A

Project Synopsis Report

On

College Management System

Submitted
For

partial fulfillment of

the award of the Diploma
in

Computer Science & Engineering

Submitted By

Dharmendra Vishvkarma

Enrollment No.- (E20481335500015)

Supervised By

Rakesh Kumar Maurya

(Faculty of C.S.E. Branch)

Department of Computer Science & Engineering

MMIT SIDDHARTHANAGAR , BANSI

Session 2022-23

CONTENTS

1.CERTIFICATE

This is to certify that the project report entitled ***College Management system*** *submitted by* ***Dharmendra Vishvkarma*** *the govt. College of Engineering, MMIT Siddharthanagar, Bansi, in partial fulfillment for the award of the Diploma* ***in Computer Science & Engineering*** *is a bonafide record of project work carried out by him/her under my/our supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.*

Counter signature of HOD

_____

(2022-2023)

2.DECLARATION

I/We declare that this project report titled *College Management System* in partial fulfil ment of the diploma in Computer Science & Engineering is a record of original work carried out by me/us under the supervision of *Mr. Rakesh Kumar Maurya* , and has not formed the basis for the award of any other degree ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Dharmendra Vishvkarma
Enrollment No.- E20481335500015
Roll No.- 2352323555014

3. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my trainer and mentor *Er. Rakesh Kumar Maurya* Senior Consultant, (MMIT Siddharthanagar , Bansi), who gave me his full support and encouraged me to work in an innovative and challenging project for Educational field.

His knowledge and guidance were extremely helpful in completion of my training. I would also extend my thanks to every member of MMIT Siddharthanagar , Bansi for their support and co-operation.

Dharmendra Vishvkarma
Enrollment No.- E20481335500015
Roll No.- 2352323555014

4. Objective & Scope

1.1 Objective

This is a web oriented application allows us to access the whole information about the college, staffs, students, facilities etc. This application provides a virtual tour of Campus. Here we will get the latest information about the students and staffs. This generic application designed for assisting the students of an institute regarding information on the courses, subjects, classes, assignments, grades and timetable. It also provides support that a faculty can also check about his daily schedule, can upload assignments, and notices to the students. Here administrator will manage the accounts of the student and faculties, makes the timetable, and upload the latest information about the campus.

1.2 Scope

- College information: Through this service one can access the complete information about the college campus such as courses available, admission procedure, placements, college events, achievements etc.
- Student tracking: Any company or any organization that want to check the summary about the student of the college, so that they will be able to choose the particular students for their campus placement And for that purpose they will be given a particular link through which they can access the information required.
- Student attendance status: It gives the attendance status of students. Faculty will update the attendance periodically and can be seen by students and parents.
- Student's performance in exams: This facility provides the performance of the student in each exam which is conducted by university or college such as midterm performance. Marks obtained by students in exams will be updated by faculties that can be access by students and parents.
- Exam Notification: This facility notifies students and parents about examination schedule.

- Events: It will give information about different events that will be conducted by college time to time. Information about these events will be updated by administrator.

- Online assignments: This service provides the facility to faculty to upload assignments and to students to submit these assignments online.

- Information about staff: It will help in maintaining complete information about college faculty members such as their department, cadre, date of joining, salary, etc. Administrator will register new faculties and remove their account when they leave the college.

5. Theoretical Background

Today in colleges student details are entered manually. The student details in separate records are tedious task. Referring to all these records and updating is needed. There is a chance for more manual errors.

Problems in existing system:

- It was limited to a single system.
- It was less user-friendly.
- It have a lots of manual work (Manual system does not mean that we are working with pen and paper, it also include working on spread sheets and other simple software's)
- It requires more no of employees need to work.
- It was time consuming process.
- The present system was very less secure.
- It is unable to generate different kinds of report.

Solution to these problems:

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

- User friendliness is provided in the application with various controls.
- The system makes the overall project management much easier and flexible.
- It can be accessed over the Internet.
- Various classes have been used to provide file upload and mail features.
- There is no risk of data mismanagement at any level while the project development is under process.
- It provides high level of security using different protocols like https etc.

6. Problem Definition

The problem is to provide the complete information about the college campus. In which the college staff members, students and parents can access the information and will be familiar with college campus. It will provide interactive environment for the staff, students and parents by getting knowledge of student attendance, remarks, exams performances, grades, timetables, notices etc.

7. System analysis & planning v/s user requirement

7.1 User requirements:

The following requirements are raised during the analysis of the needs of the users:

- A Person Should be able to login to the system through the first page of the Application.
- The Administrator can create users as per user requirement.
- Admin can upload the data for a particular Student. On successful completion of upload, user (Student/Parent/Faculty) can view reports.
- A general user will have access to see the status of particular Student id number.
- Student (user) can use all the facilities, same as which are provided to him in the college.
- Student can see attendance, notices, grades, report and other facilities in updated manner.
- There will be a separate page for every student as his account in which he can get notices, attendance, grades, assignments etc.
- Parent can just view the record of student with the username and password provided.
- Faculty can give the attendances and notices for the students.
- The administrator verifies all these reports and generates them for users to view them.

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

7.2 Analysis Model

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
    1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning and designing the second prototype.
    4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operatingcost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.

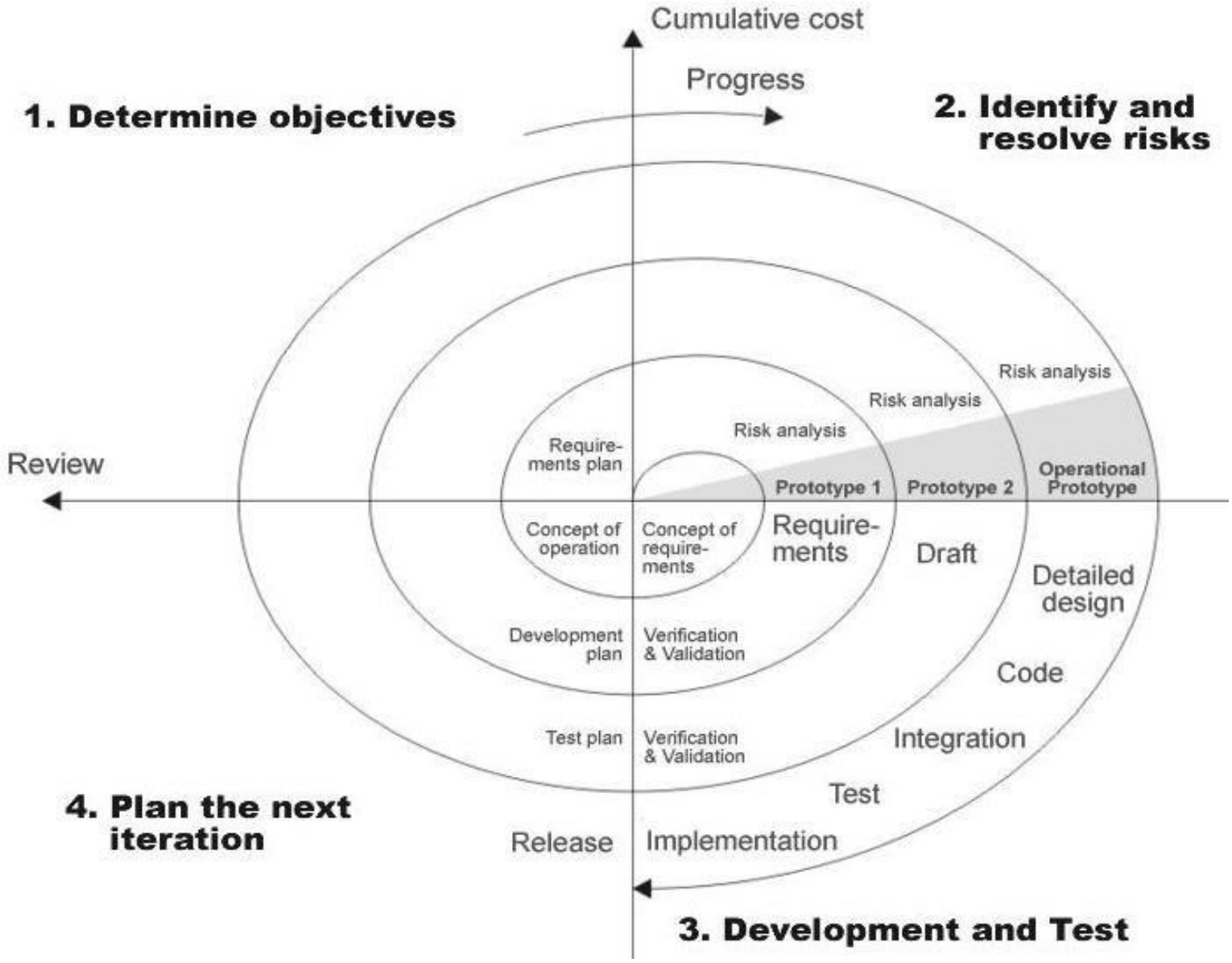
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

- The final system is constructed, based on the refined prototype.

- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:



7.3 Study of the System:

7.3.1 Graphical user interface

In the flexibility of the uses the interface has been developed a graphics concept in mind, associated through a browses interface. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The interfaces help the administrations with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The operational or generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.

7.4 Number of Modules

The system after careful analysis has been identified to be presented with the following modules:

The modules involved are:

1. College information: Through this service one can access the complete information about the college campus such as courses available, admission procedure, placements, college events, achievements etc.
2. Student tracking: Any company or any organization that want to check the summary about the student of the college, so that they will be able to choose the particular students for their campus placement And for that purpose they will be given a particular link through which they can access the information required.

