A Mini Project Report on

“Mentor Mentee Management System”

Submitted in partial fulfillment of the requirement for
Degree in T.E. of Engineering (Information Technology)

By

Janhavi Patil (5020139)
Hishali Rana (5020146)
Akhil Suryam (5020149)
Divya Vinod (5020163)

Guided by:

Mrs. Poonam Bari

Department of Information Technology
Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai – 400703

University of Mumbai
2022-2023

CERTIFICATE

This is to certify that the Mini Project entitled

"MENTOR MENTEE MANAGEMENT SYSTEM"

Submitted By

Janhavi Patil
Hishali Rana
Akhil Suryam
Divya Vinod

In partial fulfillment of the degree of **T.E**. in **Information Technology** for term work of the Semester V Mini Project – 2 A is approved.

External Examiner **Internal Examiner**

_____ _____

Internal Guide

_____

Head of the Department **Principal**

_____ _____

Date: - **College Seal**

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all academic honesty and integrity principles and have not misrepresented, fabricated, or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

(Name of student and Roll No.)

Date:

ABSTRACT

A mentee is like a bird with wings, who doesn't know how to fly. In today's day and age education system, the student-mentor relationship is the most pivotal component for all-around development. A mentor may share with the mentee information about his or her own career path, as well as provide guidance, motivation, emotional support, and role modeling. A mentor may help explore careers, set goals, develop contacts, and identify resources.

The traditional and manual methodology of the mentor-mentee management system is tedious and time inefficient and it creates a bridge between the mentor and mentee thus leading to miscommunication.

Our website aims to bridge the gap between a student and a mentor. It presents an innovative counterpart to the traditional method of result generation feedback and grievances, with the help of automation and confidentiality this will create a well-managed system for teachers and students to properly address student issues and provide an accurate analysis for teachers which would further promote better performance. This would generate revenue by pitching this website to education institutes.

INDEX

LIST OF FIGURES

ACKNOWLEDGEMENTS

The making of the project “Mentor Mentee Management System” involves the contribution of many people. We would like to convey our sincere thanks to Dr. S.M. Khot, principal Fr. C Rodrigues Institute of Technology, Vashi for giving us the opportunity to showcase our skills and providing us with the necessary resources. We would also like to convey our heartfelt gratitude to the Head of Department of Information Technology, Prof. Vaishali Bodade for her constant support and motivation. We express deep gratitude to our project guide and mentor Mrs. Poonam Bari for her constant motivation to think out of the box and immense contribution throughout this project. Last but not the least, we convey our heartfelt thanks to the project coordinator, Prof Lakshmi Gadhikar for supporting and guiding us throughout the process. We also extend our heartfelt thanks to our families and well-wishers.

Yours sincerely.

Janhavi Patil (5020139)

Hishali Rana (5020146)

Akhil Suryam (5020149)

Divya Vinod (5020164)

INTRODUCTION

Background and Motivation

Development and securing excellent human resources under both internal and external environmental changes is the key deciding factor of national competitiveness. Currently, most colleges provide students with relevant information and vocational guidance via systems such as an on/off-line career information office or consultation center and an internship. However, since a systematic connection between individual students is not made, its effect is utterly limited. Therefore, it is considered that college graduates generally cannot meet the demand of Industry. The Mentor-Mentee Management System helps in bridging this gap between college authorities and students, providing them with superior academic knowledge.

Problem Definition

To develop an efficient web application that will automate the process of mentoring and reduce manual processes.

Conventional Mentoring system is time consuming, tedious and may cause several inconsistencies in maintaining data.

Traditional mentoring also leads to a wide communication gap between the mentor and the mentee which leads to a huge downfall in their performance and does not promote overall growth.

Scope

- Digitalising a manual process hence eradicating paperwork for the mentors as they can easily add and update data for the mentees which can be further automated.
- Data would be more correct in digital mode and there is less scope for mistakes to occur.
- The entire process would remain more confidential and transparent.
- It will be more convenient for both the mentors and the mentees, the mentees can also have more open and one-to-one communication which will remain strictly confidential.

Limitations

- The key issue with this application would be that although it would automate the basic structure of the mentor-mentee process, automating the internal processes would be tough, and hence the application would be less dynamic.
- Although the mentee would be validated with little rights while using the application, it would be less secure thus leading to various security issues.
- Data has to be updated every now and then, although this task could be automated but there are several loopholes which could lead to the occurrence of anomalies and exceptions in the process of automation.

2. LITERATURE SURVEY AND ANALYSIS

2.1 Related Work

It makes use of two-tier architecture that acts as an interface between the mentor and the mentee. The Mentor-Mentee Management System is developed on a client-server model that has a user application on the client side and a data source on the server side. Overall the system contains one main admin under which many mentors and each mentor has a set of students allocated by the admin at the same time the mentor is willingly taking the students for giving valuable encouragement for the improvement of the student in an academic institute. The architecture of Mentoring specified here is specific to the academic institution and if the mentoring is required in other institutions or organizations this architecture is not applied and has to be changed accordingly. The mentors also play a critical role by giving the right feedback to the right students. The mentors are mediators between the admin users and the student's system use. Mentors are also provided with login credentials by the admin to log in and check the information of the students and do an analysis of each and every mentee.

