PYTHONE PROJECT

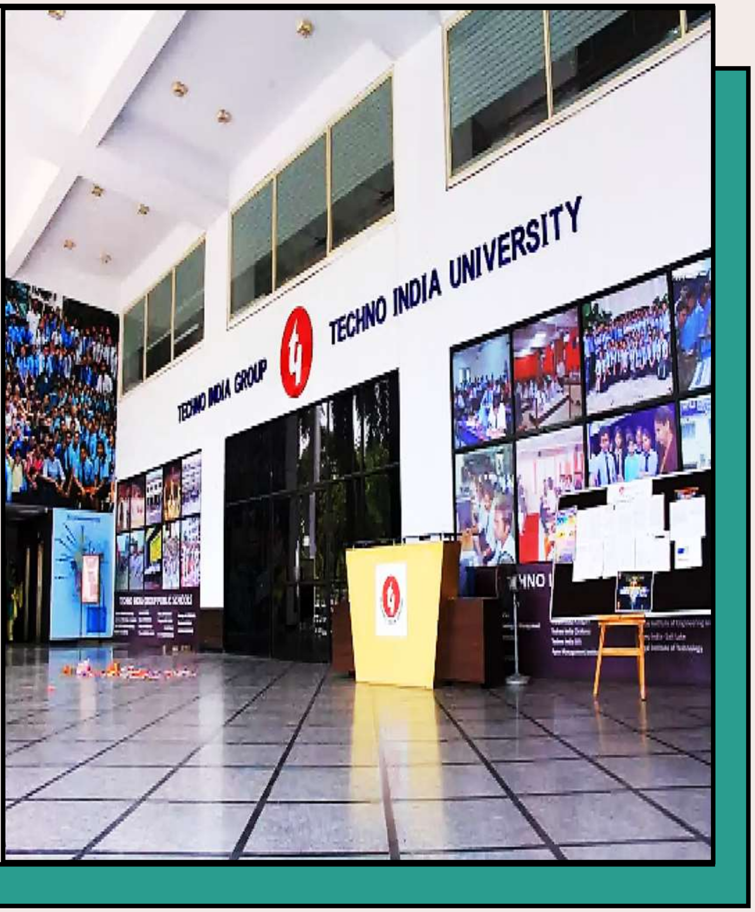
College Placement Management System

Team Member :

- KOYANA PRASAD --231001709013
- PRITAM PRADHAN --231001709007

Guided by

Ms Humaira Khatoon



Agenda

1

Introduction

2

Motivation

3

ER diagram

4

Code
&
output

5

Requirement
&
Advantage

Introduction

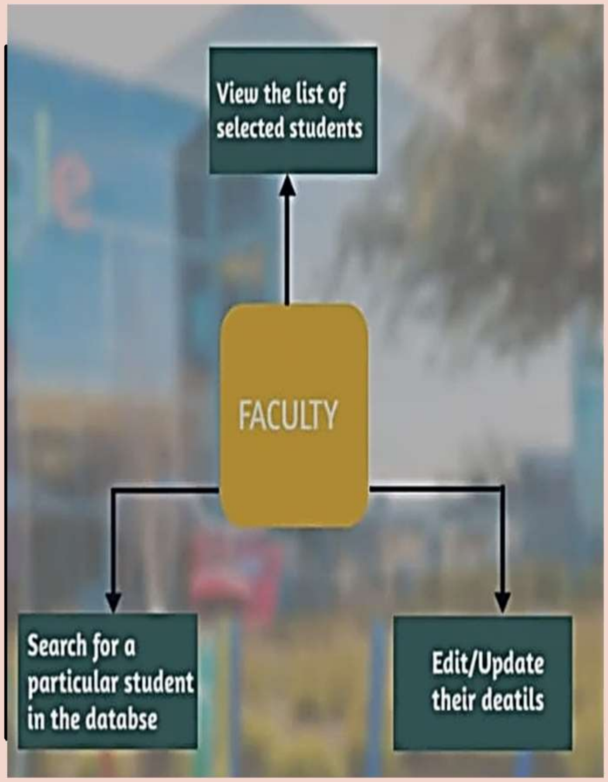
❑ The project aims to automate the process of placement management system and make the process of recruitment easier.