3. Student attendance status: It gives the attendance status of students. Faculty will update the attendance periodically and can be seen by students and parents.
4. Student's performance in exams: This facility provides the performance of the student in each exam which is conducted by university or college such as midterm performance. Marks obtained by students in exams will be updated by faculties that can be access by students and parents.
5. Exam Notification: This facility notifies students and parents about examination schedule.
6. Events: it will give information about different events that will be conducted by college time to time. Information about these events will be updated by administrator.
7. Online assignments: This service provides the facility to faculty to upload assignments and to students to submit these assignments online.
8. Information about staff: It will help in maintaining complete information about college faculty members such as their department, cadre, date of joining, salary, etc. Administrator will register new faculties and remove their account when they leave the college.

8. SYSTEM PLANNING (PERT CHART)

Perform and evaluate feasibility studies like cost-benefit analysis, technical feasibility, time feasibility and operational feasibility for the project. Project Scheduling should be made using PERT charts.

Feasibility study is carried out to decide whether the proposed system is feasible for the company. The feasibility study is to serve as a decision document it must answer three key questions:

1. Is there a new and better way to do the job that will benefit the user?
2. What are the cost and the savings of the alternative(s)?
3. What is recommended?

Technical feasibility:

Technical feasibility centers on the existing computer system i.e. Hardware, Software etc. Bank requires SQL database management that are all easily available with extensive development support through manuals and blogs.

Economical feasibility:

Economical Feasibility is the most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as Cost/ Benefit analysis, the procedure is to determine the benefits and savings that are expected from the candidate system and compare them with costs. If the benefits outweigh costs, then the decision is made to design and implement the system.

9. Methodology Adopted, System Implementation & Details of Hardware & Software Used

9.1 Methodology adopted and System implementation:

1. Apache tomcat is used as a web server to host the application.
2. All the environment variables are set.
3. The application is pasted in the webapps folder.
4. Web server is started now.
5. Application is run using the web browser by typing http://localhost/cis
6. Web.xml file is used to control the flow and user actions.

9.2 Details of hardware & software used:

Hardware Specification (Minimum):

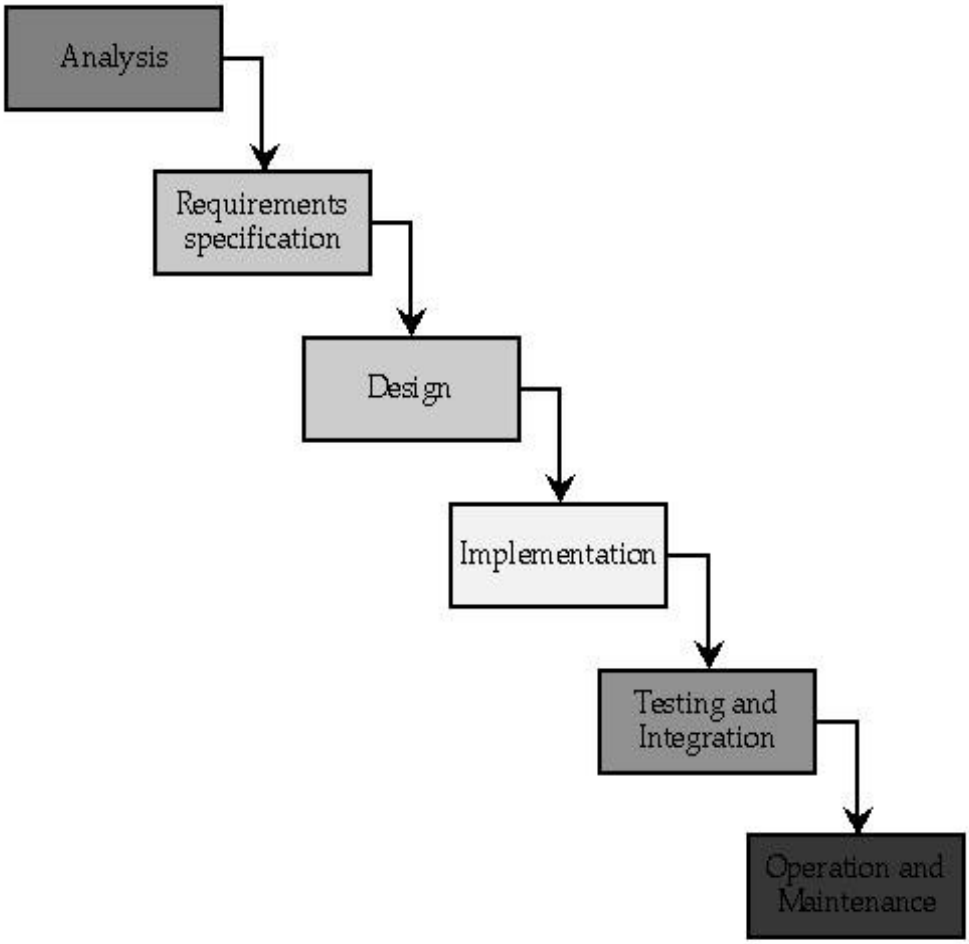
Disc Space:	40 GB
PC Used:	IBM Compatible
Processor:	Intel
Memory:	512 MB RAM
File System:	32 Bit

Software Specification:

Operating System (Server Side):	Windows XP.
Operating System (Client Side):	Windows XP.
Client End Language:	HTML
Server Side Language:	Python With Django
Database:	My Sql
Web Server:	XAMPP server
Web Browser:	Chrome / Internet Explorer / Mozilla Firefox / Etc.

10.DETAILED LIFE CYCLE OF PROJECT

We have used Waterfall Model as Software Engineering life Cycle Process. It is the simplest; oldest and most widely used process model for software development .This model acquires its name from the fact that classic software life cycle is represented as a sequence of descending steps.



10.1 Requirement Analysis:

This process is also known as feasibility study. In this phase, the development team studied the site requirement. They investigate the need for possible dynamic representation of the site and increase security features. By the end of feasibility study, the team furnishes a document that holds the different specific recommendations for the candidate system. It also includes personnel assignments, costs, project schedules, target dates etc. the requirement gathering process is intensified and focused specially

on software. The essential purpose of this phase is to find the need and to define the problem that needs to be solved. During this phase following facts were gathered.

- Determined the user need
- Identified the facts
- Establish the goals and objective for the proposed system
- Feasibility for the new system

10.2 System Analysis and Design:

In this phase the software's overall structure and its nuances are defined. In terms of client server technology the no of tiers needed for the package architecture, database design, data structure design etc are defined in this phase. Analysis and Design are very crucial in entire development cycle. Any glitch in this phase could be expensive to solve in the later stage of software development. Hence following is the essential approach taken during website designing:

- DFD
- Database Designing
- Form Designing
- Pseudo code for methods

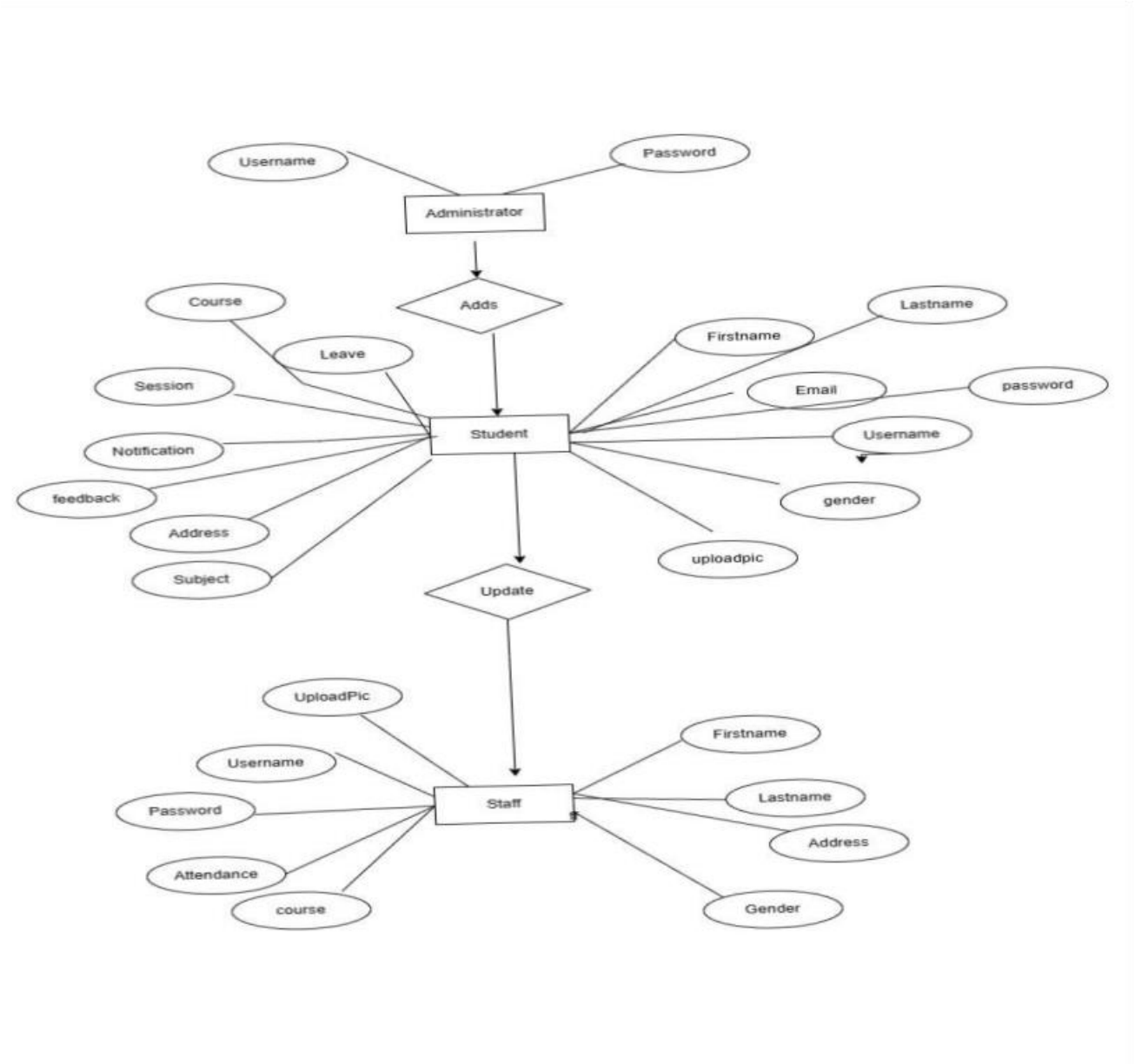
10.3 Testing:

Once the code is generated, the website testing begins. Different testing methodologies are done to unravel the bugs that were committed during the previous phases. Different testing methodologies are used:

- Acceptance testing ❖ White Box Testing
- Black Box Testing

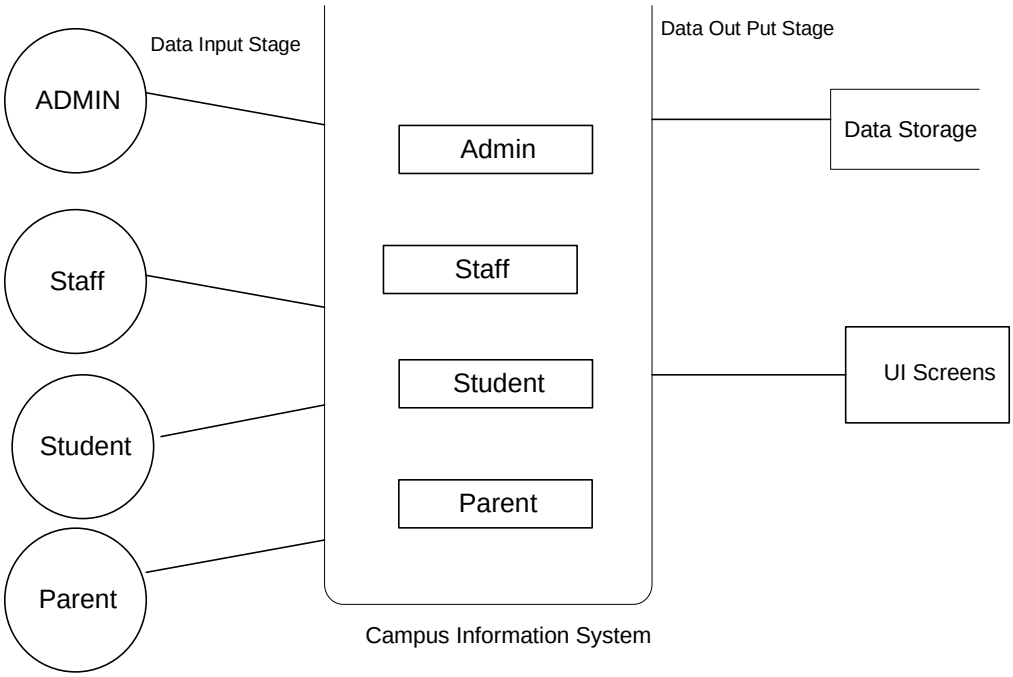
11. ER-Diagram and Data Flow Diagram

ER - DIAGRAM

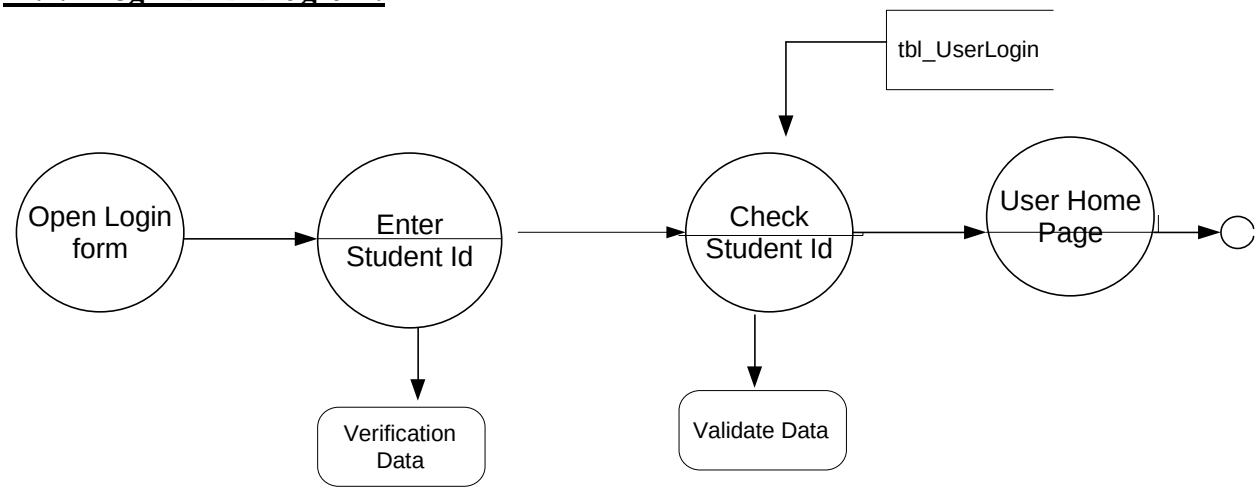


Data Flow Diagram:-

11.2.1 Context 0th Level Diagram:

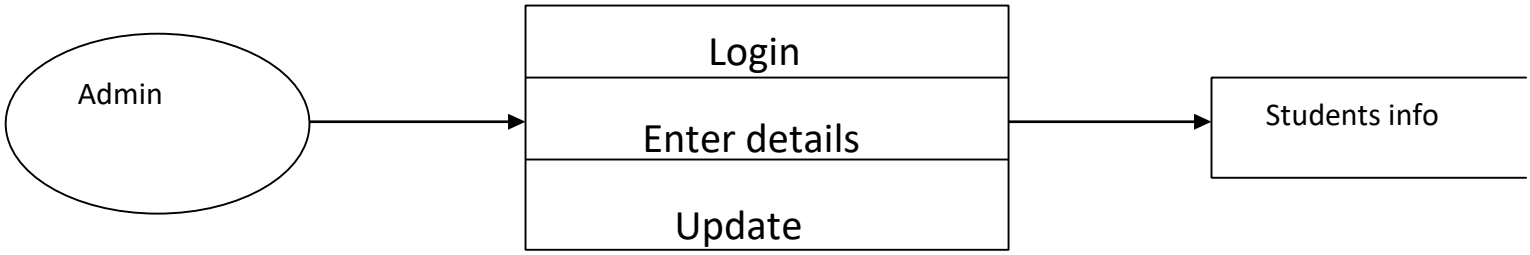


11.2.2 Login DFD Diagram:

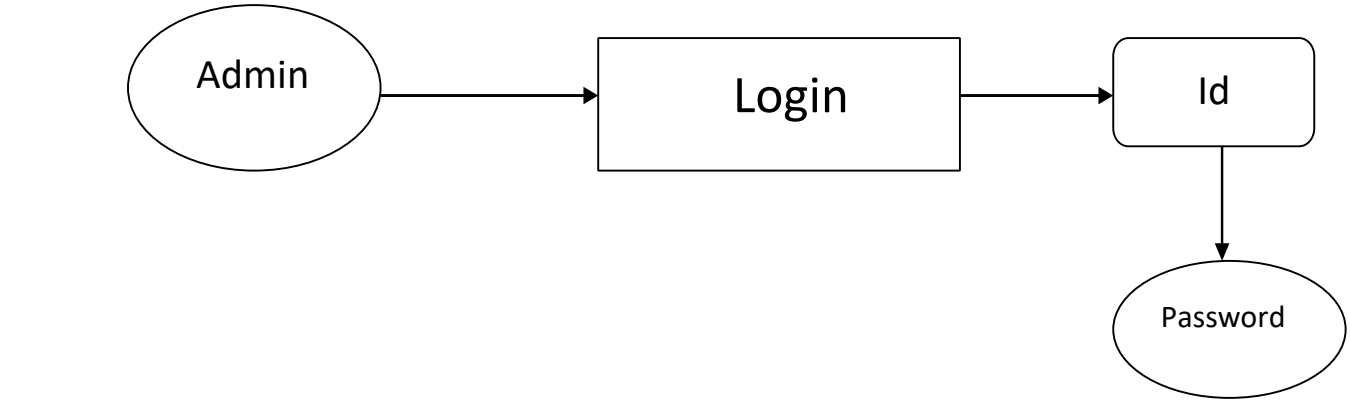


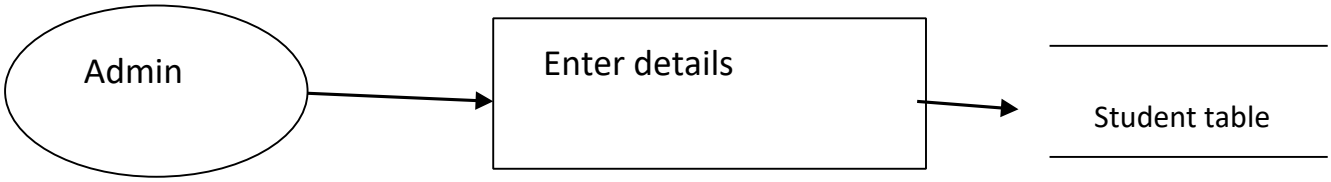
11.2.3 Admin Details Data Flow:

1st level DFD:



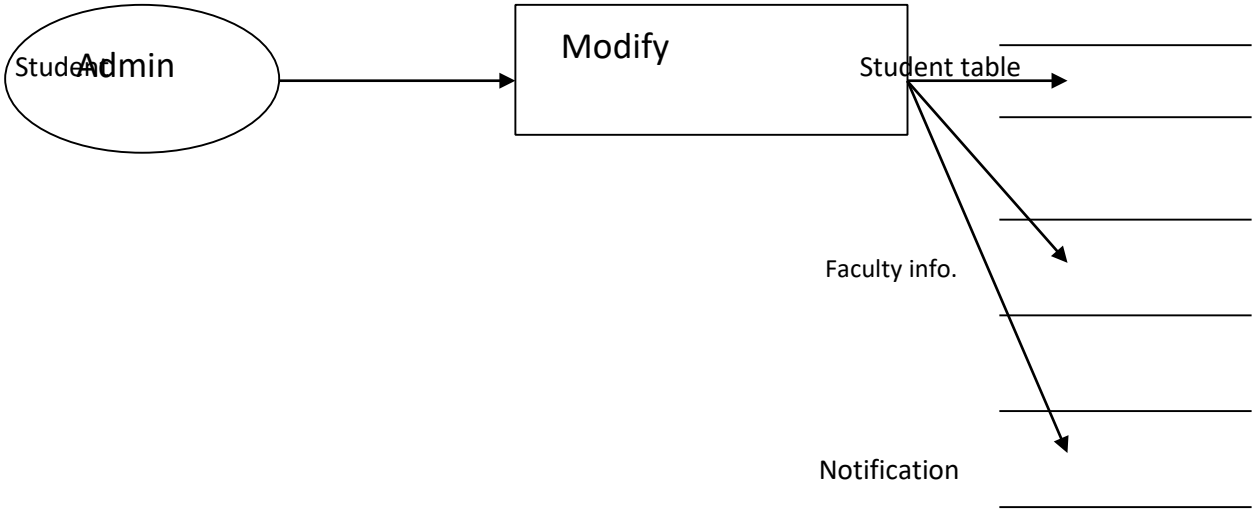
2nd Level DFD



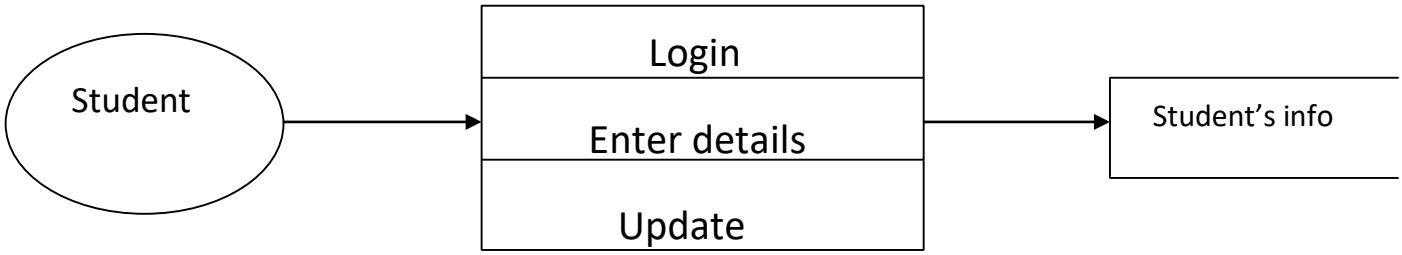


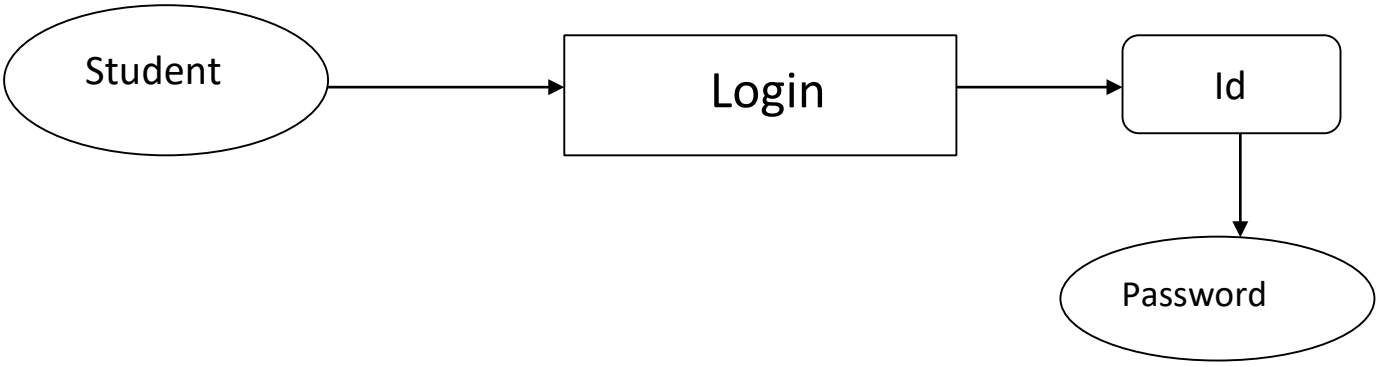
11.2.4 Student Details Data Flow

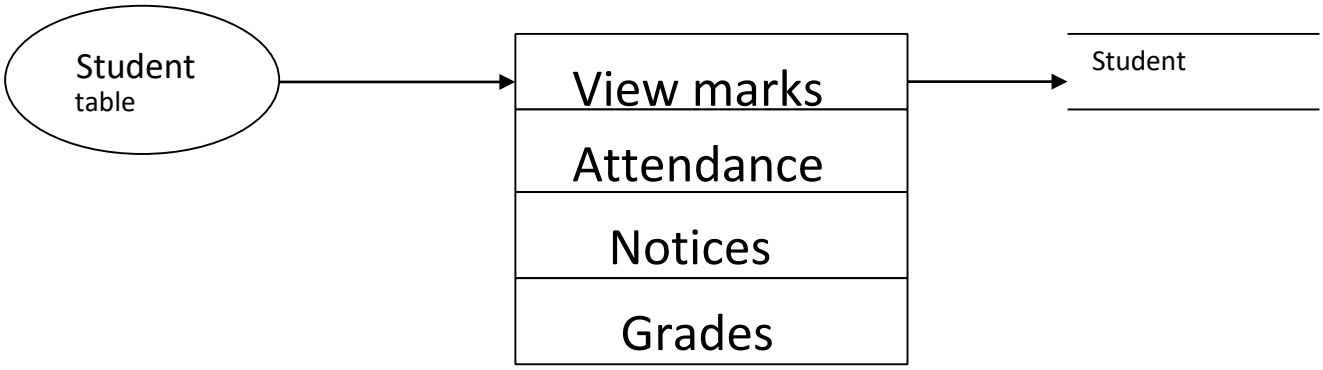
1st level DFD

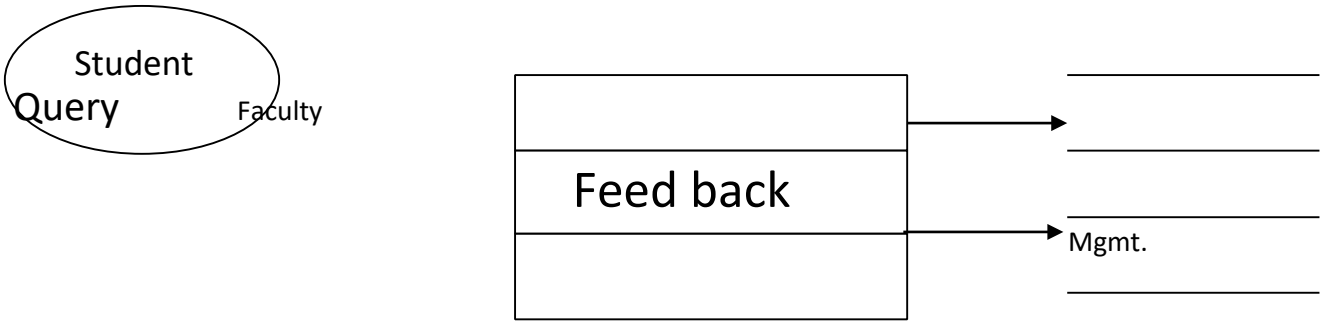


2nd Level DFD






Student
table
View marks
Attendance
Notices
Grades
Student


Student
Query
Faculty
Feed back
Mgmt.

12.Database Design

Table Name :-StudentUser

Colume Name	Data type
User	Charfield(50)
User_type	Charfield(50)
Profile_Pic	Image_field()

Table Name :-HOD

Colume Name	Data Type
Profile pick	Image field()
Firmname	charfield (100)
lastname	charfield (100)
Gender	charfield (255)
Email	charfield (15) primary key
Username	charfield (50) primary key
password	charfield (50)

Table Name :-Course

Colume Name	Data Type
name	charfield (100)
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Session Year

Colume Name	Data Type
Session_year_start	charfield (100)
Session_year_end	charfield (100)

Table Name:- Login

Colume Name	Data type
userid	charfield (50) primary key
Password	charfield (20)
usertype	charfield (50)

Table Name :- Student

Colume Name	Data Type
Profile pick	Image field()
Firmname	charfield (100)
lastname	charfield (100)
Gender	charfield (255)
Email	charfield (15) primary key
Username	charfield (50)
password	charfield (50)
address	Textfield(100)
Course_id	Foreingnkey()
Session year_id	Foreingnkey()

Table Name:- Staff

Colume Name	Data Type
Admin	OneToOneField()
Profile pick	Image field()
address	Textfield(100)
Gender	charfield (255)
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Subject

Colume Name	Data type
Name	Charfield(100)
Course	Foreingnkey()
Staff	Foreingnkey()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Staff Notifications

Colume Name	Data type
Staff_id	Foreingnkey()
message	Textfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Student Notifications

Colume Name	Data type
Student_id	Foreingnkey()
message	Textfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:-

Colume Name	Data type
Staff_id	Foreingnkey()
date	Charfield(100)
message	Textfield()
Status	Integerfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Student Leave

Colume Name	Data type
Student_id	Foreingnkey()
date	Charfield(100)
message	Textfield()
Status	Integerfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Staff Feedback

Colume Name	Data type
Staff_id	Foreingnkey()
feedback	Textfield()
Feedback_reply	Textfield()
Status	Integerfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Attendance

Colume Name	Data type
Subject_id	Foreingnkey()
Attendance_date	DateTimeField()
Session_year_id	Foreingnkey()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Attendance Report

