Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester-IV

Subject:-Software Engineering Project

Project title: College Automation And Scheduling System

By:

1) Aneri Dhola roll no: CE026, Id: 19CEUES043

2) Khushi Doshi roll no: CE030, Id: 19CEUEG038

Guided by: Prof. Pinkal C. Chauhan.
Prof. Brijesh S. Bhatt
Prof. Jigar M. Pandya

Contents:-

1.)Abstract:

Online College Automation And Scheduling System(OCAASS) provides a simple interface for maintenance of student information. It can be used by educational institutes or colleges to maintain the records of students easily. The creation and management of accurate, up-to-date information regarding a students' academic career is critically important in the university as well as colleges. Student information system deals with all kind of student details, academic related reports,course details,batch details, and other resource related details too. It will also have faculty details. It also facilitate us explore all the activities happening in the college, Different reports and Queries can be generated based on vast options related to students, batch, course, faculty.

2.)Introduction:

The design and implementation of a comprehensive Online College Automation And Scheduling System and user interface is to replace the current paper records.The system utilizes user authentication, displaying only information necessary for an individual's duties.Additionally, each sub-system has authentication allowing authorized users to create or update information in that subsystem. All data is thoroughly reviewed and validated on the server before actual record alteration occurs. In addition to a staff user interface, the system plans for student user interface, allowing users to access information and submit requests online thus reducing processing time. All data is stored securely on SQLservers managed by the college administrator. The system features a complex logging system to track all users' access and ensure conformity to data access guidelines and is expected to increase the efficiency of the college's record management thereby decreasing the work hours needed to access and deliver student records to users.

3.) Software Requirement Specifications

College Automation And Scheduling System:-

R.1 Login management:-

R.1.1.Login:-

Description:-User can login with their valid email-id and password.After login they can redirect to the home page according to user type.

Input:-Details

Output:-redirect to the home-page.

R.1.2.Reset Password:-

Description:-If user forgot password or want to reset then they can change the password.

Input:-Email-Id.

Output:-get mail(mail has link through that link we can change the password).

R.2 Admin:-

R.2.1.Edit profile:-

Description:-Admin can edit their basic info.

Input:-Edit values.

Output:-save changes.

R.2.2.Add Staff:-

Description:-Admin can add staff here

Input:-Staff basic details.

Output:-Save staff details successfully/not.

R.2.3.View/manage staff:-

Description:-view staff which is save in database.

Input:-select a staff which we want to edit

output:-redirect to the edit page

R.2.3.1 Edit staff:-

Description:-Admin can edit staff's detail here.

Input:-Edit values.

Output:-Save Changes.

R.2.4.Add Student:-

Description:-Admin can add student here

Input:-Student basic details.

Output:-Save student details successfully/not.

R.2.5.View/manage student:-

Description:-view student which is save in database.

Input:-select a student which we want to edit

output:-redirect to the edit page

R.2.5.1 Edit student:-

Description:-Admin can edit students' detail here.

Input:-Edit values.

Output:-Save Changes.

R.2.6.Add Subject:-

Description:-Admin can add subject here

Input:-Subject basic details.

Output:-Save subject details successfully/not.

R.2.7.View/manage subject:-

Description:-view subject which is save in database.

Input:-select a subject which we want to edit

output:-redirect to the edit page

R.2.7.1 Edit subject:-

Description:-Admin can edit subject's detail here.

Input:-Edit values.

Output:-Save Changes.

R.2.8.Add Course:-

Description:-Admin can add course here

Input:-Course basic details.

Output:-Save course details successfully/not.

R.2.9.View/manage course:-

Description:-view course which is save in database.

Input:-select a student which we want to edit

output:-redirect to the edit page

R.2.9.1 Edit course:-

Description:-Admin can edit course detail here.

Input:-Edit values.

Output:-Save Changes.

R.2.10.View attendance:-

Description:-Here admin can view attendance of student.

Input:-select subject and fetch attendance date.

Output:-list of date.

Input:-select date and fetch student

Output:-student list with their attendance.

R.2.11.Student feedback:-

Description:-here admin can see the feedback from student and gev reply.

Input:-select student=> get pop-up menu for reply=>type reply=>Send Reply.

Output:-send reply/get error,

R.2.12.Staff feedback:-

Description:-here admin can see the feedback from staff and gev reply.

Input:-select staff=> get pop-up menu for reply=>type reply=>Send Reply.

Output:-send reply/get error.

R.2.13.Student leave:-

Description:-here admin can approve or disapprove leave for student.

Input:-select student=> click approve/disapprove
Output:-approve/disapprove(see status).

R.2.14.Staff leave:-

Description:-here admin can approve or disapprove leave for staff.

Input:-select staff=> click approve/disapprove
Output:-approve/disapprove(see status).

R.3 Staff:-

R.3.1.Edit profile:-

Description:-staff can edit their basic info.

Input:-Edit values.

Output:-save changes.

R.3.2.ApplyLeave:-

Description:-Staff can apply for leave.

Input:-Date And reason=>apply for leave.

Output:-sent leave to admin.

R.3.3.Feddback:-

Description:-Staff can gave feedback to the admin.

Input:-feedback(message)=>Leave Feedback

Output:-sent feedback to the admin.

R.3.4.AddResult:-

Description:-Staff can add result for student.

Input:-select subject=>fetchStudent=>SelectStudent=>write marks=>Save result.

Output:-save/update result successfully/not.

R.3.5.TakeAttendance:-

Description:-Staff can take attendance of students.

Input:-select subject=>fetch student=>give date=>mark attendance.

Output:-save attendance/error.

R.3.6.UpdateAttendance:-

Description:-Staff can update attendance of students.

Input:-select subject=>fetch date=>select date=>fetch student=>mark attendance.

Output:-save attendance/error.

R.4 Student:-

R.4.1.Edit profile:-

Description:-student can edit their basic info.

Input:-Edit values.

Output:-save changes.

R.4.2.ApplyLeave:-

Description:-Student can apply for leave.

Input:-Date And reason=>applyforleave.

Output:-sent leave to admin.

R.4.3.Feddback:-

Description:-Student can gave feedback to the admin.

Input:-feedback(message)=>Leavefeedback

Output:-sent feedback to the admin.

R.4.4.Grade card:-

Description:-View marks.

Output:-Subject-Marks-Status.

R.4.5.ViewAttendance:-

Description:-View attendance here.

Input:-subject-start to end date

Output:-can see attendance date wise/error.

Functional Requirements of the College Management System:-

The functional requirements of this system are:-

Register new students.

Record the attendance of students.

Record the internal marks of students.

Record the feedback details of students.

Record the feedback details of staffs.

Register a new staffs.

Record the course details and subject information.

Record attendance of students/marks.

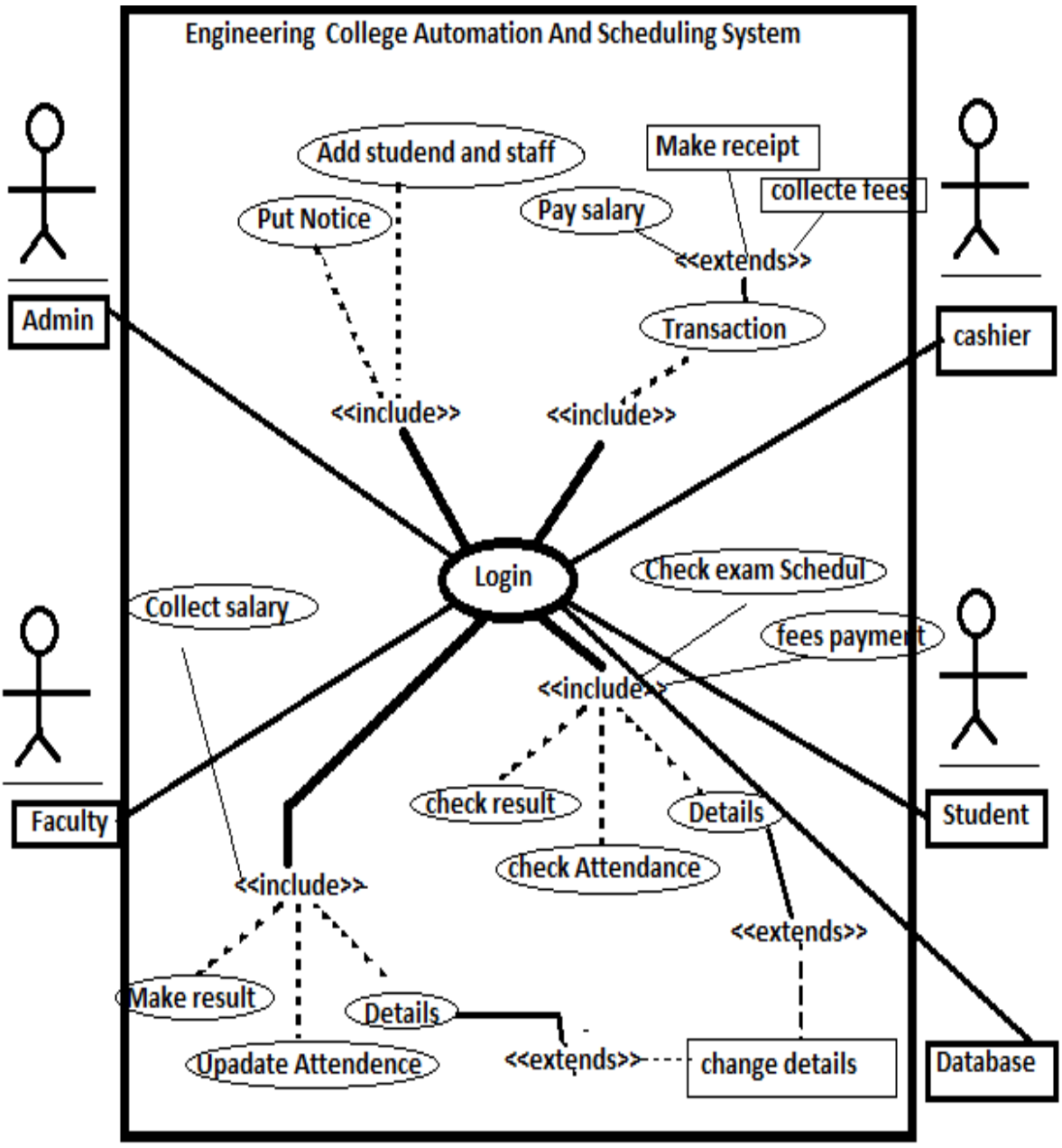
Non- Functional Requirements of the College Management System

In this system, the authentication of the user is an important factor. In this system, user authentication will be done by login by user name and password and classified by user type. Users will get access to the system as permissions are classified for that type of user.