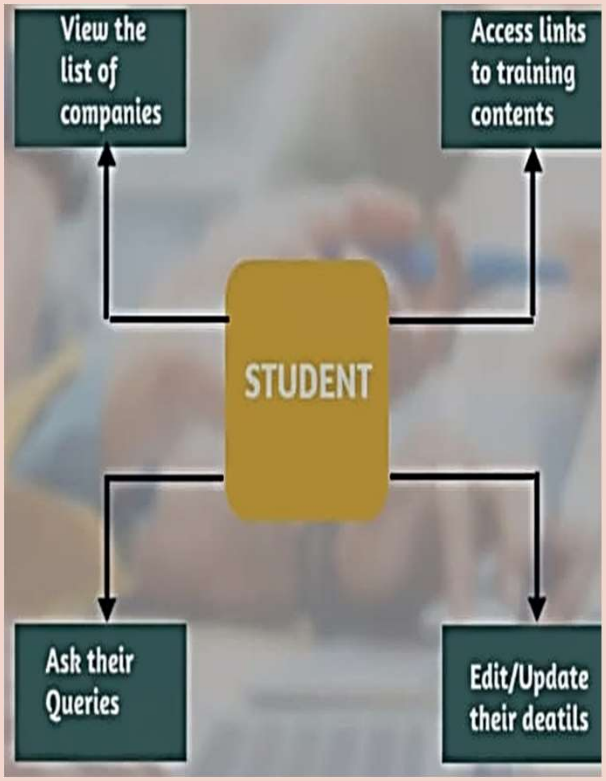
❑ It provides a platform where it allows the students to view the companies that are currently recruiting and also provides information on the companies and the packages they offer so that the students may view and assess their opportunities.

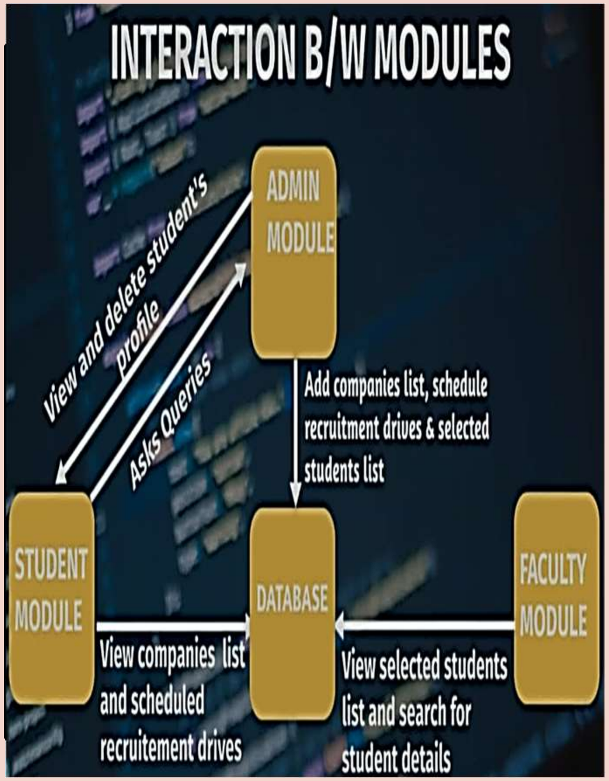
❑ It helps the company to find the eligible candidates according to their criteria.There are mainly 3 modules in this system:

1.Student Module
2.Faculty Module
3.Admin Module

Relation B/W Modules

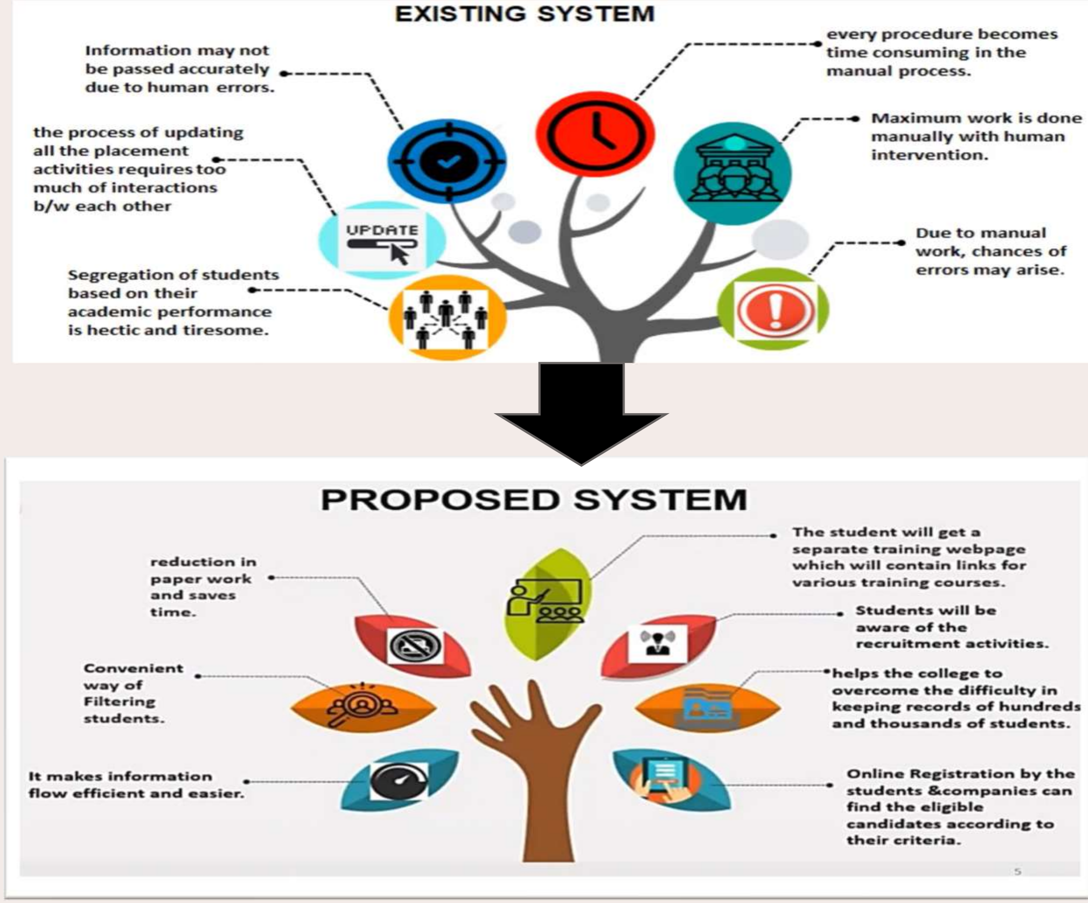

View the list of selected students
FACULTY
Search for a particular student in the databse
Edit/Update their deatils