2.2. Existing System

There are numerous examples of student mentoring systems. Our college website, the FCRIT portal, will be something we are all familiar with. Although it is a much more dynamic system than what we are aiming for, some of the characteristics, such as attendance and IA marks, are similar. Teachers and students can both use the FCRIT portal. Teachers can submit student information such as IA scores and attendance records. This data is saved and can be accessed by both teachers and students at a later time. Students, on the other hand, can submit feedback forms and course exit surveys. This makes it easier to keep track of everything.

Pushfar

PushFar provides a highly effective, efficient and easily customisable solution to launching, running and scaling mentoring programs, schemes and initiatives. Combined with our training, resources and support, our data-driven matching algorithms, integration with existing LMS and SSO solutions, administrator panels and our iOS and Android apps, PushFar is a game changing mentoring solution

Mentorcity

MentorCity offers comprehensive, easy-to-use and cost effective online mentoring software for companies, schools and associations for their member engagement, succession planning, leadership development, and diversity and inclusion strategies. The MentorCity platform saves organizations time and money by minimizing matching responsibilities, allowing program administrator(s) to focus their efforts on building a mentoring culture that achieves exceptional business results.

Zintal

Zigtal helps businesses engage, develop and retain talent by matching employees' aspirations with organizational goals. It includes an employee personal branding model, an outcome-led engagement model, and gamification capabilities to improve outcomes. Managers can gain insights into engagements, behavior patterns, in-demand skills and other metrics to build strategies.

Appreiz

Appreiz is a web-based performance management software designed to help businesses reward and recognize staff members to facilitate employee engagement. Key features include peer appraisals, custom rating scales, weighted performance measures, individual development plans, goal setting and benchmarking.

3. SYSTEM DESIGN

3.1. Architectural Diagram/ block diagram

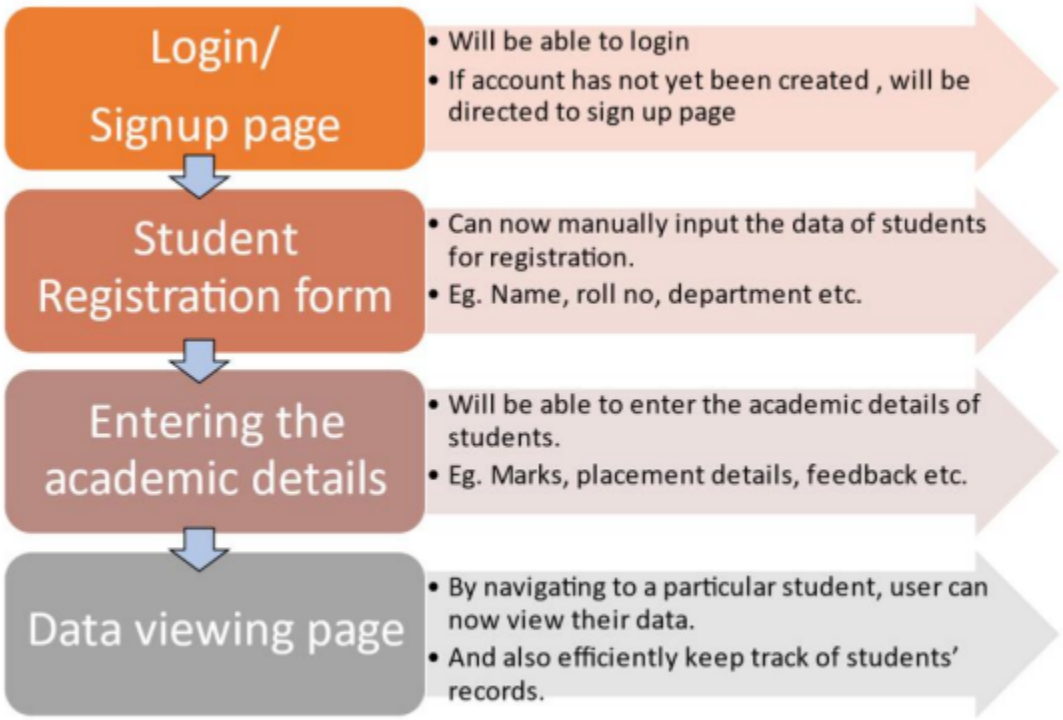
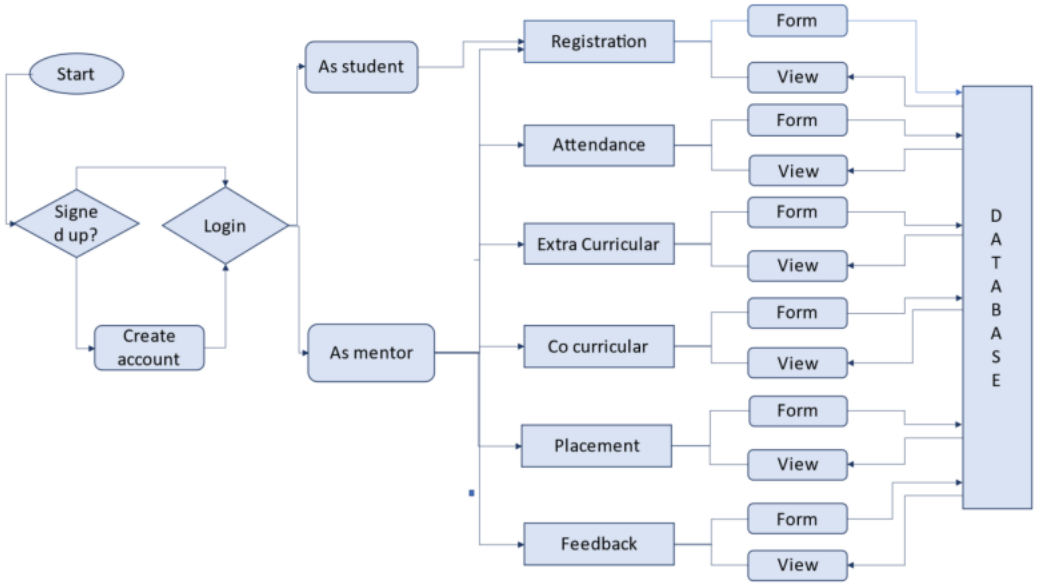