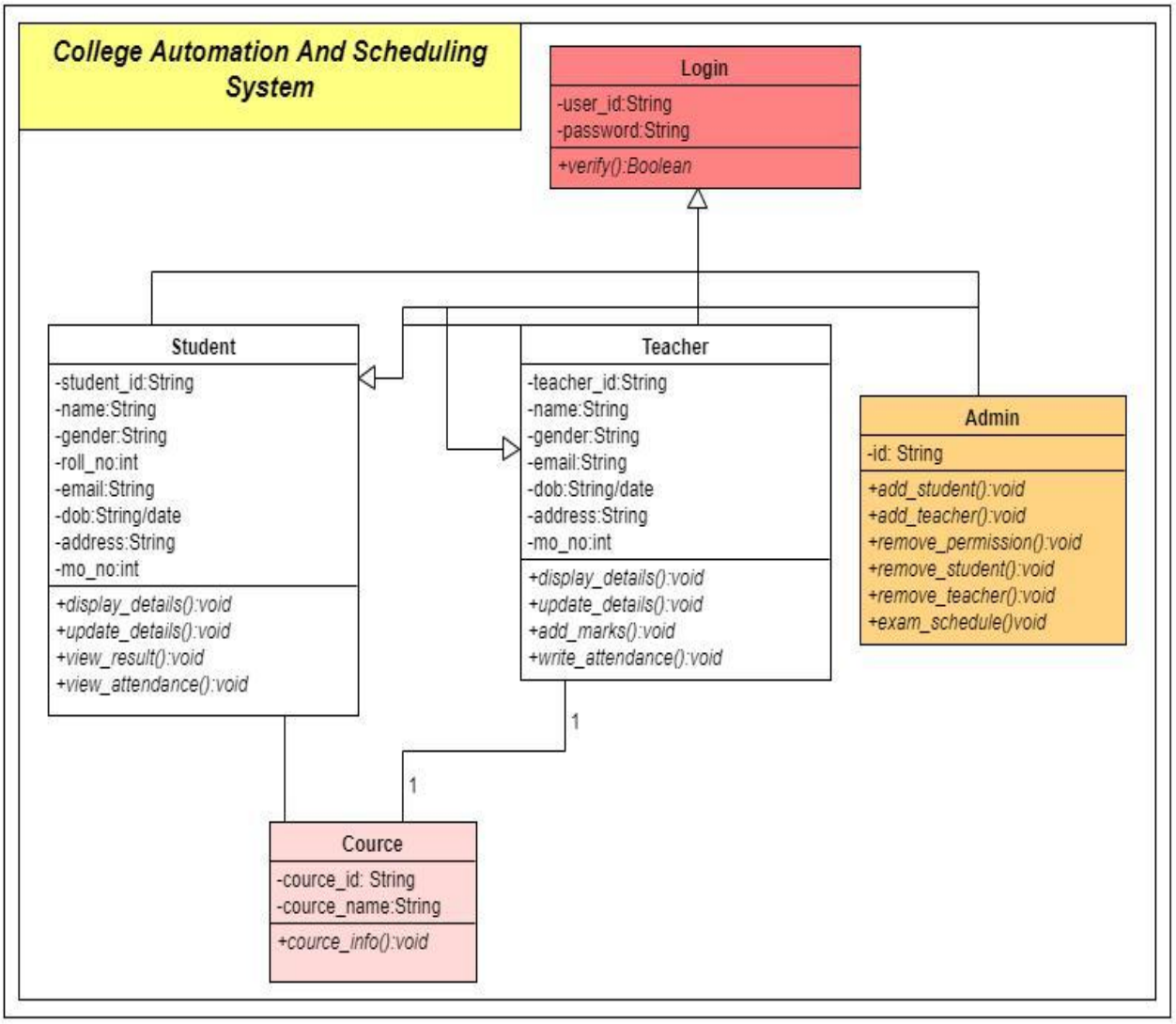
The system has a consistent interface so that the system is easy to use and in the interface of our system buttons and forms are used to enter data related to a specific module.

4)Design:-

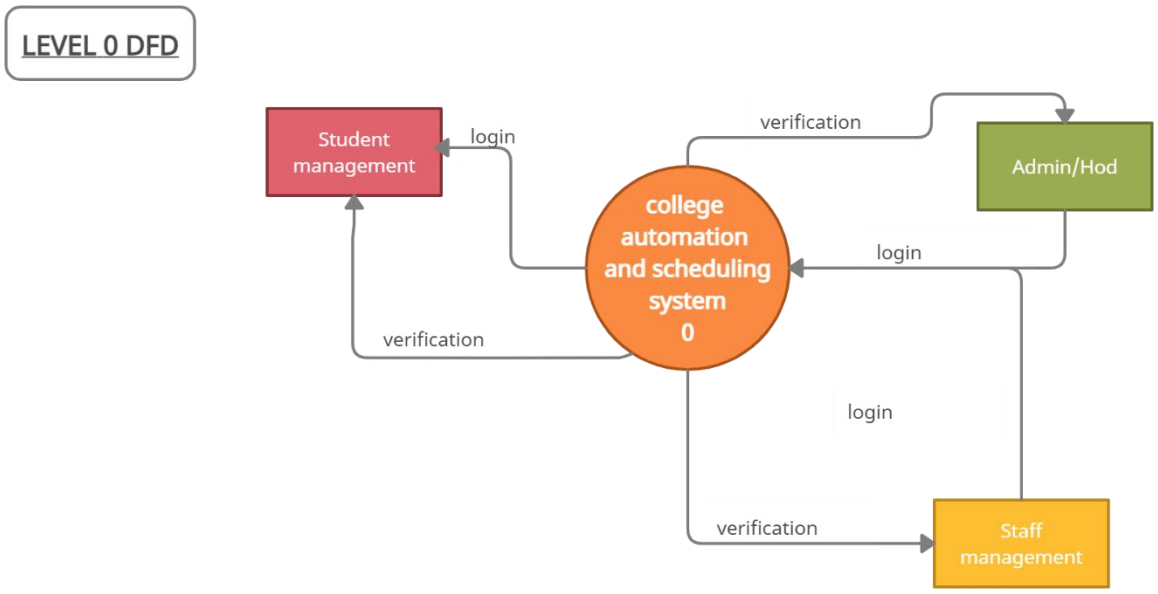
4.1) Use Case Diagram:

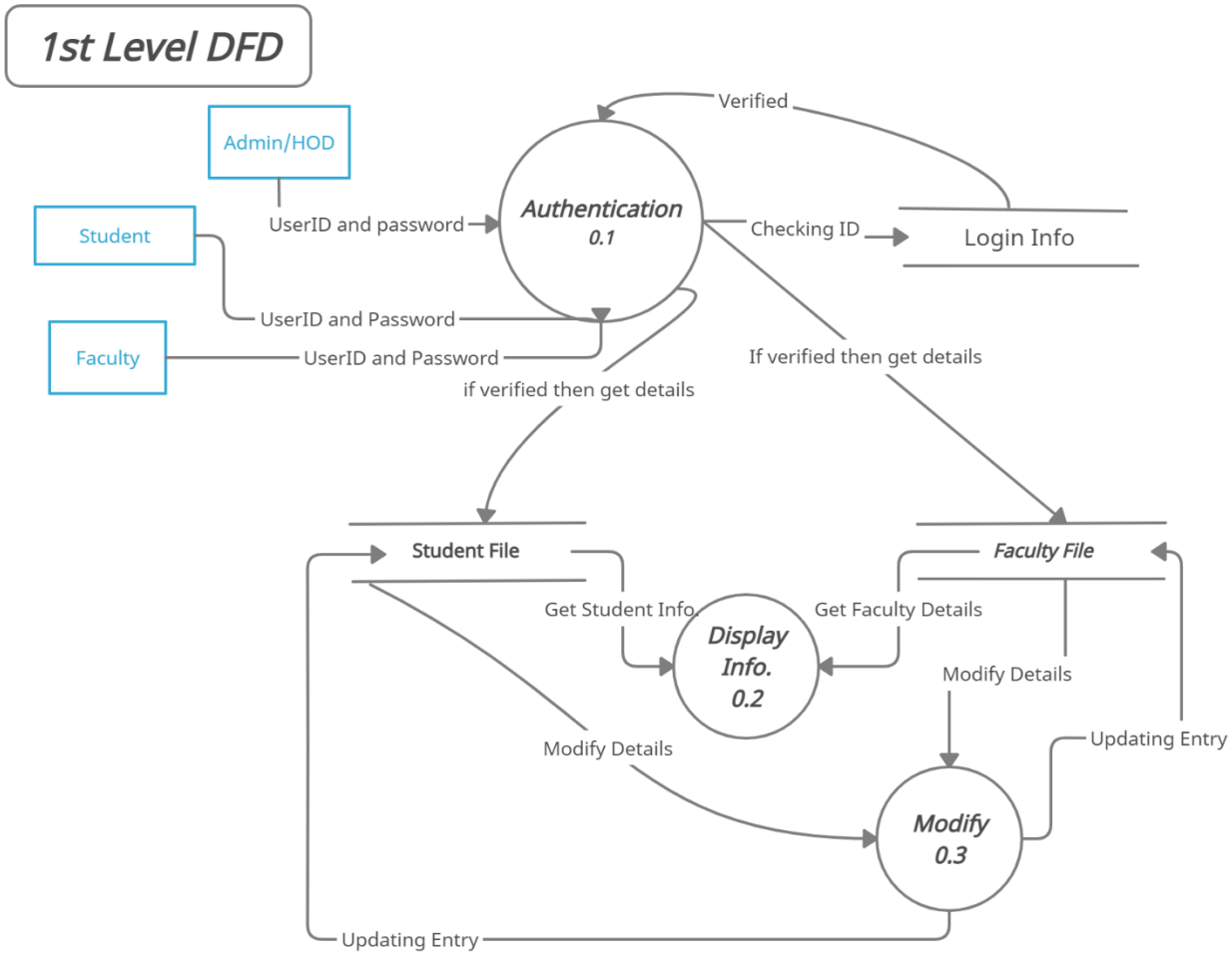


4.2)Class Diagram:

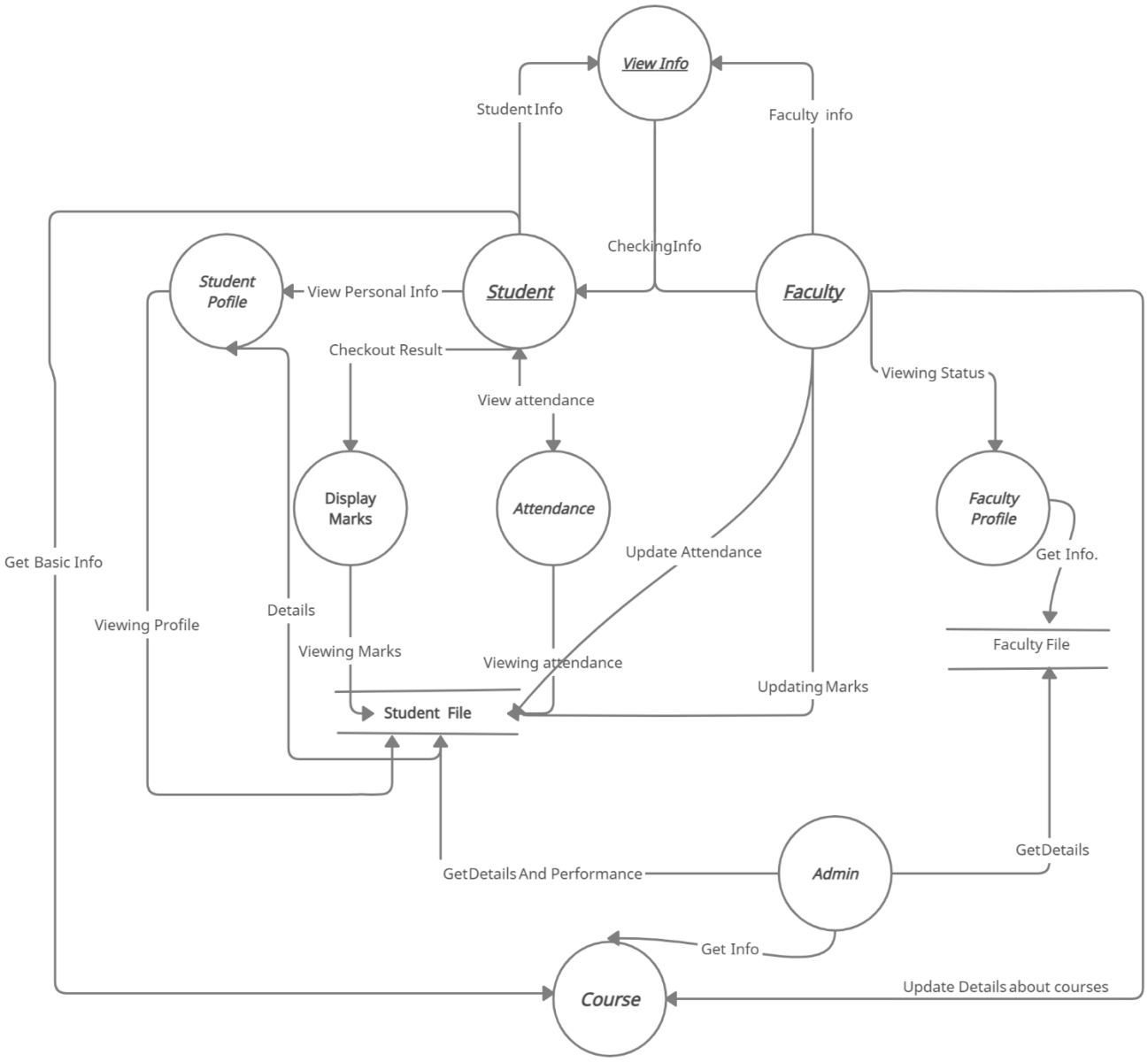


4.3)Data Flow Diagram:

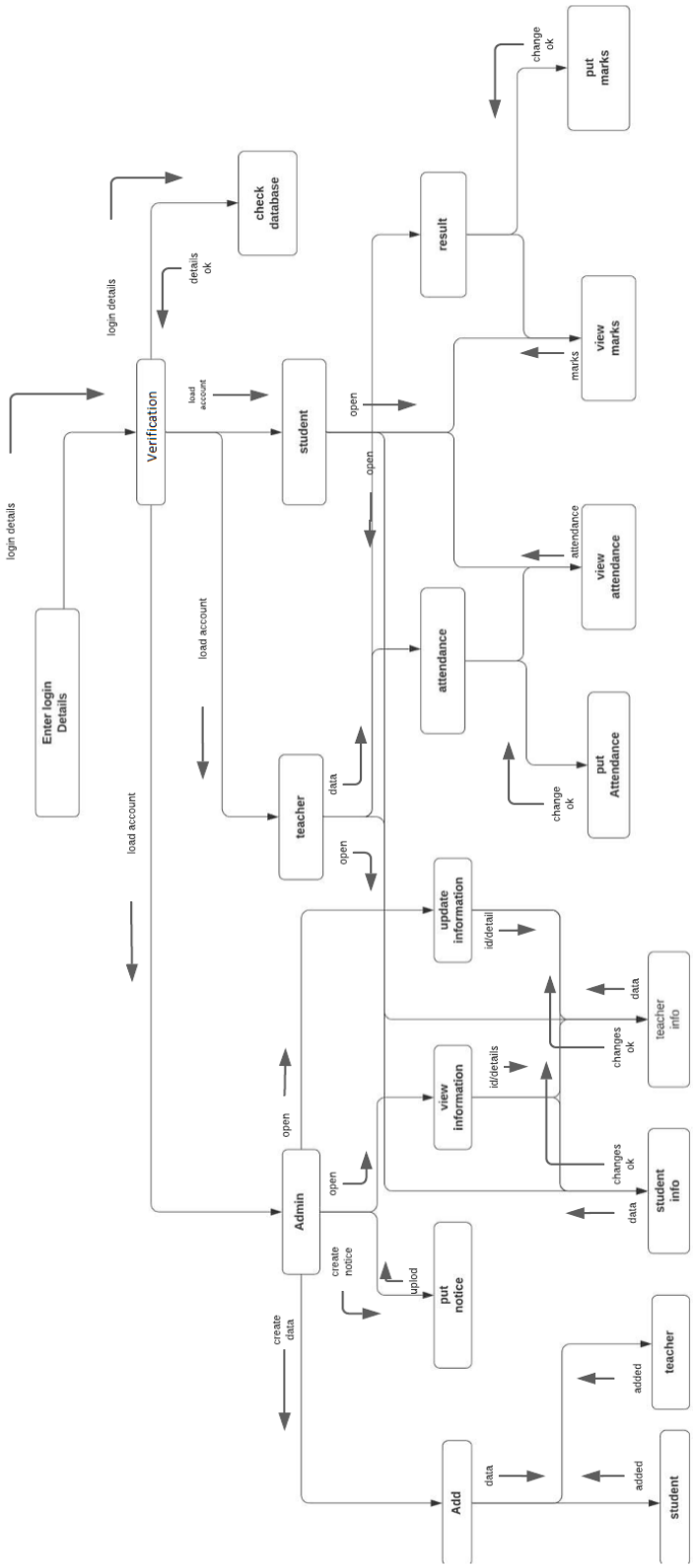




2ND Level Diagram:

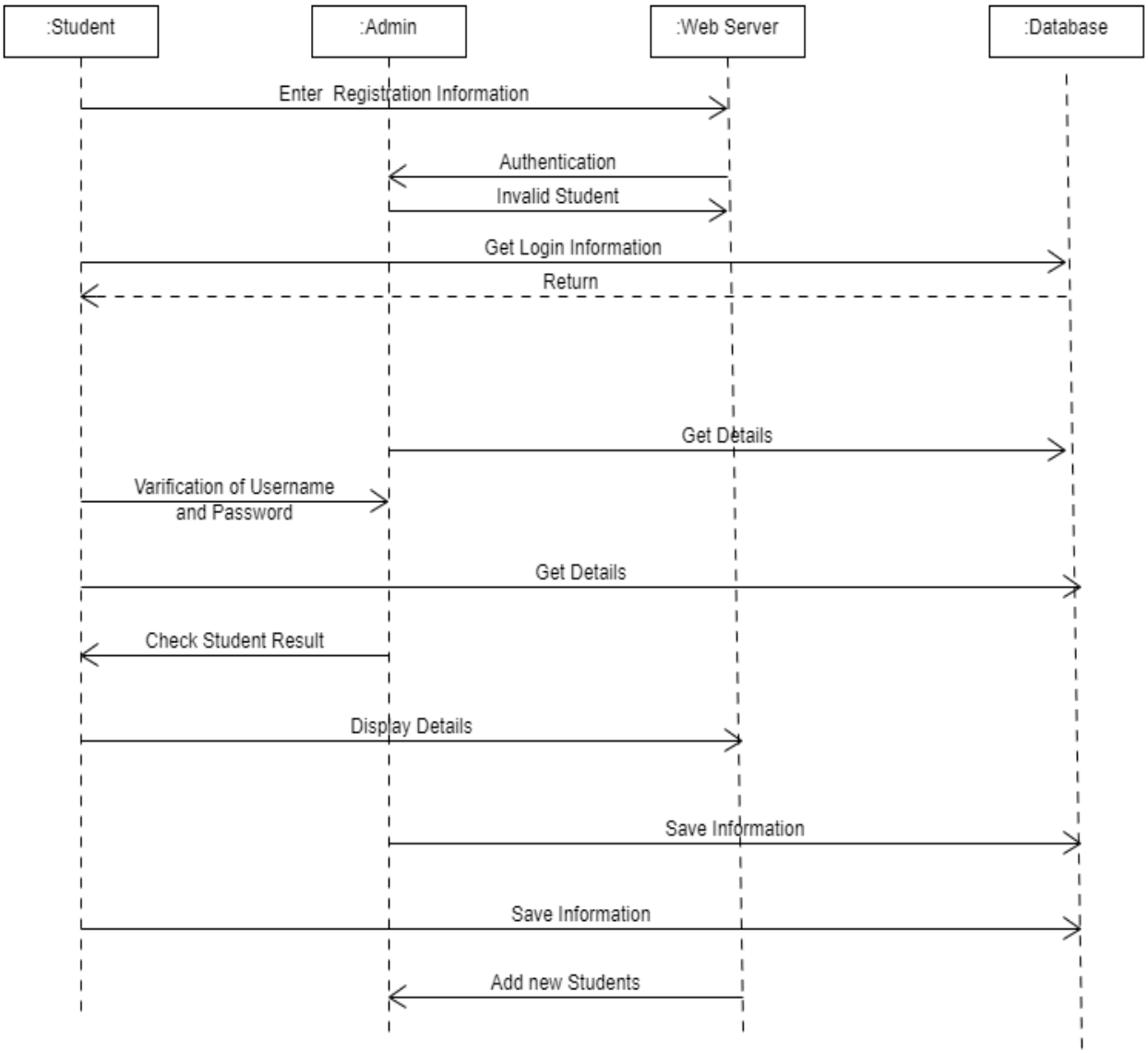


4.4)Structure Chart:

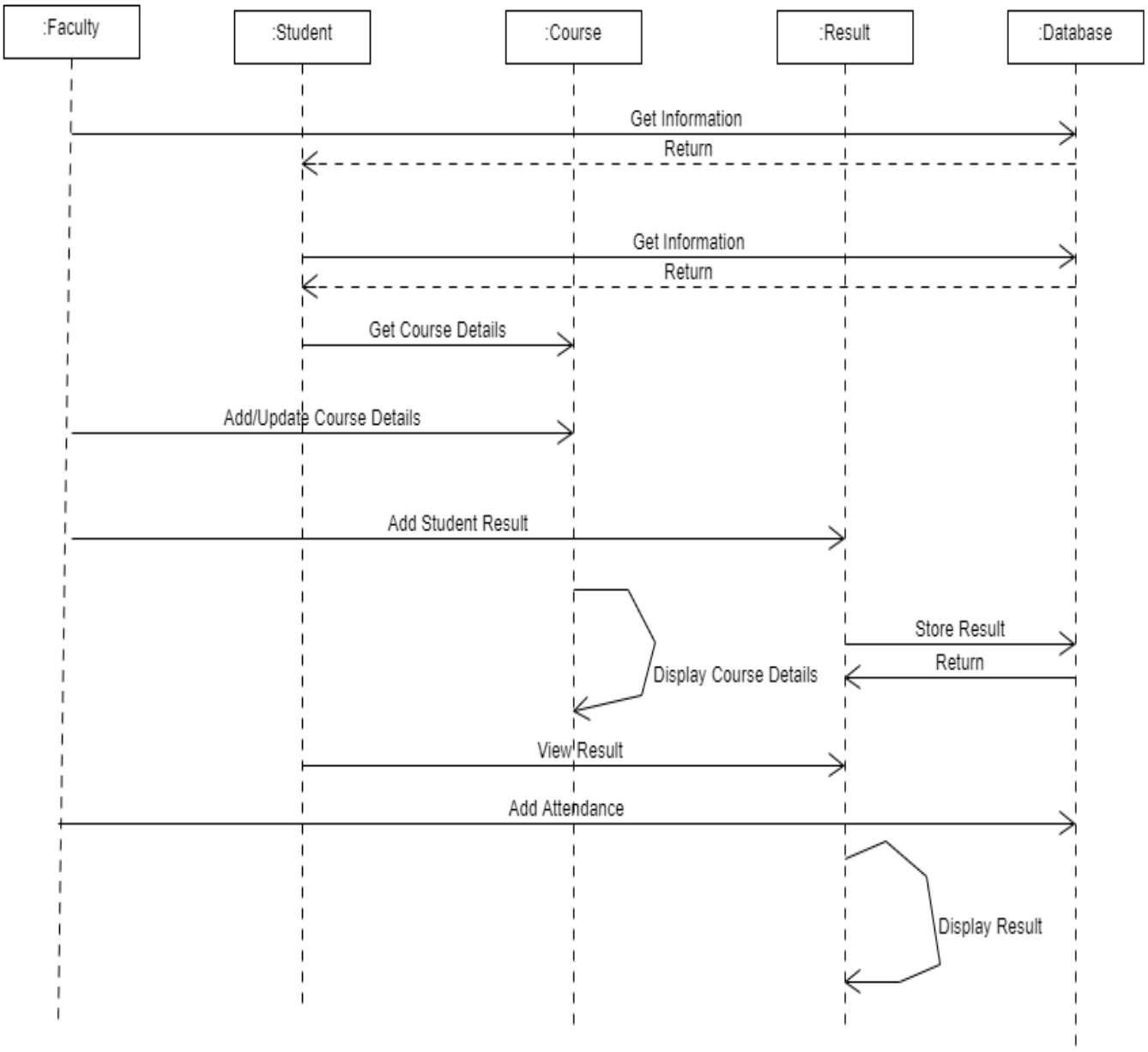


4.5)Sequence Diagram:

1St):-

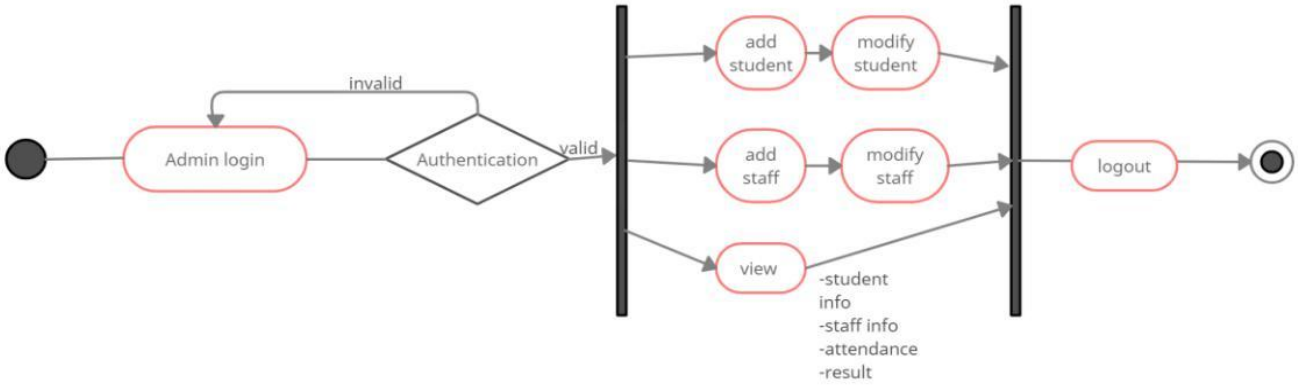


2nd):-

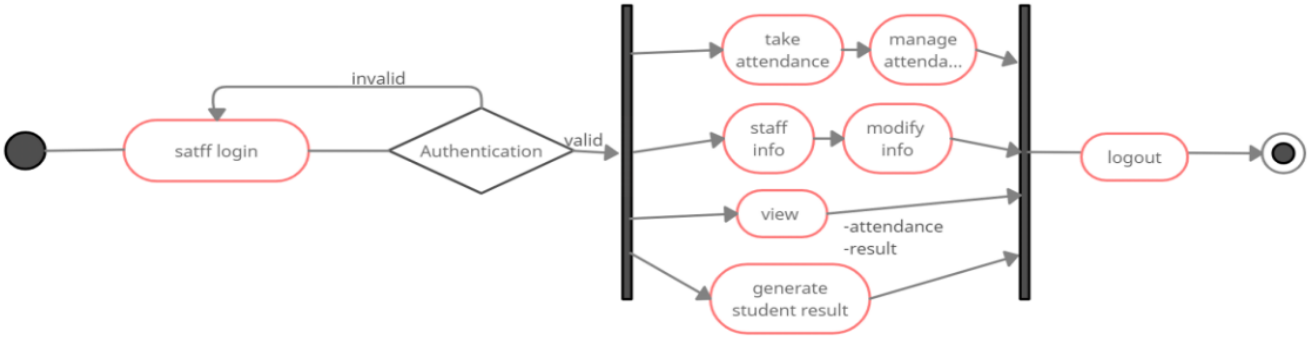

:Faculty
:Student
:Course
:Result
:Database
Get Information
Return
Get Information
Return
Get Course Details
Add/Update Course Details
Add Student Result
Store Result
Return
Display Course Details
View Result
Add Attendance
Display Result

4.6)Activity diagrams:

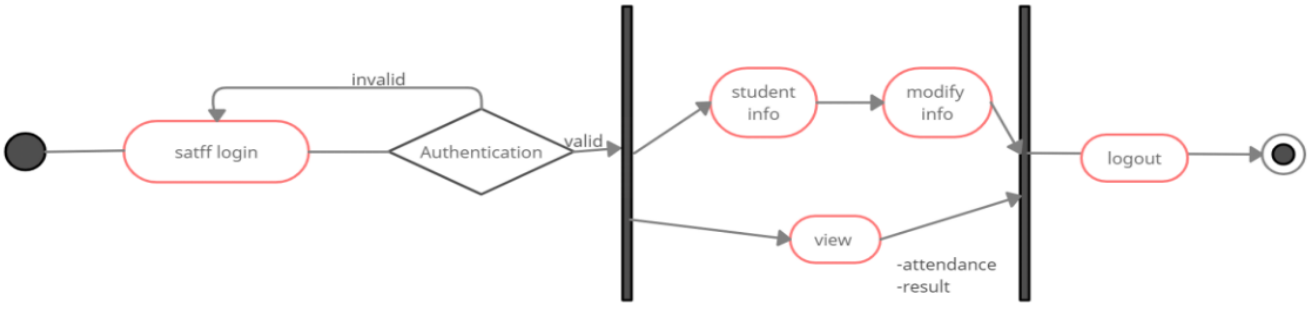
Admin activity diagram



satff activity diagram



student activity diagram



4.7)Data Dictionary:

4.7.1)Custom User:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
password	varchar(128)	NO		NULL	
last_login	datetime(6)	YES		NULL	
is_superuser	tinyint(1)	NO		NULL	
username	varchar(150)	NO	UNI	NULL	
first_name	varchar(150)	NO		NULL	
last_name	varchar(150)	NO		NULL	
email	varchar(254)	NO		NULL	
is_staff	tinyint(1)	NO		NULL	
is_active	tinyint(1)	NO		NULL	
date_joined	datetime(6)	NO		NULL	
user_type	varchar(10)	NO		NULL	

4.7.2)AdminHod:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
admin_id	int(11)	NO	UNI	NULL	

4.7.3)Staffs:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
address	longtext	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
admin_id	int(11)	NO	UNI	NULL	

4.7.4)Subjects:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
subject_name	varchar(255)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
course_id_id	int(11)	NO	MUL	NULL	
staff_id_id	int(11)	NO	MUL	NULL	

4.7.5) Courses:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
course_name	varchar(255)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	

4.7.6)Students:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
gender	varchar(255)	NO		NULL	
address	longtext	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
admin_id	int(11)	NO	UNI	NULL	
course_id_id	int(11)	NO	MUL	NULL	

4.7.7)Attendance:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
attendance_date	datetime(6)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
subject_id_id	int(11)	NO	MUL	NULL	

4.7.8)AttendanceReport:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
status	tinyint(1)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
attendance_id_id	int(11)	NO	MUL	NULL	
student_id_id	int(11)	NO	MUL	NULL	

4.7.9)LeaveReportStudent:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
leave_date	varchar(255)	NO		NULL	
leave_message	longtext	NO		NULL	
leave_status	int(11)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
student_id_id	int(11)	NO	MUL	NULL	

4.7.10) LeaveReportStaff:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
leave_date	varchar(255)	NO		NULL	
leave_message	longtext	NO		NULL	
leave_status	tinyint(1)	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
staff_id_id	int(11)	NO	MUL	NULL	

4.7.11)FeedbackStudent:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
feedback	longtext	NO		NULL	
feedback_reply	longtext	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
student_id_id	int(11)	NO	MUL	NULL	

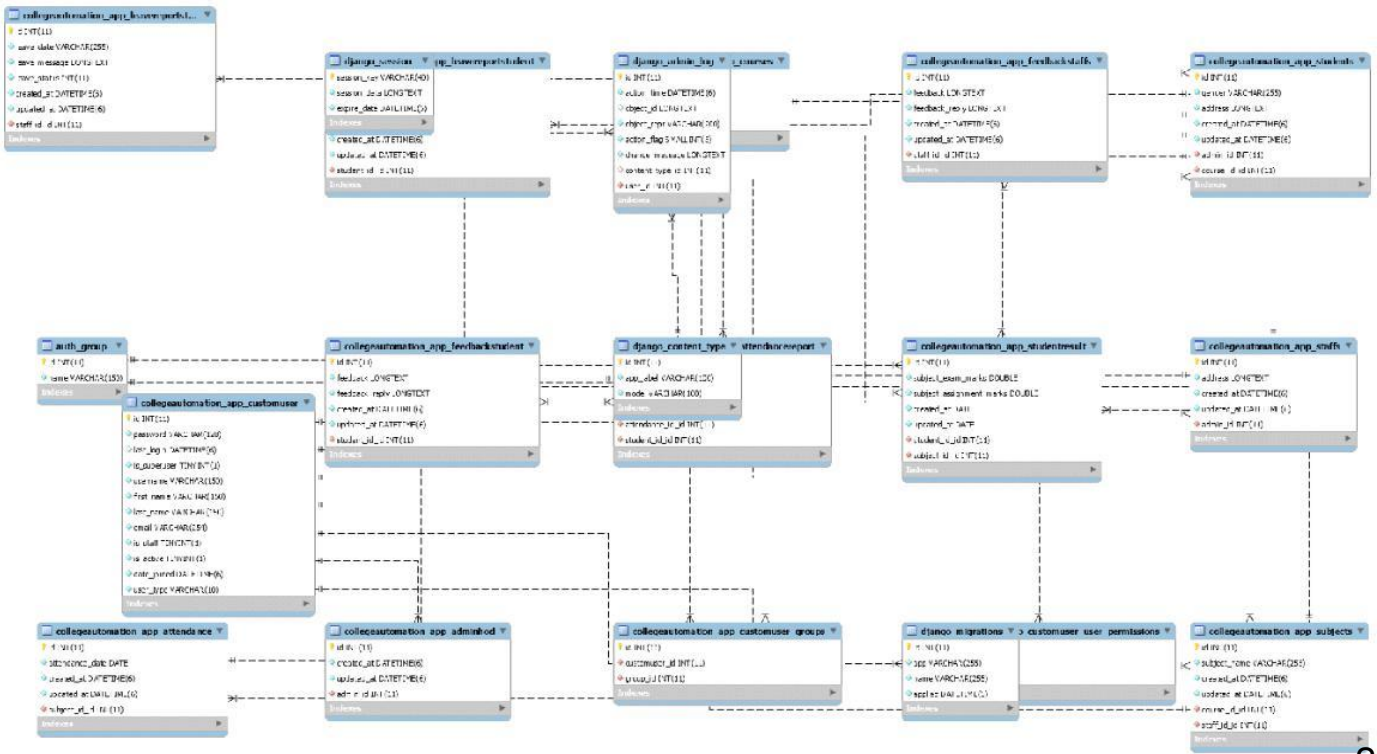
4.7.12)FeedbackStaff:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
feedback	longtext	NO		NULL	
feedback_reply	longtext	NO		NULL	
created_at	datetime(6)	NO		NULL	
updated_at	datetime(6)	NO		NULL	
staff_id_id	int(11)	NO	MUL	NULL	

4.7.13)StudentResult:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
subject_exam_marks	double	NO		NULL	
subject_assignment_marks	double	NO		NULL	
created_at	date	NO		NULL	
updated_at	date	NO		NULL	
student_id_id	int(11)	NO	MUL	NULL	
subject_id_id	int(11)	NO	MUL	NULL	

4.8)E-R Diagram:



5.)Implementation:

1. Modules:

1. ADMINISTRATOR

The administrator is responsible for entering the new student and managing the student Accounts.The administrator also manages the faulty accounts like entering a new faculty assigning the faculty to the subjects.The administrator will check all the updates i.e. student updates, faculty updates etc. The administrator has the highest level of power in the college automation and scheduling system.

Admin Modules : -

Courses	- add, change .
subject	- add,change .
Students	- add, change .
Teachers	- add, change .
Feed-back reply	-give.
attendance/report-card	-view
leave	-approve/disapprove

2. FACULTY

The staff can update the information regarding the students attendance, internal marks of the student. They can also view the student report-card details for better understanding the student performance and improving the efficiency of the student. The staff also gets the reply from the admin for feedback and leave which can sent by faculty.Faculty can update their basic information.

Teacher Modules : -

Details	-change.
FeedBack	-gave.
Leave	-apply.

Attendance -Enter the attendance of the students based on the date they are in. There is also the provision to edit the attendance student .

Marks -Enter the marks of the students based on the paper they are in. The marks of the students can also be edited.

3. STUDENT

The student is of center focus, because in every college student plays the very important role. Student can access the information of the their report card as well as attendance.As Faculty ,Student can also allow to send feedback to admin ,also get reply.Student can apply for leave to the admin and they can see the status of it.student can update their basic information.

Student Modules : -

Details -change.

FeedBack -gave.

Leave -apply.

Attendance -View the attendance status for each of your courses.

Marks -View the marks obtained for each of your courses.

2.)FunctionPrototype:

2.1)Login:(Views.py)

```
def ShowLoginPage(request):
    return render(request,"login_page.html")

def doLogin(request):
    if request.method!="POST":
        return HttpResponse("<h2>Method Not Allowed</h2>")
    else:
        user=EmailBackEnd.authenticate(request,username=request.POST.get("email"),password=request.POST.get("password"))
        if user!=None:
            login(request,user)
            if user.user_type=="1":
                return HttpResponseRedirect('/admin_home')
            elif user.user_type=="2":
                return HttpResponseRedirect('/staff_home')
            elif user.user_type=="3":
                return HttpResponseRedirect('/student_home')

        else  :
            messages.error(request,"Invalid Login Details")
            return HttpResponseRedirect("/")
```

2.2)Add staff:(HodViewa.py)

```
def add_staff(request):
    return render(request,"hod_template/add_staff_template.html")
def add_staff_save(request):
    if request.method!="POST":
        return HttpResponse("Method Not Allowed")
    else:
        first_name=request.POST.get("first_name")
        last_name=request.POST.get("last_name")
        username=request.POST.get("username")
        email=request.POST.get("email")
        password=request.POST.get("password")
        address=request.POST.get("address")
        try:
            user=CustomUser.objects.create_user(username=username,password=password,email=email,last_name=last_name,
            first_name=first_name,user_type=2)
            user.staffs.address=address
            user.save()
            messages.success(request,"Successfully Added Staff")
            return HttpResponseRedirect("/add_staff")
        except:
            messages.error(request,"Failed to Add Staff")
            return HttpResponseRedirect("/add_staff")
```

2.3)Manage and edit staff:(HodViewa.py)(code for edit/update)

```
def manage_subject(request):
    subjects=Subjects.objects.all()
    return render(request,"hod_template/manage_subject_template.html",{"subjects":subjects})
def manage_staff(request):
    staffs=Staffs.objects.all()
    return render(request,"hod_template/manage_staff_template.html",{"staffs":staffs})
```

```
def edit_staff_save(request):
    if request.method!="POST":
        return HttpResponse("<h2>Method Not Allowed</h2>")
    else:
        staff_id=request.POST.get("staff_id")
        first_name=request.POST.get("first_name")
        last_name=request.POST.get("last_name")
        email=request.POST.get("email")
        username=request.POST.get("username")
        address=request.POST.get("address")

        try:
            user=CustomUser.objects.get(id=staff_id)
            user.first_name=first_name
            user.last_name=last_name
            user.email=email
            user.username=username
            user.save()

            staff_model=Staffs.objects.get(admin=staff_id)
            staff_model.address=address
            staff_model.save()
            messages.success(request,"Successfully Edited Staff")
            return HttpResponseRedirect(reverse("edit_staff",kwargs={"staff_id":staff_id}))
        except:
            messages.error(request,"Failed to Edit Staff")
            return HttpResponseRedirect(reverse("edit_staff",kwargs={"staff_id":staff_id}))
```