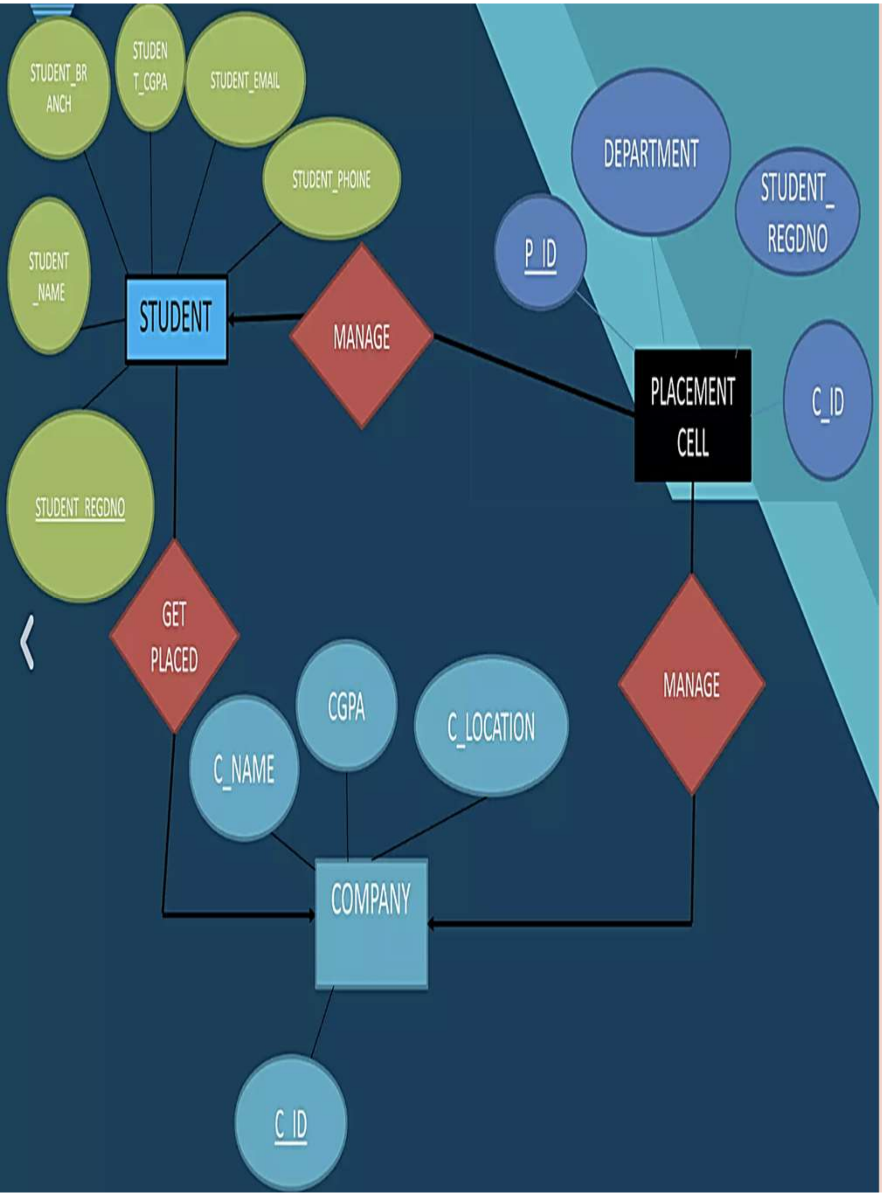

View the list of companies
Access links to training contents
STUDENT
Ask their Queries
Edit/Update their deatils


INTERACTION B/W MODULES
ADMIN MODULE
View and delete student's profile
Asks Queries
Add companies list, schedule recruitment drives & selected students list
STUDENT MODULE
DATABASE
FACULTY MODULE
View companies list and scheduled recruitement drives
View selected students list and search for student details

MOTIVATION

- ❑ The institute's industrial image is enhanced by placements.
- ❑ At times it becomes tedious to manage student data while keeping track of various interested companies.
- ❑ During several parallel placements, miscommunication or loss of notifications is quite common that creates a poor impression of either the organization or on their faculties.
- ❑ To resolve these problems, a smart management system is required to maintain accurate data records, placement scheduling, student notifications to stay tuned and track the placements.


EXISTING SYSTEM
Information may not be passed accurately due to human errors.
every procedure becomes time consuming in the manual process.
the process of updating all the placement activities requires too much of interactions b/w each other
Maximum work is done manually with human intervention.
UPDATE
Due to manual work, chances of errors may arise.
Segregation of students based on their academic performance is hectic and tiresome.
PROPOSED SYSTEM
reduction in paper work and saves time.
The student will get a separate training webpage which will contain links for various training courses.
Students will be aware of the recruitment activities.
Convenient way of Filtering students.
helps the college to overcome the difficulty in keeping records of hundreds and thousands of students.
It makes information flow efficient and easier.
Online Registration by the students &companies can find the eligible candidates according to their criteria.