Figure 1. Architectural Diagram

Figure 1 demonstrates the system architecture and flow of data from the user to the backend. The Login and signup page would be used to authenticate the users. Data would be collected from the student registration form and would be used in the process of report generation which can be viewed by the mentee for whom the report is generated.

3.2. Flow Chart



-

Figure 2. Flowchart

Figure 2 represents the flowchart for the system design. The mainpage would be the Login Page for the mentor and the mentee in case they have already logged in else they would be asked to sign up. Separate access rights would be given to the mentor and the mentee. Academic details, Personal details, etc will be collected from the mentee and will be sent to the assigned mentor and only he/she will be able to view it.

4.IMPLEMENTATION DETAILS

4.1: Software Requirements:

- HTML
- CSS
- JavaScript
- Python
- Django

Hardware Requirements

Any device that can support the above-mentioned software

4.2 Methodology

- LoginCheckMiddleWare:- The module allows users to create their respective account on the application. If an account is already created, it lets the user login with username and credentials. There are three types of login available, one as mentee (student), mentor (teacher) and admin.
- STAFF VIEWS : - This module enables user access with the staff information from the database. It also lets users view all staff related details such as corresponding mentee assigned to the staff , generic staff details, feedback on particular staff and other related details.
- StudentViews : - This module allows user access with the student information from the database. It also lets users view all student related details such as corresponding mentor assigned to the student , generic students details, feedback log to view all previous problems raised and associated solutions in detail.
- Forms:- This particular module is responsible for all the form generation. The data generated by the forms module is used as input for report generation.
- Admin:- This module takes in all generic activities performed by admin into its functionality. It lets admin view and manage mentor and mentee details. Also helping admin assign roles and responsibility to mentors about their particular mentee.

- EmailBackend:- This specific module enables online reach and secured communication between mentor and mentee. It allows mentors to send corresponding reports generated of the particular mentee, directly to the mentees email and thus confidentiality as the security goal is achieved.
- ReportGen:- This module forms the crux of taking mentee data from databases such as their attendance, academic performance and other details to produce a transparent report .This report is finally mailed to the respective mentee via mails from the EmailBackend module. The process will enable mentee view a detailed report, analyse their performance and discuss ways to improve their overall accomplishments

EXPERIMENTAL RESULTS

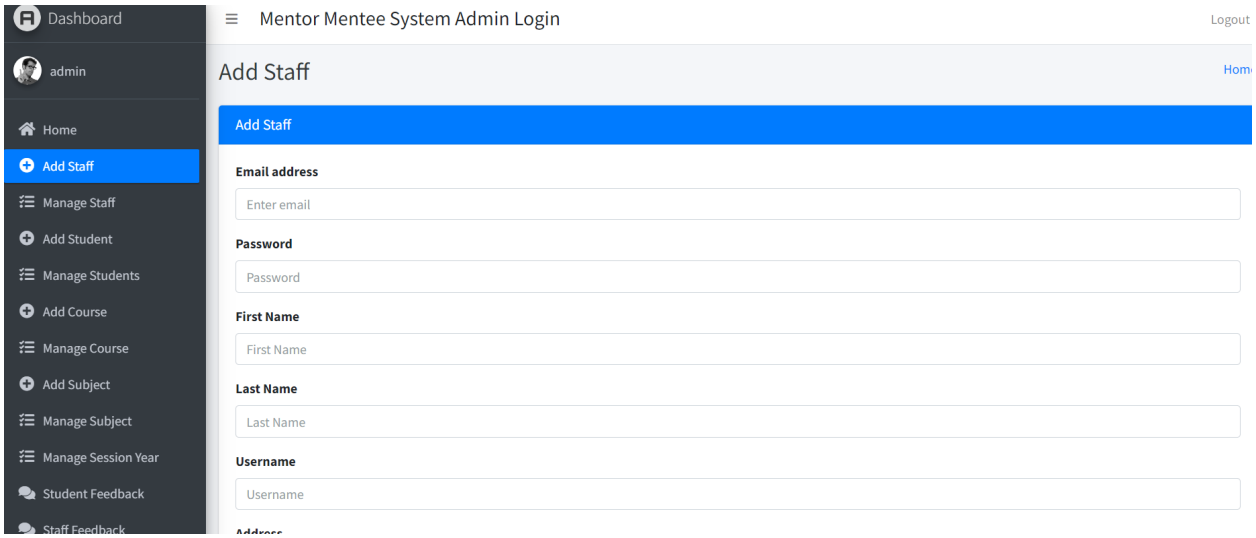


Figure. 3 Add staff

Admin has to login their credentials, click on add staff and enter details of new staff with their corresponding information. On enter , this staff information will be saved in staff database.