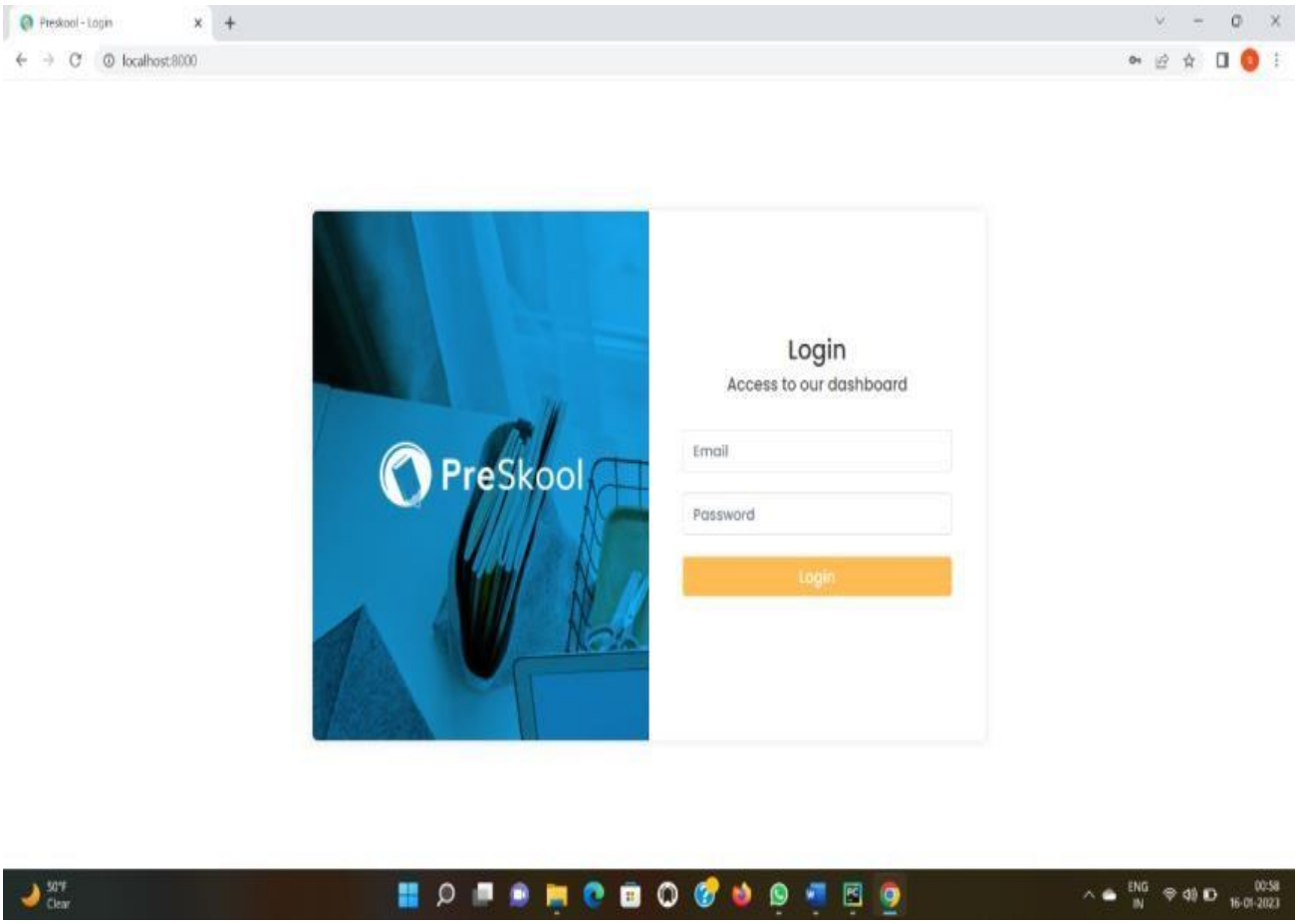
Colume Name	Data type
Subject_id	Foreingnkey()
Attendance_date	DateTimeField()
Created_at	DateTimeField()
Updated_at	DateTimeField()

Table Name:- Student Result

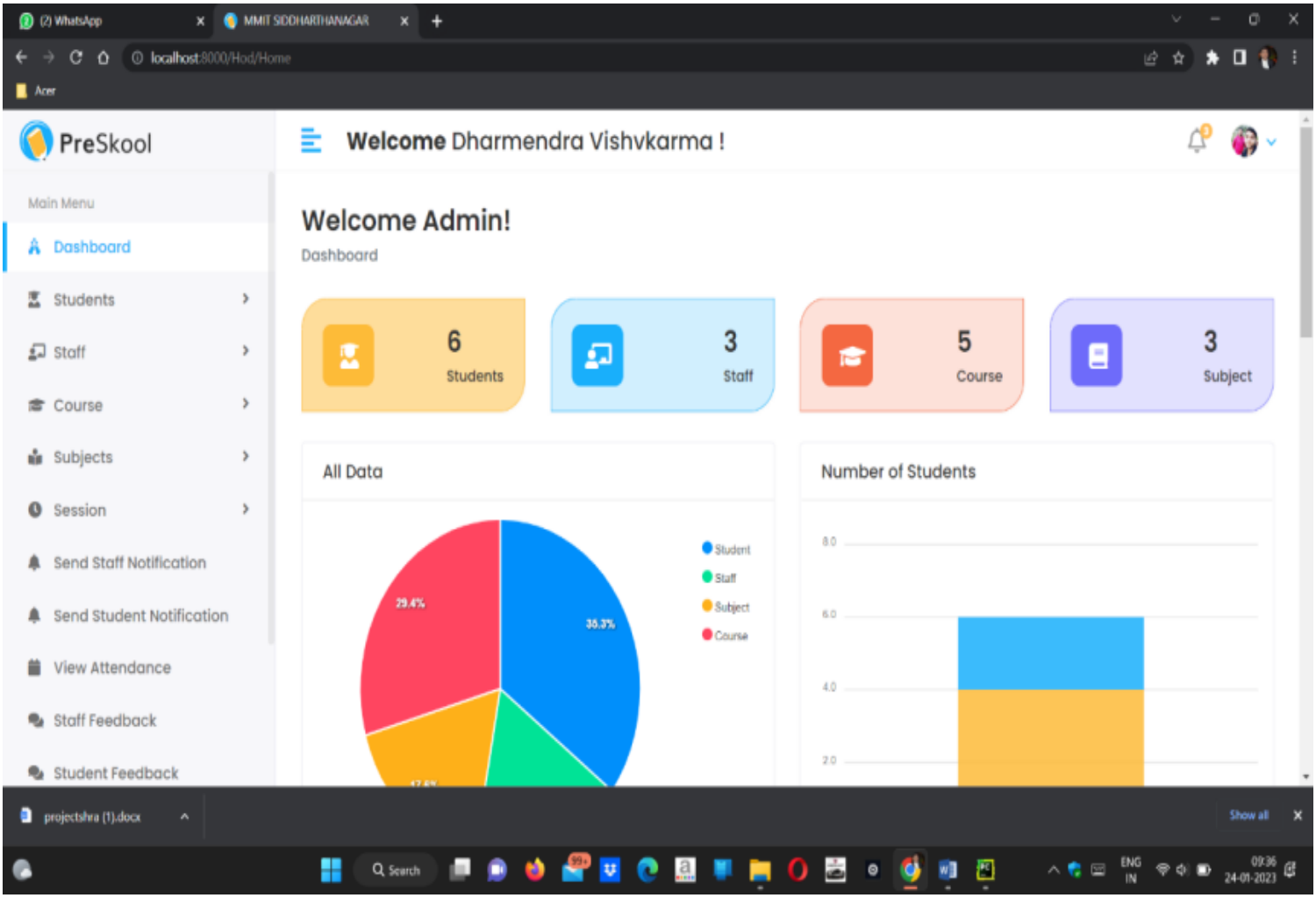
Colume Name	Data type
Subject_id	Foreingnkey()
Staff_id	Foreingnkey()
Assignment_mark	Integerfield()
Exam_Mark	Integerfield()
Created_at	DateTimeField()
Updated_at	DateTimeField()

13.Input and Output Screen Design (Snapshots)