ER Diagrarepresentsm

This ER (Entity Relationship) Diagram the model of College Placement Management System Entity. The entity-relationship diagram of College Placement Management System shows all the visual instrument of database tables and the relations between Students, Company etc. It used structure data and to define the relationships between structured data groups of College Placement Management System functionalities. The main entities of the College Placement Management System are Students, Company, and Placement Cell.

Code

```
class CollegePlacementSystem:
    def __init__(self):
        self.students = {}
        self.jobs = {}

    def add_student(self, student_id, name, skills):
        self.students[student_id] = {'name': name, 'skills': skills, 'placement': None}

    def add_job(self, job_id, title, required_skills):
        self.jobs[job_id] = {'title': title, 'required_skills': required_skills, 'filled': False}

    def display_students(self):
        print("\nStudents:")
        for student_id, details in self.students.items():
            print(f"{student_id}: {details['name']} - Skills: {details['skills']} - Placement:
{details['placement']}")

    def display_jobs(self):
        print("\nJobs:")
        for job_id, details in self.jobs.items():
            status = "Filled" if details['filled'] else "Available"
            print(f"{job_id}: {details['title']} - Required Skills: {details['required_skills']} - Status:
{status}")

    def match_students_to_jobs(self):
        for job_id, job_details in self.jobs.items():
            if not job_details['filled']:
                for student_id, student_details in self.students.items():
                    if not student_details['placement'] and
set(job_details['required_skills']).issubset(set(student_details['skills'])):
# Match found
                        student_details['placement'] = job_details['title']
                        job_details['filled'] = True
                        break

    def run_placement_system(self):
        # Sample data
        self.add_student(1, 'John', ['Python', 'Java', 'C++'])
        self.add_student(2, 'Alice', ['Java', 'JavaScript'])
        self.add_student(3, 'Bob', ['Python', 'C#'])

        self.add_job(101, 'Software Developer', ['Python', 'Java'])
        self.add_job(102, 'Web Developer', ['JavaScript', 'HTML', 'CSS'])
        self.add_job(103, 'Data Analyst', ['Python', 'SQL'])

        # Display initial state
        self.display_students()
        self.display_jobs()

        # Match students to jobs
        self.match_students_to_jobs()

        # Display final state
        print("\nPlacement Results:")
        self.display_students()
        self.display_jobs()

if __name__ == "__main__":
    placement_system = CollegePlacementSystem()
    placement_system.run_placement_system()
```

Output

```
Students:
1: John - Skills: ['Python', 'Java', 'C++'] - Placement: None
2: Alice - Skills: ['Java', 'JavaScript'] - Placement: None
3: Bob - Skills: ['Python', 'C#'] - Placement: None

Jobs:
101: Software Developer - Required Skills: ['Python', 'Java'] - Status: Available
102: Web Developer - Required Skills: ['JavaScript', 'HTML', 'CSS'] - Status: Available
103: Data Analyst - Required Skills: ['Python', 'SQL'] - Status: Available

Placement Results:

Students:
1: John - Skills: ['Python', 'Java', 'C++'] - Placement: Software Developer
2: Alice - Skills: ['Java', 'JavaScript'] - Placement: None
3: Bob - Skills: ['Python', 'C#'] - Placement: None

Jobs:
101: Software Developer - Required Skills: ['Python', 'Java'] - Status: Filled
102: Web Developer - Required Skills: ['JavaScript', 'HTML', 'CSS'] - Status: Available
103: Data Analyst - Required Skills: ['Python', 'SQL'] - Status: Available
```