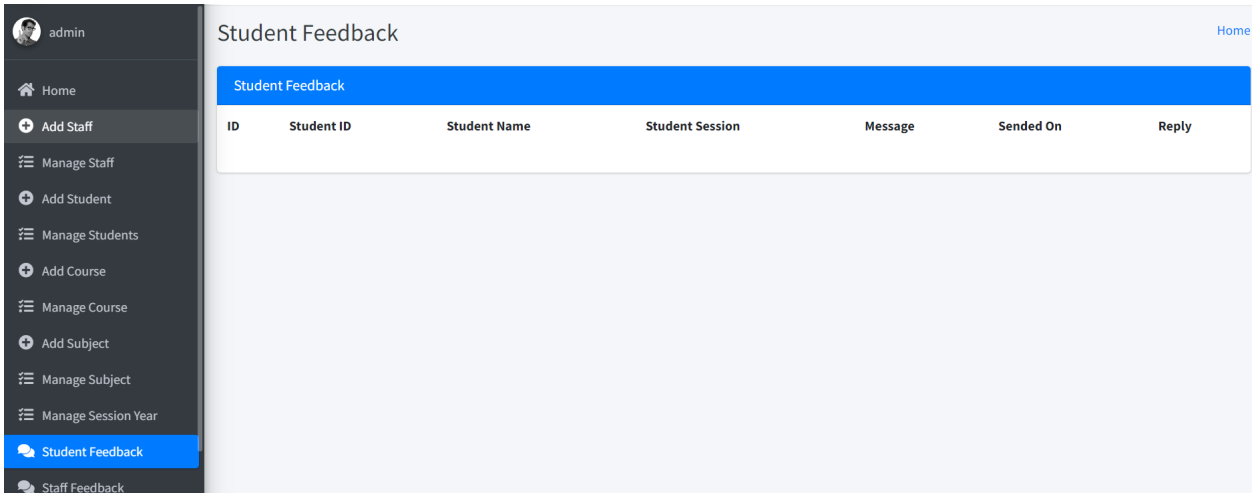


Figure. 4 Student Feedback

Admin will be access all the feedbacks provided by mentees to their corresponding mentors.After they click on student feedback they will be able to access these information.

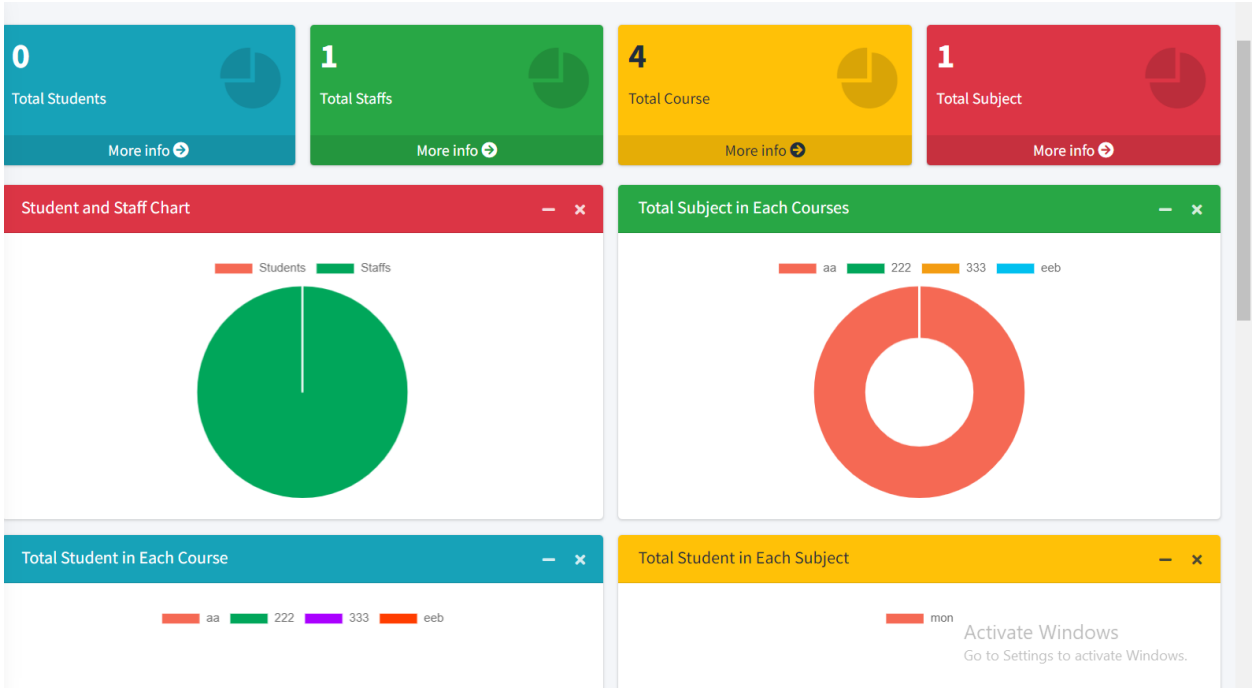


Figure. 5 Dashboard

Admin has to login their credentials, this dashboard will be displayed on the front page. This allows admin to access staff members,total student present , total subjects and corresponding pie charts.