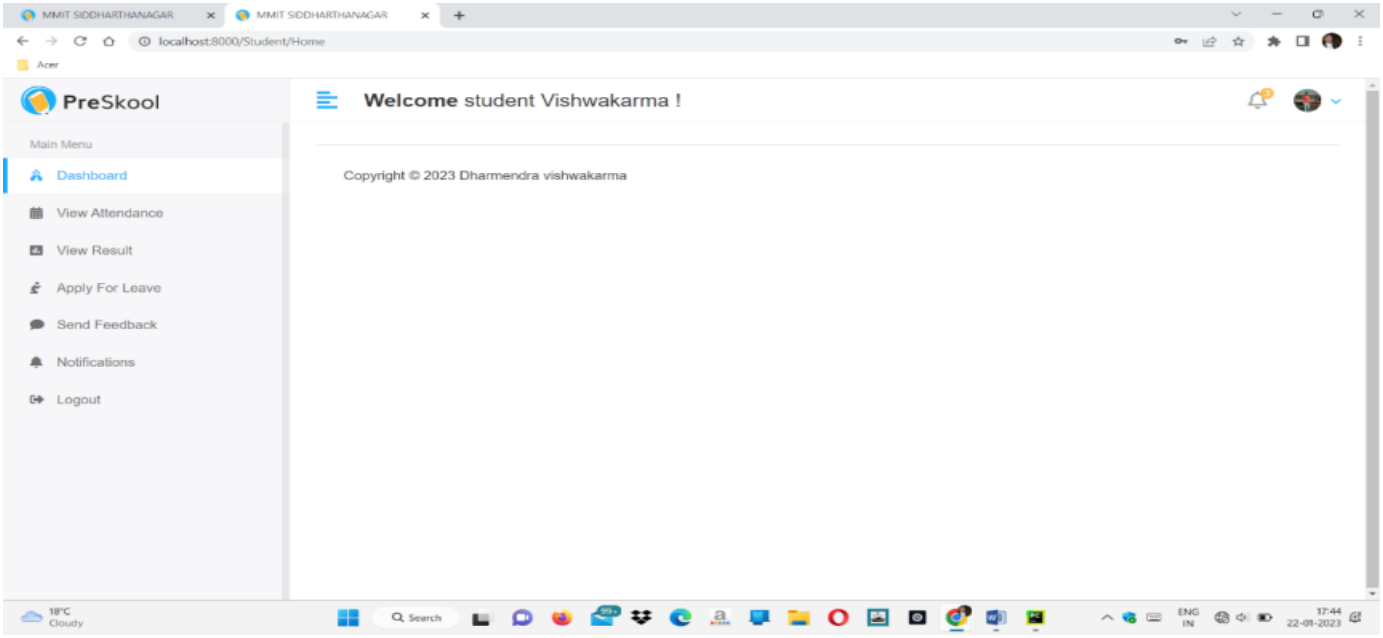
LOGIN PAGE



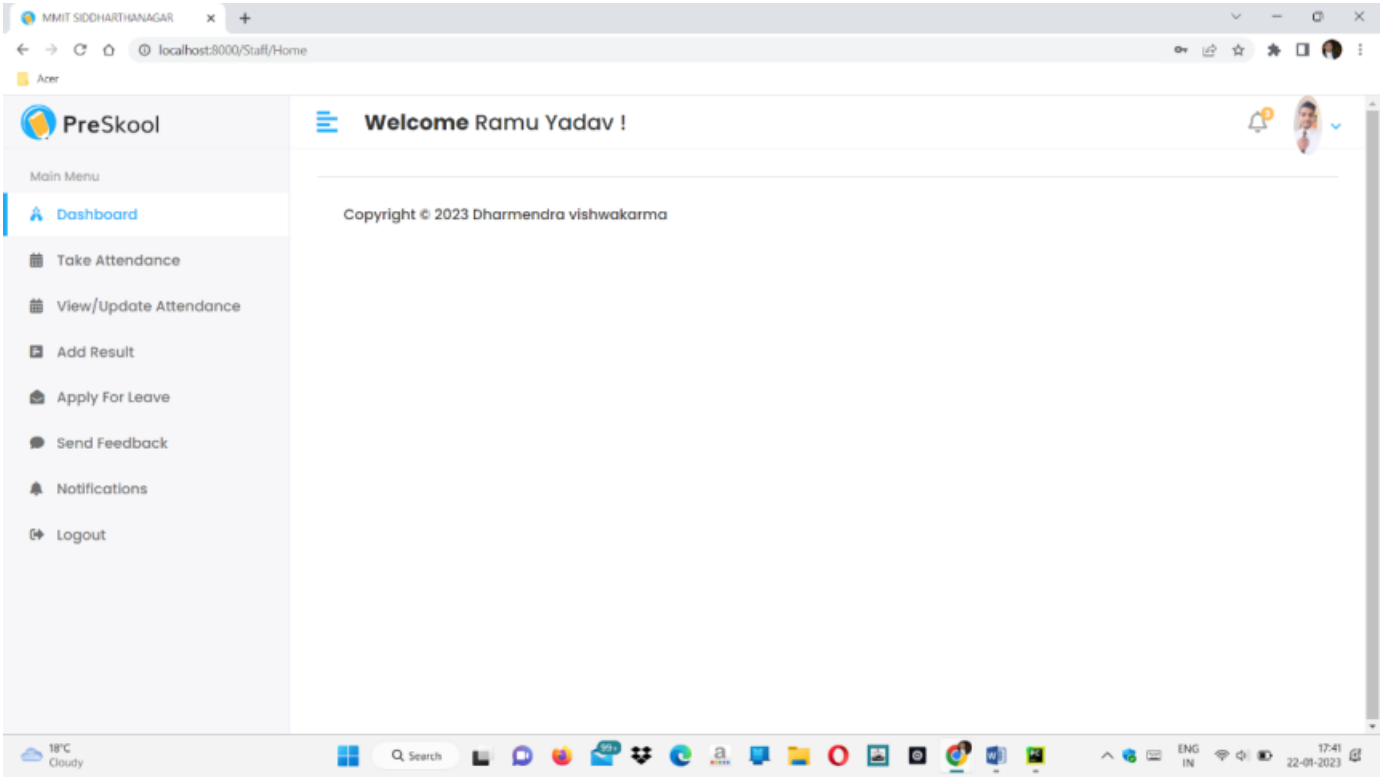
HOME PAGE


PreSkool
Welcome Dharmendra Vishvkarma !
Main Menu
Dashboard
Students
Staff
Course
Subjects
Session
Send Staff Notification
Send Student Notification
View Attendance
Staff Feedback
Student Feedback
Welcome Admin!
Dashboard
6
Students
3
Staff
5
Course
3
Subject
All Data
Number of Students
Student
Staff
Subject
Course
36.3%
29.4%
8.0
6.0
4.0
2.0

STUDENT PAGE



STAFF PAGE



14. Methodology used for testing

The completion of a system will be achieved only after it has been thoroughly tested. Though this gives a feel the project is completed, there cannot be any project without going through this stage. Hence in this stage it is decided whether the project can undergo the real time environment execution without any break downs, therefore a package can be rejected even at this stage.

14.1 Testing methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

1) **Black box testing** - Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.
2) **White box testing** - White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these).White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.
3) **Grey Box Testing** - Grey box testing involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as "grey box," because the input and output are clearly outside of the "black-box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed

for test. Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

4) **Acceptance testing -** Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.
2. Acceptance testing performed by the customer is known as user acceptance testing (UAT).

5) Regression Testing - Regression testing is any type of software testing that seeks to uncover software regressions. Such regression occurs whenever software functionality that was previously working correctly stops working as intended. Typically regressions occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

6) Non Functional Software Testing - Special methods exist to test non-functional aspects of software.

- Performance testing checks to see if the software can handle large quantities of data or users. This is generally referred to as software scalability. This activity of Non Functional Software Testing is often times referred to as Load Testing.
- Stability testing checks to see if the software can continuously function well in or above an acceptable period. This activity of Non Functional Software Testing is often times referred to as indurations test.
- Usability testing is needed to check if the user interface is easy to use and understand.
- Security testing is essential for software which processes confidential data and to prevent system intrusion by hackers.
- Internationalization and localization is needed to test these aspects of software, for which a pseudo localization method can be used.

15. CODING:-

Models.py

```
from django.db import models from
django.contrib.auth.models import AbstractUser


# create your models here


class CustomUser(AbstractUser):
    USER = (
        (1, 'HOD'),
        (2, 'STAFF'),
        (3, 'STUDENT'),
    )

    user_type = models.CharField(choices=USER, max_length=50, default=1)
profile_pic = models.ImageField(upload_to='media/profile_pic')

    # Create Course and session Model


class Course(models.Model):
    name = models.CharField(max_length=100)
created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)

    def _str_(self):
 return self.name
class Session_Year(models.Model):
```

```
    session_start = models.CharField(max_length=100)
session_end = models.CharField(max_length=100)

    def _str_(self):
        return self.session_start + "To" + self.session_end

    # Create Student Model


class Student(models.Model):
admin = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
    address = models.TextField()
  gender = models.CharField(max_length=100)
 course_id = models.ForeignKey(Course, on_delete=models.DO_NOTHING)
    session_year_id=models.ForeignKey(Session_Year,
on_delete=models.DO_NOTHING)
created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)

    def _str_(self):
        return self.admin.first_name + " " + self.admin.last_name


class Staff(models.Model):
admin = models.OneToOneField(CustomUser, on_delete=models.CASCADE)
address = models.TextField()
gender = models.CharField(max_length=100)
created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def _str_(self):
```

```
        return self.admin.username


class Subject(models.Model):
    name = models.CharField(max_length=100)
course = models.ForeignKey(Course, on_delete=models.CASCADE)
 staff = models.ForeignKey(Staff, on_delete=models.CASCADE)
created_at = models.DateTimeField(auto_now_add=True, null=True)
updated_at = models.DateTimeField(auto_now=True)

    def _str_(self):
return self.name


class Staff_Notification(models.Model):
    staff_id = models.ForeignKey(Staff, on_delete=models.CASCADE)
message = models.TextField()
 created_at = models.DateTimeField(auto_now_add=True)
status = models.IntegerField(null=True, default=0)

    def _str_(self):
        return self.staff_id.admin.first_name


class Student_Notification(models.Model):
    student_id = models.ForeignKey(Student, on_delete=models.CASCADE)
 message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
 status = models.IntegerField(null=True, default=0)

    def _str_(self):
```

```
        return self.student_id.admin.first_name
```

admin.py

```
from django.contrib import admin from .models import * from
django.contrib.auth.admin import UserAdmin


class UserModel(UserAdmin):
    list_display = ['username', 'user_type']


admin.site.register(CustomUser,UserModel)
admin.site.register(Course)
admin.site.register(Session_Year)
admin.site.register(Student)
admin.site.register(Staff)
admin.site.register(Subject)
admin.site.register(Staff_Notification)
admin.site.register(Student_Notification)
```