Like vise for Student,Subject,Course…

2.4)LeaveReportReply:(HodViews.py)

```
def student_leave_view(request):
    leaves=LeaveReportStudent.objects.all()
    return render(request,"hod_template/student_leave_view.html",{"leaves":leaves})

def student_approve_leave(request,leave_id):
    leave=LeaveReportStudent.objects.get(id=leave_id)
    leave.leave_status=1
    leave.save()
    return HttpResponseRedirect(reverse("student_leave_view"))
```

2.5)FeedBackReply:(HodViews.py)

```
def student_feedback_message(request):
    feedbacks=FeedBackStudent.objects.all()
    return render(request,"hod_template/student_feedback_template.html",{"feedbacks":feedbacks})

@csrf_exempt
def student_feedback_message_replied(request):
    feedback_id=request.POST.get("id")
    feedback_message=request.POST.get("message")

    try:
        feedback=FeedBackStaffs.objects.get(id=feedback_id)
        feedback.feedback_reply=feedback_message
        feedback.save()
        return HttpResponse("True")
    except:
        return HttpResponse("False")
```

2.6)TakeAttendnace(StaffViews.py)(Add Result...)

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

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


    try:
        attendance=Attendance(subject_id=subject_model,attendance_date=attendance_date)
        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")
```

2.7)ForFetching(Date,Students...)

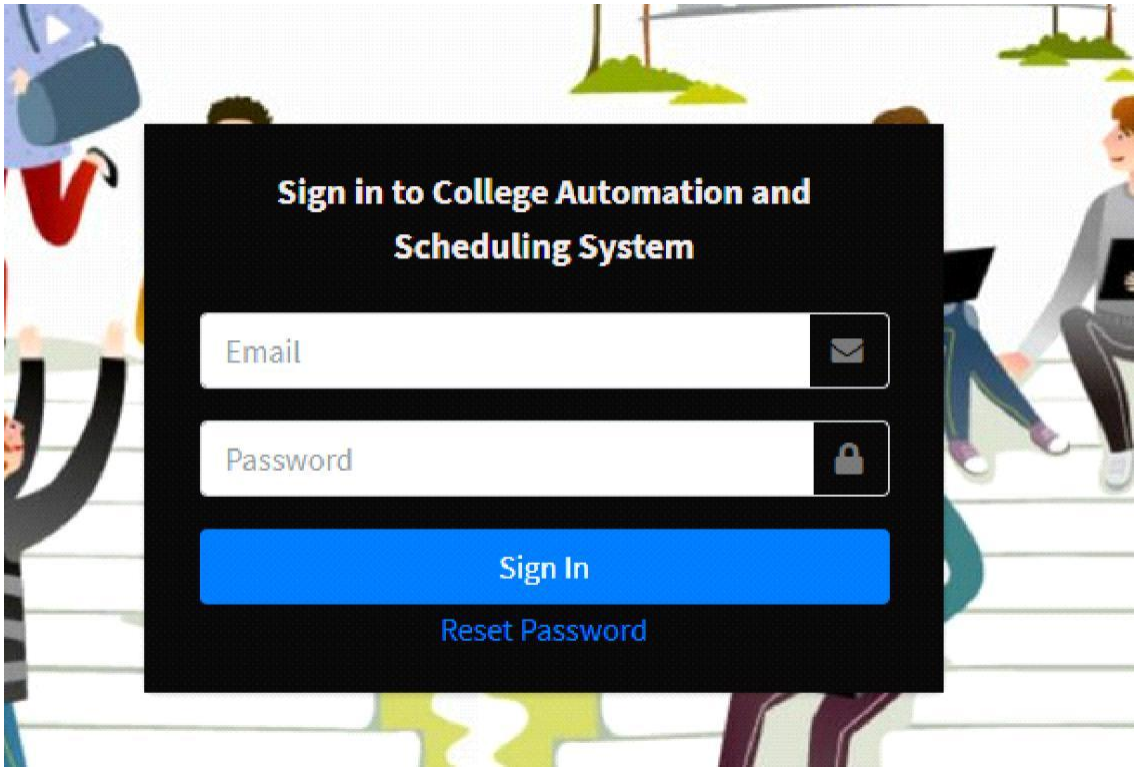
```
@csrf_exempt
def get_students(request):
    subject_id=request.POST.get("subject")

    subject=Subjects.objects.get(id=subject_id)
    students=Students.objects.filter(course_id=subject.course_id)
    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)
```

2.8)ApplyLeave(Student,staff..)(Almost same for sent feedback)

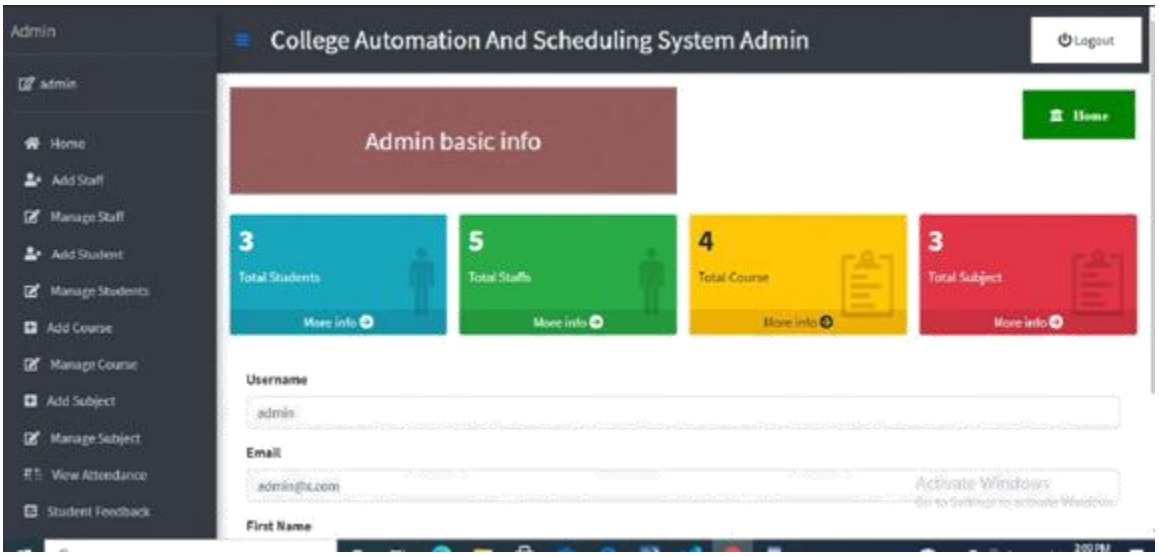
```
def staff_apply_leave(request):
    staff_obj = Staffs.objects.get(admin=request.user.id)
    leave_data=LeaveReportStaff.objects.filter(staff_id=staff_obj)
    return render(request,"staff_template/staff_apply_leave.html",{"leave_data":leave_data})