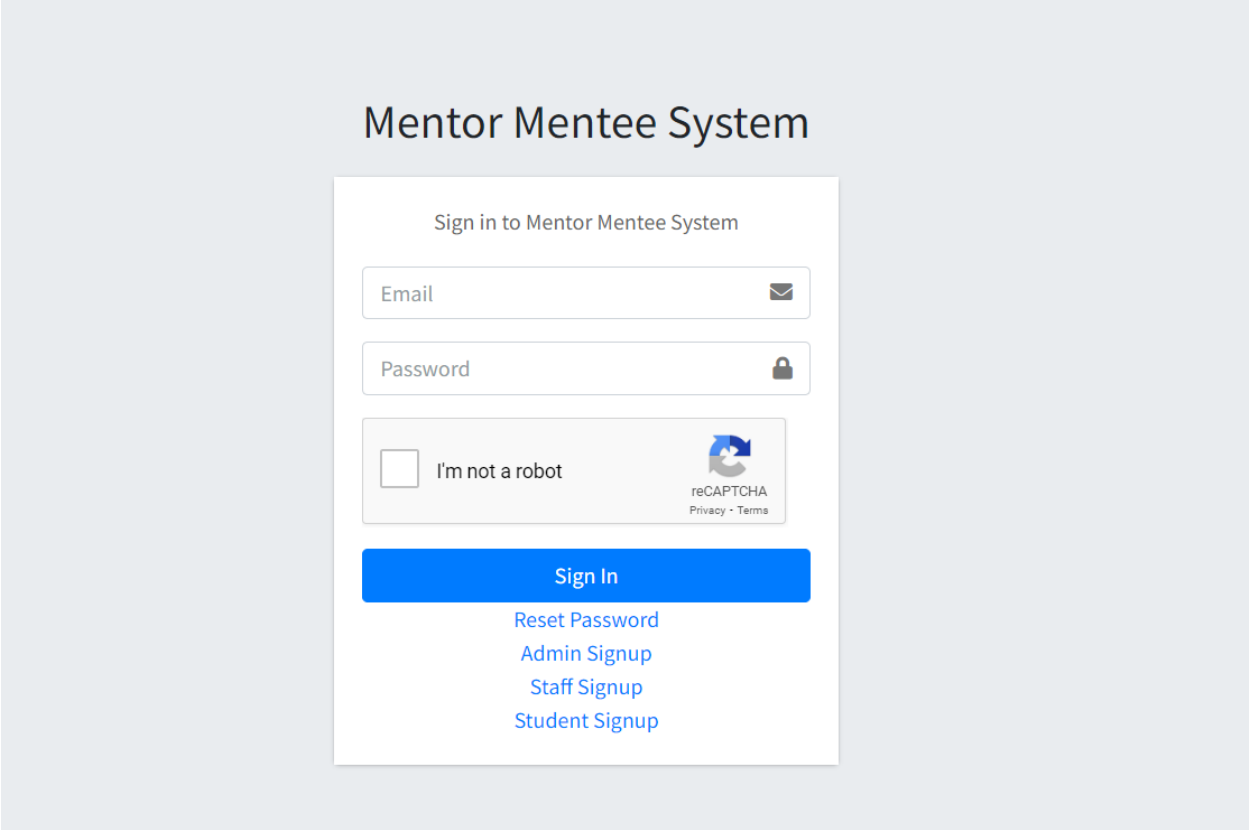


Figure. 6 Login Page

User will face this login page where they have to enter their credentials such as user name , password. Also fill cache and finally sign in.

CONCLUSIONS AND FUTURE SCOPE

Conclusions

To address the gap between the mentor and the mentee, we have successfully developed a web application that will act as an interface between the mentor and the mentee and automate the traditional method of mentoring. This application could also be used in corporations, and other educational institutions and also for distance education platforms. It would eradicate the conventional methods of mentoring and would improve the interaction by automating tedious tasks such as manually filling in the information, automating the task of report generation, and providing the mentees with continuous feedback.

Future Scope

- All the processes like filling in personal information, and academic details could be automated.
- The application would be integrated with other student data management portals.
- Providing the functionality to the application to work on smartphones as well as on Laptops/PCs

REFERENCES

- [1] Kaveh Abhari, David Williams, Pooja Pawar, Kashish Panjwani - Smart Entrepreneurial Systems: An Application of Deep Reinforcement Learning in Improving Entrepreneurship Mentorship - Future of Information and Communication Conference - 2021
- [2] Hyeyoung Cho, Sul-Ah Ah - An Automated Data-Driven Matching Process for Mentoring Program - ICCC - 2014
- [3] Sandra L. Williams, Justin (Jin-Hong) Kim- E-mentoring in Online Course Projects: Description of an EMentoring Scheme- International Journal of Evidence-Based Coaching and Mentoring Vol. 9, No. 2 - August 2011
- [4] Joanne D. Leck, Penny M. Wood, Forming Trust in EMentoring: A Research Agenda, American Journal of Industrial and Business Management, 2013.
- [5] Cavallaro, F. & Tan, K. . Computer-Mediated Peer-toPeer Mentoring. AACE Journal, 14(2), 129138. Chesapeake, VA: Association for the Advancement of Computing in Education (AACE)-2006

