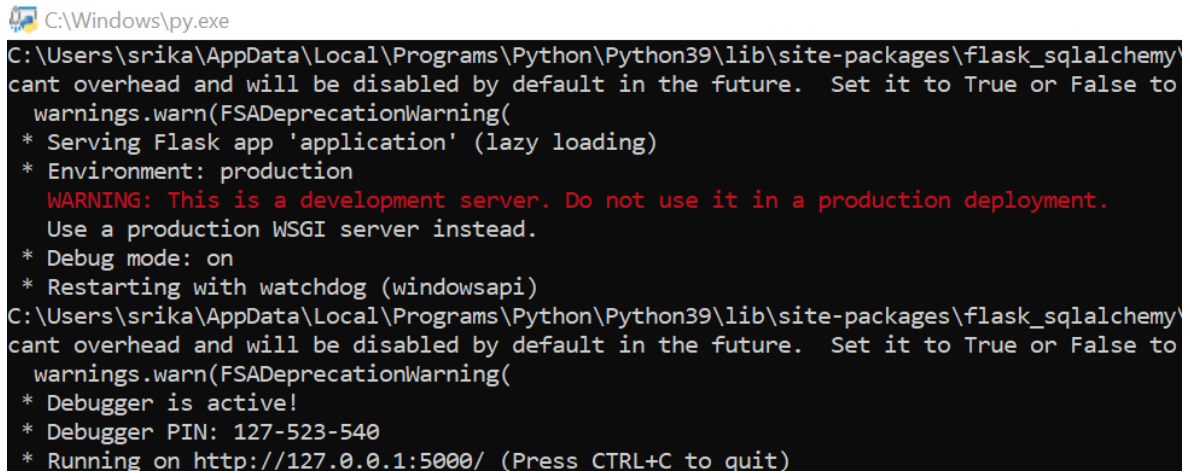
Confirm password

Submit

---

These are the following sequential order of pages which are there in the web portal.

## Execution of code:



```
C:\Windows\py.exe
C:\Users\srika\AppData\Local\Programs\Python\Python39\lib\site-packages\flask_sqlalchemy\
cant overhead and will be disabled by default in the future. Set it to True or False to
warnings.warn(FSADeprecationWarning(
* Serving Flask app 'application' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
C:\Users\srika\AppData\Local\Programs\Python\Python39\lib\site-packages\flask_sqlalchemy\
cant overhead and will be disabled by default in the future. Set it to True or False to
warnings.warn(FSADeprecationWarning(
* Debugger is active!
* Debugger PIN: 127-523-540
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The code will run on the web server which can be accessed by clicking on the link which is given below and once you quit from the session the web portal can't be accessed on browser.

#### **4. SUMMARY/CONCLUSION:**

- In the following way this web portal is helpful for many students and teachers to get updated about their status of placements in college and also can view the information that the faculty want to convey to the student in a simple and efficient way.



## **BIBLIOGRAPHY**

### **REFERENCES USED:**

- ✓ Websites for day to day updates on placement related info :
- ✓ TalentBattle : <https://talentbattle.in/placement-updates>
- ✓ PrepInsta : <https://prepinsta.com/>
- ✓ GeeksGod : <https://geeksgod.com/category/campus-drives/>