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"))
```

6.)Screen short:

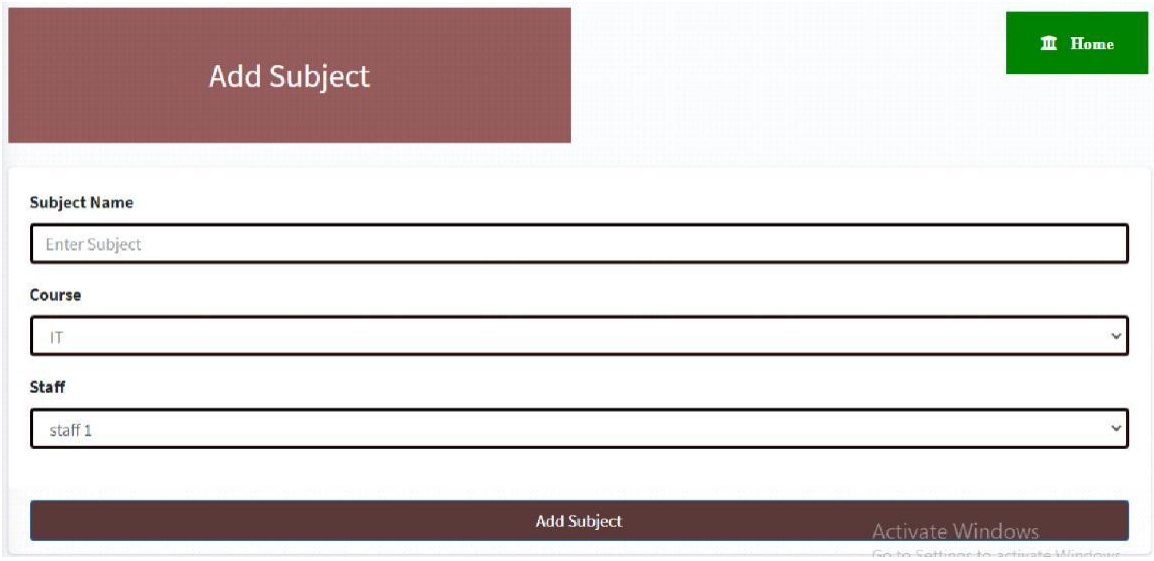
6.1)Login page:



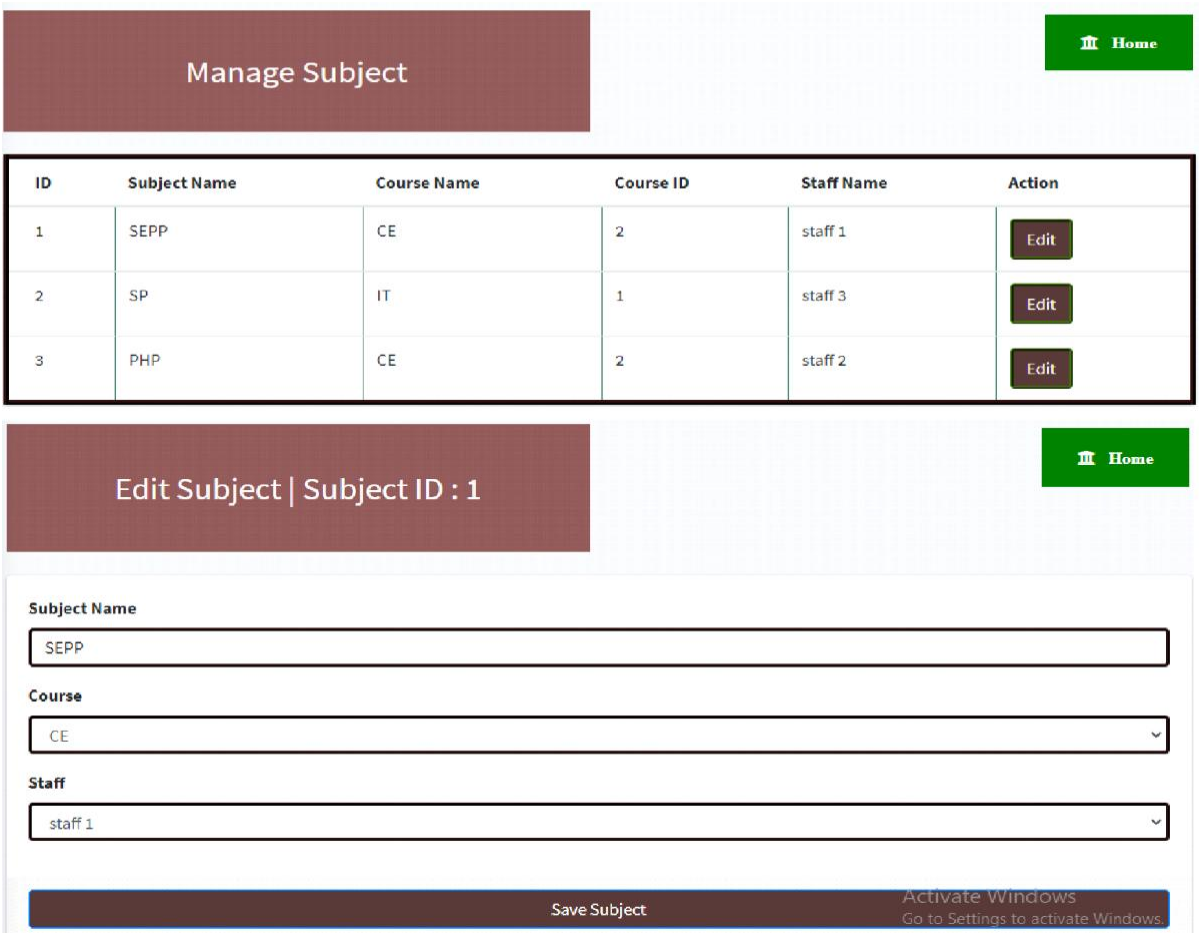
6.2)AdminHomePage:(same for student,staff but content was different)



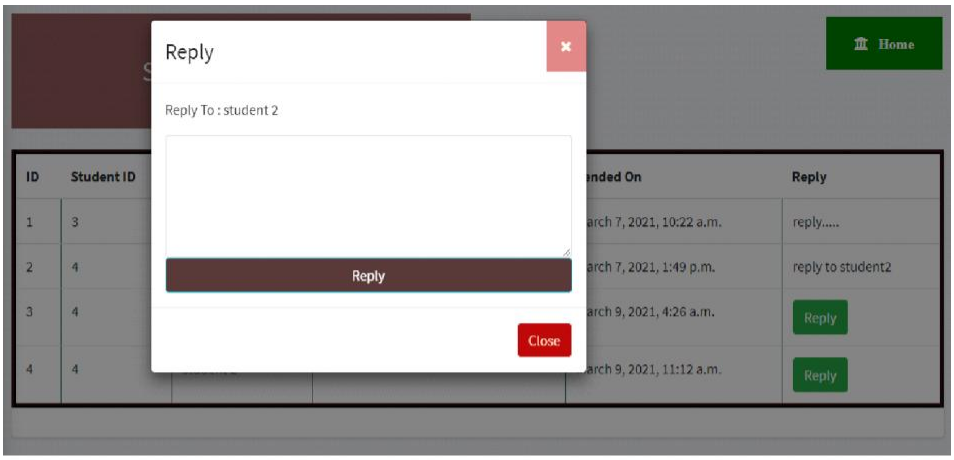
6.3)Add subject(student,staff,course)



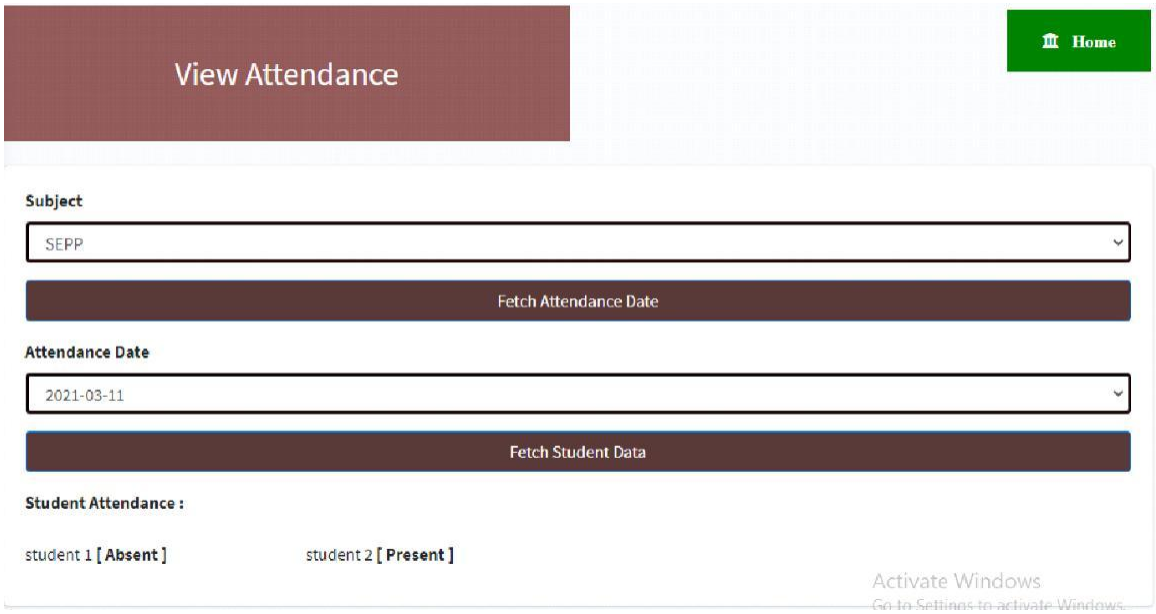
6.4)Manage/Edit subject(student,staff,course)(for Edit profile also look same):



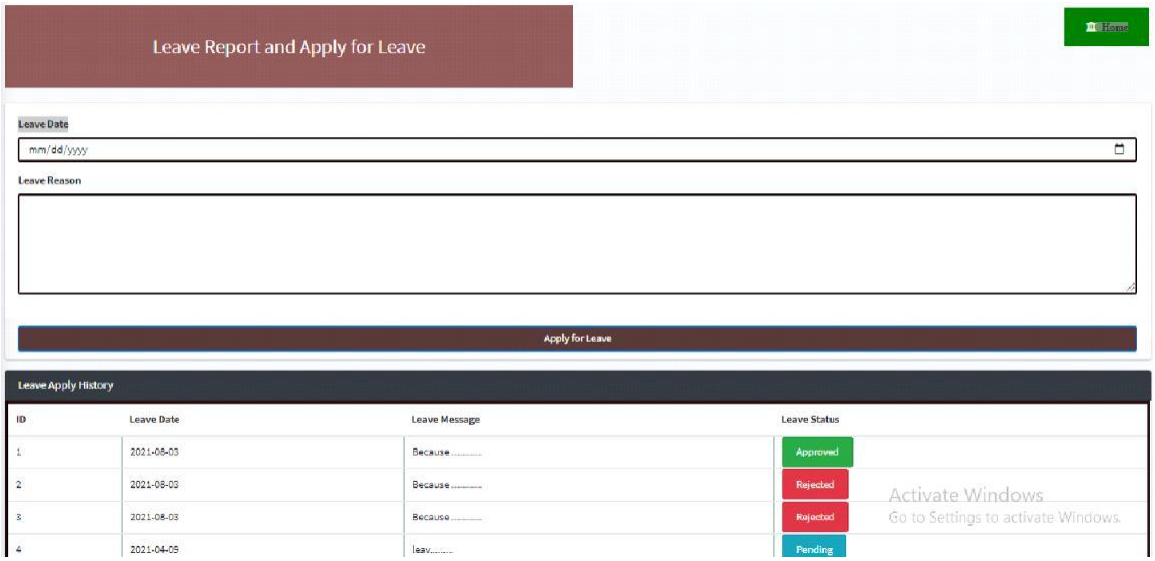
6.5)Reply FeedBack/Leave:



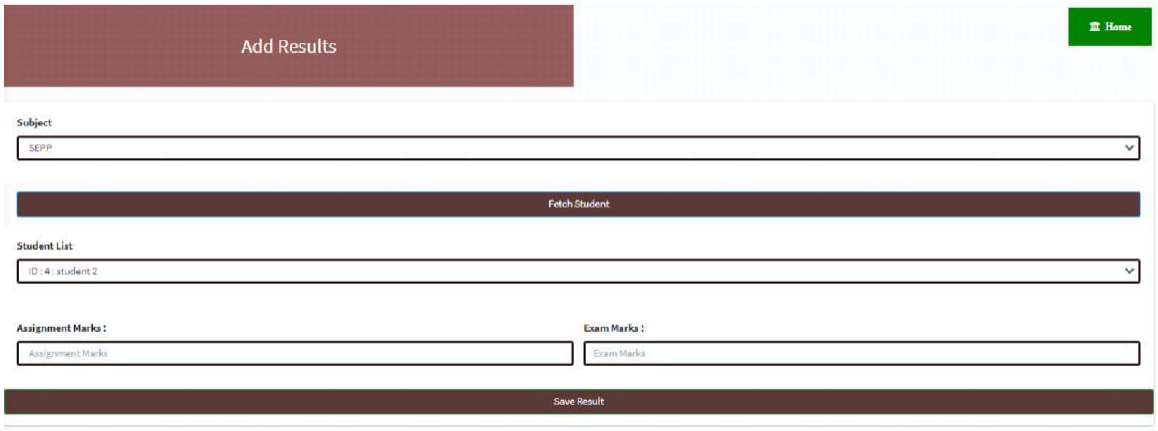
6.6)ViewAttendance(in admin)(in staff formate will use for updating attendance):



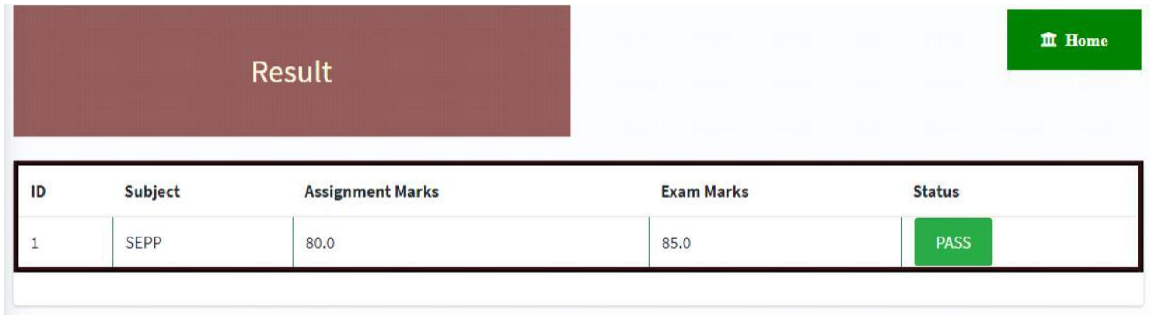
6.6)ApplyLeave(also Same for feedback):

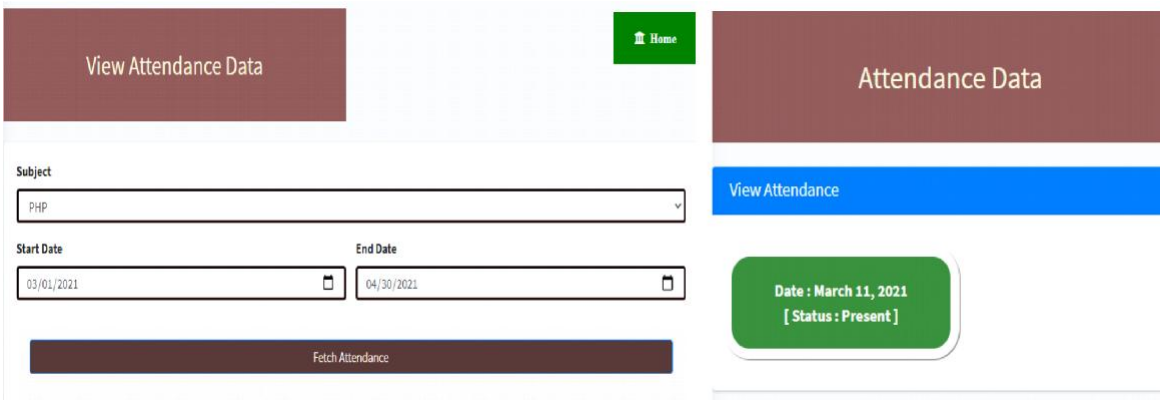


6.7)AddResult:(ForTake attendance...)

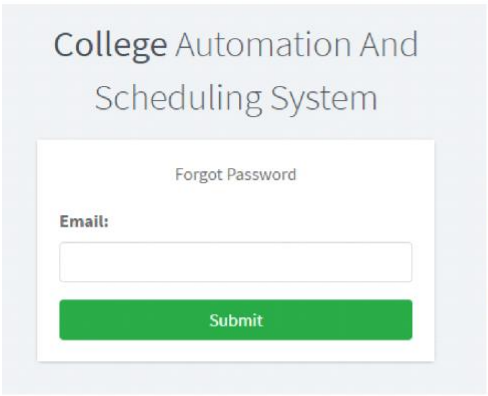


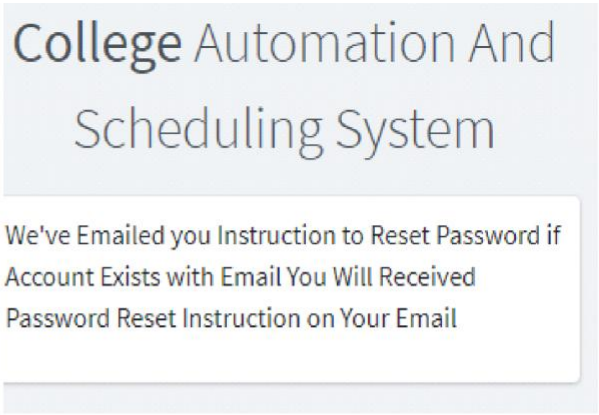
6.8)Grade Card/view attendance:

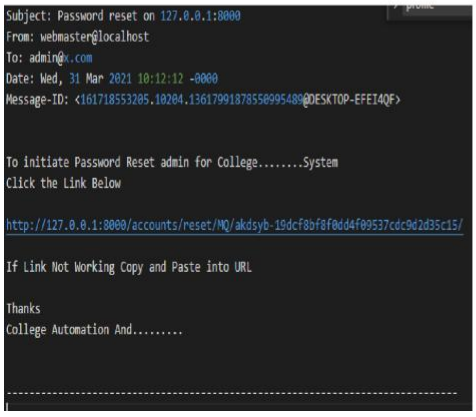


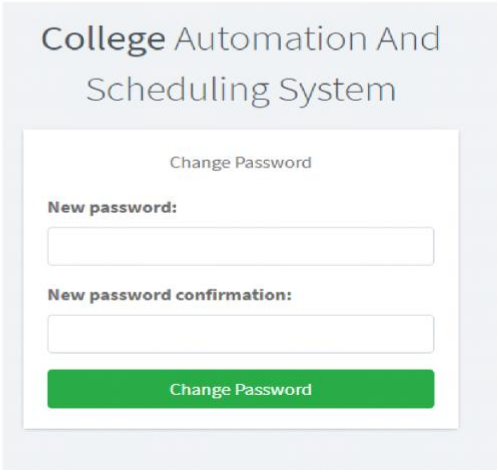


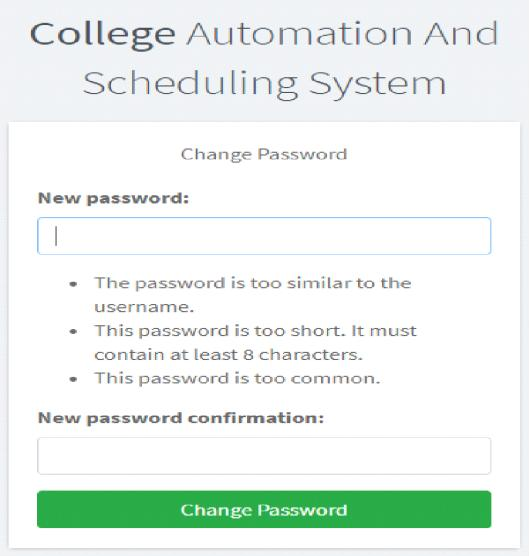
6.9)ForgotPassword:

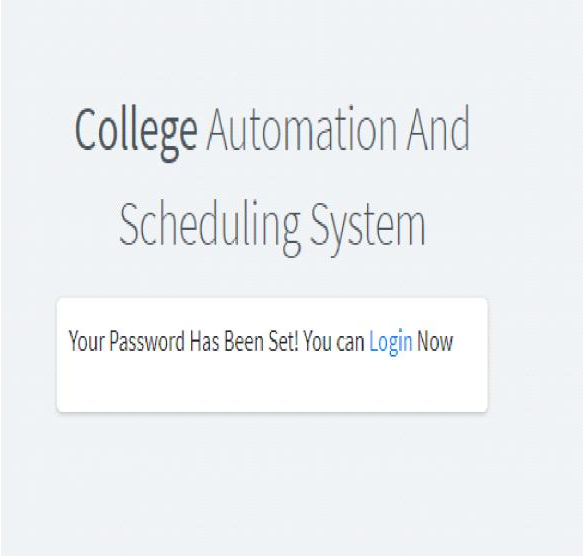












7.)Conclusion:

This paper assists in modifying the existing system to site based system.This is a paperless work. It can be monitored and controlled remotely. It reduces the manpower required. It provides accurate information always. Malpractice can be reduced. All gathered and extra information can be saved and can be accessed at any time. The data which is stored in the project helps in taking intelligent and quick decisions by the management. So it is better to have a Web-Based Information Management system. All the stakeholders,. staff members can get the desired information without delay. This system is essential in the colleges/hostels and universities.

8.)Limitation:-

- **-This system can not work offline.it's must requires internet.**
- -In our system we can not enter data of student according to semester/year/session.
- so we can not identified the student based on year which he/she study in.
- -in attendance we have to add subject and date every time,It's not predefined class and dates than we can just click enter class and student/today's date can be fetched.
- -user_id is pre-increment so if we delete record from database there will be a gape in id and
- also this id not in sequence for particular type of user
- like first we create admin...id==1
- than add two staff for staff table id will start 2 missing 1 like ....
- -For student when he/she view attendance they can fetch according date which they can
- etered.not show the table/excel export....

9.)Future Extension:

-We improve our project and overcome from this limitation and also add more modules like EXAM SECTION,PLACEMENT SECTION , E-LIBRARY etc.Add more functionality in pre-defined module.

10.)Bibliography:

Websites :

https://stackoverflow.com/

https://www.w3schools.com/

https://docs.djangoproject.com/en/3.1/

https://github.com/