SOURCE CODE

app.js

login check:

```
from django.http import HttpResponseRedirect
from django.urls import reverse
from django.utils.deprecation import MiddlewareMixin


class LoginCheckMiddleWare(MiddlewareMixin):

    def process_view(self,request,view_func,view_args,view_kwargs):
        modulename=view_func.__module__
        print(modulename)
        user=request.user
        if user.is_authenticated:
            if user.user_type == "1":
                if modulename == "student_management_app.HodViews":
                    pass
                        elif modulename == "student_management_app.views" or modulename == "django.views.static":
                    pass
                                elif modulename == "django.contrib.auth.views" or modulename =="django.contrib.admin.sites":
                    pass
                else:
                    return HttpResponseRedirect(reverse("admin_home"))
            elif user.user_type == "2":
                        if modulename == "student_management_app.StaffViews" or modulename == "student_management_app.EditResultVIewClass":
                    pass
```

```
                    elif modulename == "student_management_app.views" or modulename ==
"django.views.static":
                pass
            else:
                return HttpResponseRedirect(reverse("staff_home"))
        elif user.user_type == "3":
                if modulename == "student_management_app.StudentViews" or modulename ==
"django.views.static":
                pass
            elif modulename == "student_mana3
gement_app.views":
                pass
            else:
                return HttpResponseRedirect(reverse("student_home"))
        else:
            return HttpResponseRedirect(reverse("show_login"))

    else:
            if request.path == reverse("show_login") or request.path == reverse("do_login") or
modulename == "django.contrib.auth.views" or modulename =="django.contrib.admin.sites" or
modulename=="student_management_app.views":
            pass
        else:
            return HttpResponseRedirect(reverse("show_login"))

STAFF VIEWS
import json
from datetime import datetime
from uuid import uuid4

from django.contrib import messages
```

```
from django.core import serializers
from django.forms import model_to_dict
from django.http import HttpResponse, JsonResponse, HttpResponseRedirect
from django.shortcuts import render
from django.urls import reverse
from django.views.decorators.csrf import csrf_exempt

from student_management_app.models import Subjects, SessionYearModel, Students, Attendance, AttendanceReport, \
    LeaveReportStaff, Staffs, FeedBackStaffs, CustomUser, Courses, NotificationStaffs, StudentResult, OnlineClassRoom


def staff_home(request):
    #For Fetch All Student Under Staff
    subjects=Subjects.objects.filter(staff_id=request.user.id)
    course_id_list=[]
    for subject in subjects:
        course=Courses.objects.get(id=subject.course_id.id)
        course_id_list.append(course.id)

    final_course=[]
    #removing Duplicate Course ID
    for course_id in course_id_list:
        if course_id not in final_course:
            final_course.append(course_id)

    students_count=Students.objects.filter(course_id__in=final_course).count()

    #Fetch All Attendance Count
    attendance_count=Attendance.objects.filter(subject_id__in=subjects).count()
```

```
    #Fetch All Approve Leave
    staff=Staffs.objects.get(admin=request.user.id)
    leave_count=LeaveReportStaff.objects.filter(staff_id=staff.id,leave_status=1).count()
    subject_count=subjects.count()

    #Fetch Attendance Data by Subject
    subject_list=[]
    attendance_list=[]
    for subject in subjects:
        attendance_count1=Attendance.objects.filter(subject_id=subject.id).count()
        subject_list.append(subject.subject_name)
        attendance_list.append(attendance_count1)

    students_attendance=Students.objects.filter(course_id__in=final_course)
    student_list=[]
    student_list_attendance_present=[]
    student_list_attendance_absent=[]
    for student in students_attendance:

attendance_present_count=AttendanceReport.objects.filter(status=True,student_id=student.id).co
unt()

attendance_absent_count=AttendanceReport.objects.filter(status=False,student_id=student.id).co
unt()
        student_list.append(student.admin.username)
        student_list_attendance_present.append(attendance_present_count)
        student_list_attendance_absent.append(attendance_absent_count)

                                                                    return
render(request,"staff_template/staff_home_template.html",{"students_count":students_count,"att
```

```
endance_count":attendance_count,"leave_count":leave_count,"subject_count":subject_count,"su
bject_list":subject_list,"attendance_list":attendance_list,"student_list":student_list,"present_list":
student_list_attendance_present,"absent_list":student_list_attendance_absent})

def staff_take_attendance(request):
    subjects=Subjects.objects.filter(staff_id=request.user.id)
    session_years=SessionYearModel.object.all()
                                                                                return
render(request,"staff_template/staff_take_attendance.html",{"subjects":subjects,"session_years":
session_years})

@csrf_exempt
def get_students(request):
    subject_id=request.POST.get("subject")
    session_year=request.POST.get("session_year")

    subject=Subjects.objects.get(id=subject_id)
    session_model=SessionYearModel.object.get(id=session_year)
    students=Students.objects.filter(course_id=subject.course_id,session_year_id=session_model)
    list_data=[]

    for student in students:
                                data_small={"id":student.admin.id,"name":student.admin.first_name+"
"+student.admin.last_name}
        list_data.append(data_small)
    return JsonResponse(json.dumps(list_data),content_type="application/json",safe=False)