urls.py

```
  from django.contrib import admin from django.urls import path
from django.conf import settings from django.conf.urls.static import static


from . import views, Hod_views, Student_Views, Staff_Views

urlpatterns = [
 path('admin/', admin.site.urls),
```

```
 path('base/', views.BASE, name='base'),

            # login path
            path('', views.LOGIN, name='login'),
    path('doLogin', views.doLogin, name='doLogin'),
 path('doLogout', views.doLogout, name='logout'),

            # profile Update
path('Profile', views.PROFILE, name='profile'),
     path('Profile/update', views.PROFILE_UPDATE, name='profile_update'),

            # This is Hod Panel url
            path('Hod/Home', Hod_views.HOME, name='hod_home'),
   path('Hod/Student/Add', Hod_views.ADD_STUDENT,
name='add_student'),
path('Hod/Student/View', Hod_views.VIEW_STUDENT, name='view_student'),
            path('Hod/Student/Edit/<str:id>', Hod_views.EDIT_STUDENT, name='edit_student'),
            path('Hod/Student/Update', Hod_views.UPDATE_STUDENT, name='update_student'),
            path('Hod/Student/Delete/<str:admin>',
     Hod_views.DELETE_STUDENT, name='delete_student'),


            path('Hod/Staff/Add', Hod_views.ADD_STAFF, name='add_staff'),

   path('Hod/Staff/View', Hod_views.VIEW_STAFF, name='view_staff'),

path('Hod/Staff/Edit/<str:id>',Hod_views.EDIT_STAFF, name='edit_staff'),

  path('Hod/Staff/Update', Hod_views.UPDATE_STAFF, name='update_staff'),

            path('Hod/Staff/Delete/<str:admin>', Hod_views.DELETE_STAFF, name='delete_staff'),
```

```
        path('Hod/Course/Add',Hod_views.ADD_COURSE,name='add_course'),
path('Hod/Course/View', Hod_views.VIEW_COURSE, name='view_course'),
        path('Hod/Course/Edit/<str:id>', Hod_views.EDIT_COURSE, name='edit_course'),
        path('Hod/Course/Update', Hod_views.UPDATE_COURSE, name='update_course'),
        path('Hod/Course/Delete/<str:id>', Hod_views.DELETE_COURSE, name='delete_course'),

        path('Hod/Subject/Add',Hod_views.ADD_SUBJECT,name='add_subject'),
path('Hod/Subject/View', Hod_views.VIEW_SUBJECT, name='view_subject'),
        path('Hod/Subject/Edit/<str:id>', Hod_views.EDIT_SUBJECT, name='edit_subject'),
        path('Hod/Subject/Update', Hod_views.UPDATE_SUBJECT, name='update_subject'),
        path('Hod/Subject/Delete/<str:id>', Hod_views.DELETE_SUBJECT, name='delete_subject'),

        path('Hod/Session/Add',Hod_views.ADD_SESSION,name='add_session'),
path('Hod/Session/View', Hod_views.VIEW_SESSION, name='view_session'),
        path('Hod/Session/Edit/<str:id>', Hod_views.EDIT_SESSION, name='edit_session'),
        path('Hod/Session/Update', Hod_views.UPDATE_SESSION, name='update_session'),
        path('Hod/Session/Delete/<str:id>', Hod_views.DELETE_SESSION, name='delete_session'),

        path('Hod/Staff/Send_Notification',Hod_views.STAFF_SEND_NOTIFICATION,
name='staff_send_notification'),
path('Hod/Staff/save_notification',Hod_views.SAVE_STAFF_NOTIFICATION,
name='save_staff_notification'),


path('Hod/Student/Send_Notification',Hod_views.STUDENT_SEND_NOTIFICATION,
            name='student_send_notification'),

path('Hod/Student/Save_Notification',Hod_views.SAVE_STUDENT_NOTIFICATION,
            name='save_student_notification'),
```

```
path('staff/Add/Result',Staff_Views.STAFF_ADD_RESULT,name='staff_add_result'),

path('staff/Save/Result',Staff_Views.STAFF_SAVE_RESULT,name='staff_save_result'),



        # This is Student urls
        path('Student/Home', Student_Views.HOME, name='student_home'),

        path('Student/Notification',Student_Views.STUDENT_NOTIFICATION,
name="student_notification"),
        path('Student/mark_as_done/<str:status>',
Student_Views.STUDENT_NOTIFICATION_MARK_AS_DONE,
            name='student_notification_mark_as_done'),

        ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Student_views.py

```
  from django.shortcuts import render,redirect
from app.models import
Student_Notification,Student,Student_Feedback,Student_leave,Subject,Attendance,Att
endance_Report,StudentResult from django.contrib import messages




def HOME(request):
   return render(request,'Student/home.html')



def STUDENT_NOTIFICATION(request):
   student = Student.objects.filter(admin = request.user.id)
```

```
    for i in student:
            student_id = i.id
        notification = Student_Notification.objects.filter(student_id = student_id)

        context = {
            'notification':notification
        }
    return render(request,'Student/notification.html',context)


def STUDENT_NOTIFICATION_MARK_AS_DONE(request,status):
    notification = Student_Notification.objects.get(id = status)
   notification.status = 1     notification.save()
 return redirect('student_notification')



def STUDENT_FEEDBACK(request):
    student_id = Student.objects.get(admin = request.user.id)
feedback_history = Student_Feedback.objects.filter(student_id = student_id)

    context = {
        'feedback_history':feedback_history,
    }
    return render(request,'Student/feedback.html',context)


def STUDENT_FEEDBACK_SAVE(request):
 if request.method == "POST":
        feedback = request.POST.get('feedback')
student = Student.objects.get(admin=request.user.id)
```

```
    feedbacks = Student_Feedback(
                    student_id = student,
                    feedback = feedback,
                     feedback_reply = ""
    )
    feedbacks.save()     return redirect('student_feedback')
```

Staff_views.py

```
  from django.shortcuts import render,redirect
from app.models import
Staff,Staff_Notification,Staff_leave,Staff_Feedback,Subject,Session_Year,Student,Attendance,Attendance_
Report,StudentResult from django.contrib
import messages from django.contrib.auth.decorators
import login_required
@login_required(login_url='/')
def HOME(request):
  return render(request,'Staff/home.html')

@login_required(login_url='/') def NOTIFICATIONS(request):
  staff = Staff.objects.filter(admin = request.user.id)
  for i in staff:
      staff_id = i.id

    notification = Staff_Notification.objects.filter(staff_id = staff_id)

    context = {
      'notification':notification
    }
  return render(request,'Staff/notification.html',context)

@login_required(login_url='/')
```

```
def  STAFF_NOTIFICATION_MARK_AS_DONE(request,status):
    notification = Staff_Notification.objects.get(id = status)
      notification.status = 1
     notification.save()
    return redirect('notifications')


@login_required(login_url='/')
 def STAFF_APPLY_LEAVE(request):
    staff =Staff.objects.filter(admin = request.user.id)
    for i in staff:
     staff_id = i.id

        staff_leave_history = Staff_leave.objects.filter(staff_id = staff_id)

        context = {
            'staff_leave_history':staff_leave_history,
        }
    return render(request,'Staff/apply_leave.html',context)



        leave.save()
        messages.success(request, 'Leave Applications Are Successfully Sent!')
return redirect('staff_apply_leave')


    return None
```

hod_views.py

```
  from  django.shortcuts  import  render,redirect  from  django.contrib.auth.decorators
import login_required
from app.models import
```

```
Course,Session_Year,CustomUser,Student,Staff,Subject,Staff_Notification,Staff_leave,Staff_Feedback,Stud
ent_Notification,Student_Feedback,Student_leave,Attendance,Atten dance_Report from django.contrib
import messages

@login_required(login_url='/')
 def HOME(request):
    student_count=Student.objects.all().count()
staff_count = Staff.objects.all().count()
 course_count = Course.objects.all().count()
subject_count = Subject.objects.all().count()
student_gender_male=Student.objects.filter(gender='Male').count()
student_gender_female = Student.objects.filter(gender = 'Female').count()


    context = {
        'student_count':student_count,
        'staff_count':staff_count,
        'course_count':course_count,
        'subject_count':subject_count,
        'student_gender_male':student_gender_male,
        'student_gender_female':student_gender_female,
    }

    return render(request,'Hod/Home.html',context)

@login_required(login_url='/') def ADD_STUDENT(request):
    course = Course.objects.all()     session_year = Session_Year.objects.all()

    if request.method == "POST":
        profile_pic = request.FILES.get('profile_pic')
     first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
```

```
 email = request.POST.get('email')
    username = request.POST.get('username')
password = request.POST.get('password')
address = request.POST.get('address')
gender = request.POST.get('gender') course_id = request.POST.get('course_id')
session_year_id = request.POST.get('session_year_id')


      ifCustomUser.objects.filter(email=email).exists():
messages.warning(request,'Email IS Already Taken')
 returnredirect('add_student')                                                    if
CustomUser.objects.filter(username=username).exists():
messages.warning(request,'Username IS Already Taken')
return redirect('add_student')
      else:
          user = CustomUser(
first_name = first_name,
last_name = last_name,
username = username,
email = email,
profile_pic = profile_pic,
user_type = 3
          )
          user.set_password(password)
          user.save()

          course = Course.objects.get(id = course_id)
session_year = Session_Year.objects.get(id = session_year_id)

          student = Student(
                    admin = user,
```

```
                address = address,
            session_year_id = session_year,
            course_id = course,
            gender = gender,
        )
        student.save()
        messages.success(request, user.first_name + " " + user.last_name + " Are Successfuly Added!")
        return redirect('add_student')

    context = {
        'course':course,
        'session_year':session_year
    }

    return render(request,'Hod/add_student.html',context)

@login_required(login_url='/') def VIEW_STUDENT(request):
    student = Student.objects.all()

    context = {
        'student':student,
    }
    return render(request,'Hod/view_student.html',context)

@login_required(login_url='/') def EDIT_STUDENT(request,id):
    student = Student.objects.filter(id=id)
course = Course.objects.all()
 session_year = Session_Year.objects.all()

    context = {
     'student': student,
        'course': course,
```

```
        'session_year': session_year,
    }
    return render(request,'Hod/edit_student.html',context)