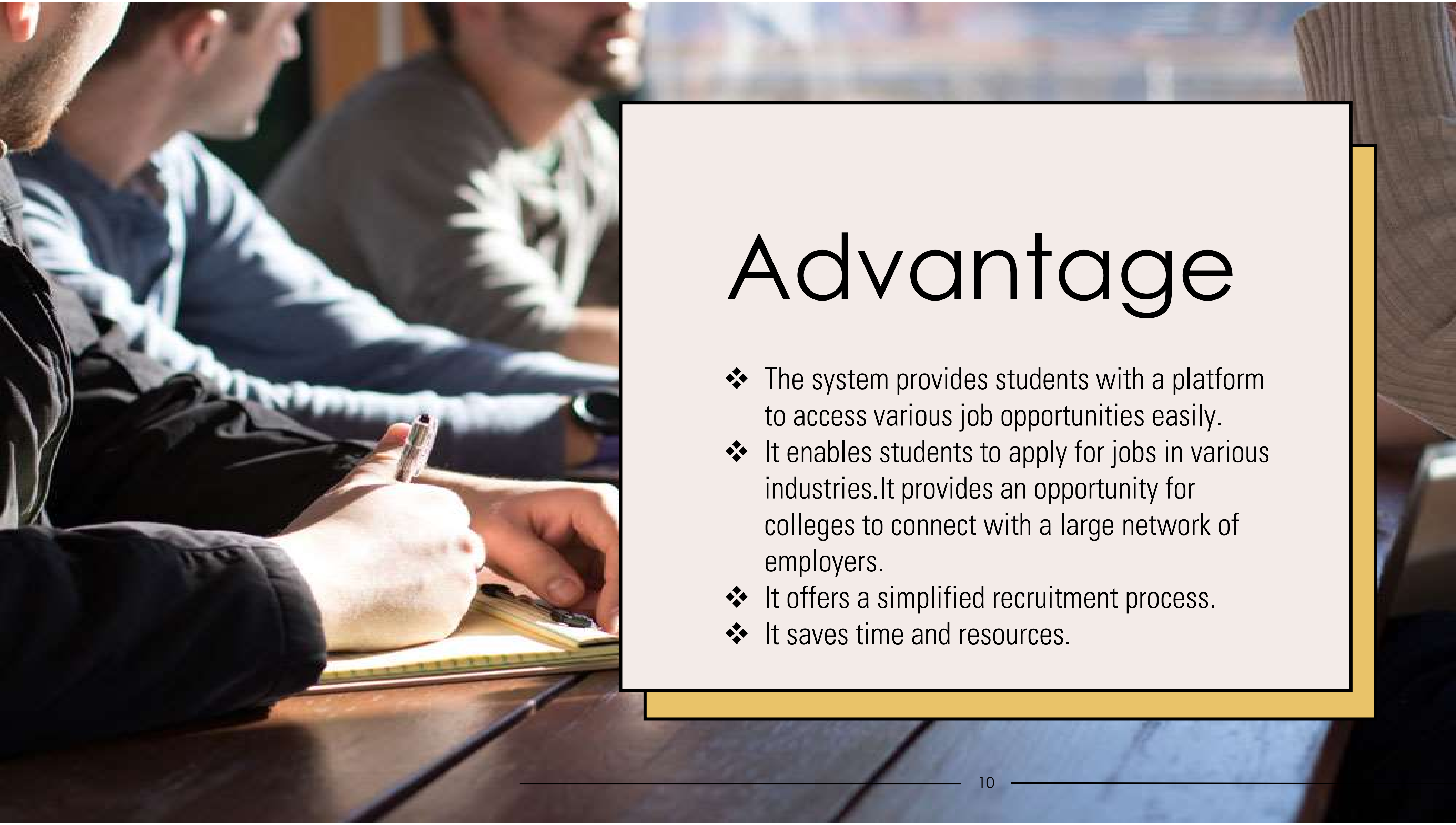
Requirement

Software requirement

- ➢ Laptop or pc
- ▪ Python
- ▪ Sublime text editor
- ▪ Xamp server

Hardware requirement

- ➢ Laptop or pc
- ▪ Windows7 or high
- ▪ I3 processor system or higher
- ▪ 4gb ram



Advantage

- The system provides students with a platform to access various job opportunities easily.
- It enables students to apply for jobs in various industries.It provides an opportunity for colleges to connect with a large network of employers.
- It offers a simplified recruitment process.
- It saves time and resources.

Thank you