@csrf_exempt
def save_attendance_data(request):
    student_ids=request.POST.get("student_ids")
    subject_id=request.POST.get("subject_id")
```

```
    attendance_date=request.POST.get("attendance_date")
    session_year_id=request.POST.get("session_year_id")

    subject_model=Subjects.objects.get(id=subject_id)
    session_model=SessionYearModel.object.get(id=session_year_id)
    json_sstudent=json.loads(student_ids)
    #print(data[0]['id'])



    try:

attendance=Attendance(subject_id=subject_model,attendance_date=attendance_date,session_yea
r_id=session_model)
        attendance.save()

        for stud in json_sstudent:
            student=Students.objects.get(admin=stud['id'])

attendance_report=AttendanceReport(student_id=student,attendance_id=attendance,status=stud['
status'])
            attendance_report.save()
        return HttpResponse("OK")
    except:
        return HttpResponse("ERR")

def staff_update_attendance(request):
    subjects=Subjects.objects.filter(staff_id=request.user.id)
    session_year_id=SessionYearModel.object.all()
                                                                                return
render(request,"staff_template/staff_update_attendance.html",{"subjects":subjects,"session_year
_id":session_year_id})
```

```
@csrf_exempt
def get_attendance_dates(request):
    subject=request.POST.get("subject")
    session_year_id=request.POST.get("session_year_id")
    subject_obj=Subjects.objects.get(id=subject)
    session_year_obj=SessionYearModel.object.get(id=session_year_id)

attendance=Attendance.objects.filter(subject_id=subject_obj,session_year_id=session_year_obj)
    attendance_obj=[]
    for attendance_single in attendance:

data={"id":attendance_single.id,"attendance_date":str(attendance_single.attendance_date),"sessi
on_year_id":attendance_single.session_year_id.id}
        attendance_obj.append(data)

    return JsonResponse(json.dumps(attendance_obj),safe=False)

@csrf_exempt
def get_attendance_student(request):
    attendance_date=request.POST.get("attendance_date")
    attendance=Attendance.objects.get(id=attendance_date)

    attendance_data=AttendanceReport.objects.filter(attendance_id=attendance)
    list_data=[]

    for student in attendance_data:

data_small={"id":student.student_id.admin.id,"name":student.student_id.admin.first_name+"
"+student.student_id.admin.last_name,"status":student.status}
        list_data.append(data_small)
```

```
    return JsonResponse(json.dumps(list_data),content_type="application/json",safe=False)

@csrf_exempt
def save_updateattendance_data(request):
    student_ids=request.POST.get("student_ids")
    attendance_date=request.POST.get("attendance_date")
    attendance=Attendance.objects.get(id=attendance_date)

    json_sstudent=json.loads(student_ids)



    try:
        for stud in json_sstudent:
            student=Students.objects.get(admin=stud['id'])

attendance_report=AttendanceReport.objects.get(student_id=student,attendance_id=attendance)
            attendance_report.status=stud['status']
            attendance_report.save()
        return HttpResponse("OK")
    except:
        return HttpResponse("ERR")

def staff_apply_leave(request):
    staff_obj = Staffs.objects.get(admin=request.user.id)
    leave_data=LeaveReportStaff.objects.filter(staff_id=staff_obj)
    return render(request,"staff_template/staff_apply_leave.html",{"leave_data":leave_data})

def staff_apply_leave_save(request):
    if request.method!="POST":
        return HttpResponseRedirect(reverse("staff_apply_leave"))
    else:
```

```
        leave_date=request.POST.get("leave_date")
        leave_msg=request.POST.get("leave_msg")

        staff_obj=Staffs.objects.get(admin=request.user.id)
        try:

leave_report=LeaveReportStaff(staff_id=staff_obj,leave_date=leave_date,leave_message=leave_
msg,leave_status=0)
            leave_report.save()
            messages.success(request, "Successfully Applied for Leave")
            return HttpResponseRedirect(reverse("staff_apply_leave"))
        except:
            messages.error(request, "Failed To Apply for Leave")
            return HttpResponseRedirect(reverse("staff_apply_leave"))



def staff_feedback(request):
    staff_id=Staffs.objects.get(admin=request.user.id)
    feedback_data=FeedBackStaffs.objects.filter(staff_id=staff_id)
    return render(request,"staff_template/staff_feedback.html",{"feedback_data":feedback_data})

def staff_feedback_save(request):
    if request.method!="POST":
        return HttpResponseRedirect(reverse("staff_feedback_save"))
    else:
        feedback_msg=request.POST.get("feedback_msg")

        staff_obj=Staffs.objects.get(admin=request.user.id)
        try:

feedback=FeedBackStaffs(staff_id=staff_obj,feedback=feedback_msg,feedback_reply="")
```

```
            feedback.save()
            messages.success(request, "Successfully Sent Feedback")
            return HttpResponseRedirect(reverse("staff_feedback"))
        except:
            messages.error(request, "Failed To Send Feedback")
            return HttpResponseRedirect(reverse("staff_feedback"))

def staff_profile(request):
    user=CustomUser.objects.get(id=request.user.id)
    staff=Staffs.objects.get(admin=user)
    return render(request,"staff_template/staff_profile.html",{"user":user,"staff":staff})

def staff_profile_save(request):
    if request.method!="POST":
        return HttpResponseRedirect(reverse("staff_profile"))
    else:
        first_name=request.POST.get("first_name")
        last_name=request.POST.get("last_name")
        address=request.POST.get("address")
        password=request.POST.get("password")
        try:
            customuser=CustomUser.objects.get(id=request.user.id)
            customuser.first_name=first_name
            customuser.last_name=last_name
            if password!=None and password!="":
                customuser.set_password(password)
            customuser.save()

            staff=Staffs.objects.get(admin=customuser.id)
            staff.address=address
            staff.save()
```

```
            messages.success(request, "Successfully Updated Profile")
            return HttpResponseRedirect(reverse("staff_profile"))
        except:
            messages.error(request, "Failed to Update Profile")
            return HttpResponseRedirect(reverse("staff_profile"))

@csrf_exempt
def staff_fcmtoken_save(request):
    token=request.POST.get("token")
    try:
        staff=Staffs.objects.get(admin=request.user.id)
        staff.fcm_token=token
        staff.save()
        return HttpResponse("True")
    except:
        return HttpResponse("False")

def staff_all_notification(request):
    staff=Staffs.objects.get(admin=request.user.id)
    notifications=NotificationStaffs.objects.filter(staff_id=staff.id)
    return render(request,"staff_template/all_notification.html",{"notifications":notifications})

def staff_add_result(request):
    subjects=Subjects.objects.filter(staff_id=request.user.id)
    session_years=SessionYearModel.object.all()
                                                                                return
render(request,"staff_template/staff_add_result.html",{"subjects":subjects,"session_years":sessio
n_years})

def save_student_result(request):
    if request.method!='POST':
```

```
        return HttpResponseRedirect('staff_add_result')
    student_admin_id=request.POST.get('student_list')
    assignment_marks=request.POST.get('assignment_marks')
    exam_marks=request.POST.get('exam_marks')
    subject_id=request.POST.get('subject')


    student_obj=Students.objects.get(admin=student_admin_id)
    subject_obj=Subjects.objects.get(id=subject_id)

    try:

check_exist=StudentResult.objects.filter(subject_id=subject_obj,student_id=student_obj).exists()
        if check_exist:
            result=StudentResult.objects.get(subject_id=subject_obj,student_id=student_obj)
            result.subject_assignment_marks=assignment_marks
            result.subject_exam_marks=exam_marks
            result.save()
            messages.success(request, "Successfully Updated Result")
            return HttpResponseRedirect(reverse("staff_add_result"))
        else:

result=StudentResult(student_id=student_obj,subject_id=subject_obj,subject_exam_marks=exa
m_marks,subject_assignment_marks=assignment_marks)
            result.save()
            messages.success(request, "Successfully Added Result")
            return HttpResponseRedirect(reverse("staff_add_result"))
    except:
        messages.error(request, "Failed to Add Result")
        return HttpResponseRedirect(reverse("staff_add_result"))
```

```
@csrf_exempt
def fetch_result_student(request):
    subject_id=request.POST.get('subject_id')
    student_id=request.POST.get('student_id')
    student_obj=Students.objects.get(admin=student_id)
    result=StudentResult.objects.filter(student_id=student_obj.id,subject_id=subject_id).exists()
    if result:
        result=StudentResult.objects.get(student_id=student_obj.id,subject_id=subject_id)

result_data={"exam_marks":result.subject_exam_marks,"assign_marks":result.subject_assignment_marks}
        return HttpResponse(json.dumps(result_data))
    else:
        return HttpResponse("False")

def start_live_classroom(request):
    subjects=Subjects.objects.filter(staff_id=request.user.id)
    session_years=SessionYearModel.object.all()
                                                                                return render(request,"staff_template/start_live_classroom.html",{"subjects":subjects,"session_years":session_years})

def start_live_classroom_process(request):
    session_year=request.POST.get("session_year")
    subject=request.POST.get("subject")

    subject_obj=Subjects.objects.get(id=subject)
    session_obj=SessionYearModel.object.get(id=session_year)

checks=OnlineClassRoom.objects.filter(subject=subject_obj,session_years=session_obj,is_active=True).exists()
```

```
    if checks:

data=OnlineClassRoom.objects.get(subject=subject_obj,session_years=session_obj,is_active=Tr
ue)
        room_pwd=data.room_pwd
        roomname=data.room_name
    else:
        room_pwd=datetime.now().strftime('%Y%m-%d%H-%M%S-') + str(uuid4())
        roomname=datetime.now().strftime('%Y%m-%d%H-%M%S-') + str(uuid4())
        staff_obj=Staffs.objects.get(admin=request.user.id)

onlineClass=OnlineClassRoom(room_name=roomname,room_pwd=room_pwd,subject=subject_
obj,session_years=session_obj,started_by=staff_obj,is_active=True)
        onlineClass.save()

                                                                                return
render(request,"staff_template/live_class_room_start.html",{"username":request.user.username,"
password":room_pwd,"roomid":roomname,"subject":subject_obj.subject_name,"session_year":s
ession_obj})


def returnHtmlWidget(request):
    return render(request,"widget.html")
```

ACKNOWLEDGEMENTS

The making of the project “Mentor Mentee Management System” involves the contribution of many people. We would like to convey our sincere thanks to Dr. S.M. Khot, principal Fr. C Rodrigues Institute of Technology, Vashi for giving us the opportunity to showcase our skills and providing us with the necessary resources. We would also like to convey our heartfelt gratitude to the Head of Department of Information Technology, Prof. Vaishali Bodade for her constant support and motivation. We express deep gratitude to our project guide and mentor Mrs. Poonam Bari for her constant motivation to think out of the box and immense contribution throughout this project. Last but not the least, we convey our heartfelt thanks to the project coordinator, Prof Lakshmi Gadhikar for supporting and guiding us throughout the process. We also extend our heartfelt thanks to our families and well-wishers.

Yours sincerely.

Janhavi Patil (5020139)

Hishali Rana (5020146)

Akhil Suryam (5020149)

Divya Vinod (5020164)