@login_required(login_url='/')
def UPDATE_STUDENT(request):
if request.method == "POST":
        student_id = request.POST.get('student_id')
 print(student_id)
  profile_pic = request.FILES.get('profile_pic')
 first_name = request.POST.get('first_name')
 last_name = request.POST.get('last_name')
  email = request.POST.get('email')
 username = request.POST.get('username')
  password = request.POST.get('password')
    address = request.POST.get('address')
 gender = request.POST.get('gender')
 course_id = request.POST.get('course_id')
 session_year_id = request.POST.get('session_year_id')

        user = CustomUser.objects.get(id=student_id)

        user.first_name = first_name
  user.last_name = last_name
    user.email = email
  user.username = username

        if password != None and password != "":
           user.set_password(password)
  if profile_pic != None and profile_pic != "":
            user.profile_pic = profile_pic
```

```
user.save()

        student = Student.objects.get(admin = student_id)
 student.address = address
    student.gender = gender

        course = Course.objects.get(id = course_id)
    student.course_id = course

        session_year = Session_Year.objects.get(id=session_year_id)
  student.session_year_id = session_year

        student.save()
 messages.success(request, 'Record Are Successfully Updated !')
 return redirect('view_student')

    return render(request,'Hod/edit_student.html')

@login_required(login_url='/') def DELETE_STUDENT(request,admin):
    student = CustomUser.objects.get(id=admin)
student.delete()     messages.success(request, 'Record Are Successfully Deleted !')
 return redirect('view_student')

@login_required(login_url='/') def ADD_COURSE(request):
    if request.method == "POST":
        course_name = request.POST.get('course_name')

        course = Course(
name=course_name,
        )
        course.save()
```

```
messages.success(request, 'Course Are Successfully Created ')
   return redirect('add_course')
  return render(request,'Hod/add_course.html')

@login_required(login_url='/')
def VIEW_COURSE(request):
    course = Course.objects.all()
  context = {
        'course': course,
    }
    return render(request,'Hod/view_course.html',context)

@login_required(login_url='/')
 def EDIT_COURSE(request,id):
course = Course.objects.get(id = id)

    context = {
        'course': course,
    }
    return render(request,'Hod/edit_course.html',context)

@login_required(login_url='/') def UPDATE_COURSE(request):
  if request.method == "POST":
 name = request.POST.get('name')
course_id = request.POST.get('course_id')

        course = Course.objects.get(id = course_id)
course.name = name
 course.save()
messages.success(request,'Course Are Successfully Update')
return redirect('view_course')
```

```
    return render(request,'Hod/edit_course.html')

@login_required(login_url='/')
def DELETE_COURSE(request,id):
course =Course.objects.get(id = id)
 course.delete()
 messages.success(request,'Course Are Successfully Deteted')

    return redirect('view_course')

@login_required(login_url='/')
def ADD_STAFF(request):
 if request.method == "POST":
        profile_pic = request.FILES.get('profile_pic')
 first_name = request.POST.get('first_name')
 last_name = request.POST.get('last_name')
email = request.POST.get('email')
  username = request.POST.get('username')
  password = request.POST.get('password')
  address = request.POST.get('address')
 gender = request.POST.get('gender')

        if CustomUser.objects.filter(email=email).exists():
     messages.warning(request,'email is Already Taken')
            return redirect('add_staff')
 ifCustomUser.objects.filter(username=username).exists():
messages.warning(request,'username is Already Taken')
 return redirect('add_staff')
         else:
```

```
        user = CustomUser(first_name = first_name,last_name =  last_name,email = email, username = username,profile_pic = profile_pic,user_type =2)
 user.set_password(password)
        user.save()

        staff = Staff(
                admin = user,
               address = address,
              gender = gender
        )
        staff.save()
messages.success(request,'Staff Are Successfully Added !')
  return redirect('add_staff')



   return render(request,'Hod/add_staff.html')

@login_required(login_url='/')
 def VIEW_STAFF(request):
   staff = Staff.objects.all()

   context = {
      'staff':staff,
   }
   return render(request,'Hod/view_staff.html',context)

@login_required(login_url='/')
def EDIT_STAFF(request,id):
  staff = Staff.objects.get(id = id)

   context = {
      'staff':staff,
```

```
    }
    return render(request,'Hod/edit_staff.html',context)

@login_required(login_url='/') def UPDATE_STAFF(request):

  if request.method == 'POST':
        staff_id = request.POST.get('staff_id')
profile_pic = request.FILES.get('profile_pic')
 first_name = request.POST.get('first_name')
 last_name = request.POST.get('last_name')
 email = request.POST.get('email')
    username = request.POST.get('username')
 password = request.POST.get('password')
address = request.POST.get('address')
 gender = request.POST.get('gender')

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

        if password != None and password != "":
            user.set_password(password)
   if profile_pic != None and profile_pic != "":
            user.profile_pic = profile_pic


        user.save()

        staff = Staff.objects.get(admin = staff_id)
```

```
 staff.gender = gender
    staff.address = address

       staff.save()
 messages.success(request,'Staff Is Successfully Updatesd.')
return redirect('view_staff')



   return render(request,'Hod/edit_staff.html')

@login_required(login_url='/')
def DELETE_STAFF(request,admin):
   staff = CustomUser.objects.get(id = admin)
   staff.delete()
messages.success(request,'Recoard Are Successfully Deleted.')
   return redirect('view_staff')

@login_required(login_url='/')
def ADD_SUBJECT(request):
 course = Course.objects.all()
  staff = Staff.objects.all()

   if request.method == "POST":
       subject_name = request.POST.get('subject_name')
  course_id = request.POST.get('course_id')
 staff_id = request.POST.get('staff_id')

       course =Course.objects.get(id = course_id)
 staff = Staff.objects.get(id = staff_id)
```

```
        subject = Subject(
name = subject_name,
course = course,
 staff = staff,
        )
        subject.save()
    messages.success(request,'Subject Are Successfully Added !')
 return redirect('add_subject')

    context = {
        'course':course,
        'staff':staff,
    }
    return render(request,'Hod/add_subject.html',context)

@login_required(login_url='/')
def VIEW_SUBJECT(request):
    subject = Subject.objects.all()

    context = {
        'subject':subject,
    }
    return render(request,'Hod/view_subject.html',context)

@login_required(login_url='/')
def EDIT_SUBJECT(request,id):
subject = Subject.objects.get(id = id)
course = Course.objects.all()
 staff = Staff.objects.all()
 context = {
        'subject':subject,
        'course':course,
```

```
        'staff':staff,
    }

    return render(request,'Hod/edit_subject.html',context)

@login_required(login_url='/')
 def UPDATE_SUBJECT(request):
if request.method == 'POST':
        subject_id = request.POST.get('subject_id')
   subject_name = request.POST.get('subject_name')
     course_id = request.POST.get('course_id')

   staff_id = request.POST.get('staff_id')


        course = Course.objects.get(id = course_id)
staff = Staff.objects.get(id = staff_id)

        subject = Subject(
  id = subject_id,
   name = subject_name,
      course = course,
     staff = staff,
        )
        subject.save()
        messages.success(request, 'Subject Are Successfully Updated!')
    return redirect('view_subject')

@login_required(login_url='/')
 def DELETE_SUBJECT(request,id):
  subject =Subject.objects.filter(id = id)
```

```
 subject.delete()
   messages.success(request,'Subject Are Successfully Deteted !')

    return redirect('view_subject')


@login_required(login_url='/')
 def ADD_SESSION(request):
    if request.method == 'POST':
        session_year_start = request.POST.get('session_year_start')
     session_year_end = request.POST.get('session_year_end')

        session = Session_Year(
   session_start = session_year_start,
    session_end = session_year_end,
        )
        session.save()
   messages.success(request,'Session Year Are Successfully Created ')
    return redirect('add_session')

    return render(request,'Hod/add_session.html')

@login_required(login_url='/')
 def VIEW_SESSION(request):
    session = Session_Year.objects.all()

    context = {
        'session':session,
    }
```

```
    return render(request,'Hod/view_session.html',context)

@login_required(login_url='/')
def EDIT_SESSION(request,id):
    session = Session_Year.objects.filter(id = id)

    context = {
        'session': session,
    }
    return render(request,'Hod/edit_session.html',context)

@login_required(login_url='/')
 def UPDATE_SESSION(request):
   if request.method == 'POST':
        session_id = request.POST.get('session_id')
session_year_start = request.POST.get('session_year_start')
 session_year_end = request.POST.get('session_year_end')

        session = Session_Year(
    id = session_id,
session_start = session_year_start,
       session_end = session_year_end,
        )
        session.save()
        messages.success(request,'Session Are Successfully Updated !')
        return redirect('view_session')

@login_required(login_url='/')
def DELETE_SESSION(request,id):
    session = Session_Year.objects.get(id = id)
   session.delete()
```

```
  messages.success(request,'Session Are Successfully Deleted !')

    return redirect('view_session')

@login_required(login_url='/')
def STAFF_SEND_NOTIFICATION(request):
    staff = Staff.objects.all()
 see_notification = Staff_Notification.objects.all().order_by('-id')[0:5]

    context = {
        'staff':staff,
        'see_notification':see_notification,
    }
    return render(request,'Hod/staff_notification.html',context)


@login_required(login_url='/')
def SAVE_STAFF_NOTIFICATION(request):
  if request.method == "POST":
        staff_id = request.POST.get('staff_id')
     message = request.POST.get('message')

        staff = Staff.objects.get(admin = staff_id)
    notification = Staff_Notification(
staff_id = staff,
            message = message,
        )
        notification.save()
 messages.success(request,'Notification Are Successfully Send')

def STUDENT_SEND_NOTIFICATION(request):
```

```
    student = Student.objects.all()
notification = Student_Notification.objects.all()
context = {
        'student':student,
        'notification':notification,
    }
    return render(request,'Hod/student_notification.html',context)


def SAVE_STUDENT_NOTIFICATION(request):
  if request.method == "POST":
        message = request.POST.get('message')
 student_id = request.POST.get('student_id')

        student = Student.objects.get(admin = student_id)

        stud_notification = Student_Notification
student_id = student,            message = message,
        )
        stud_notification.save()
   messages.success(request,'Student Notification Are Successfully Sent')
   return redirect('student_send_notification')
```

base.html

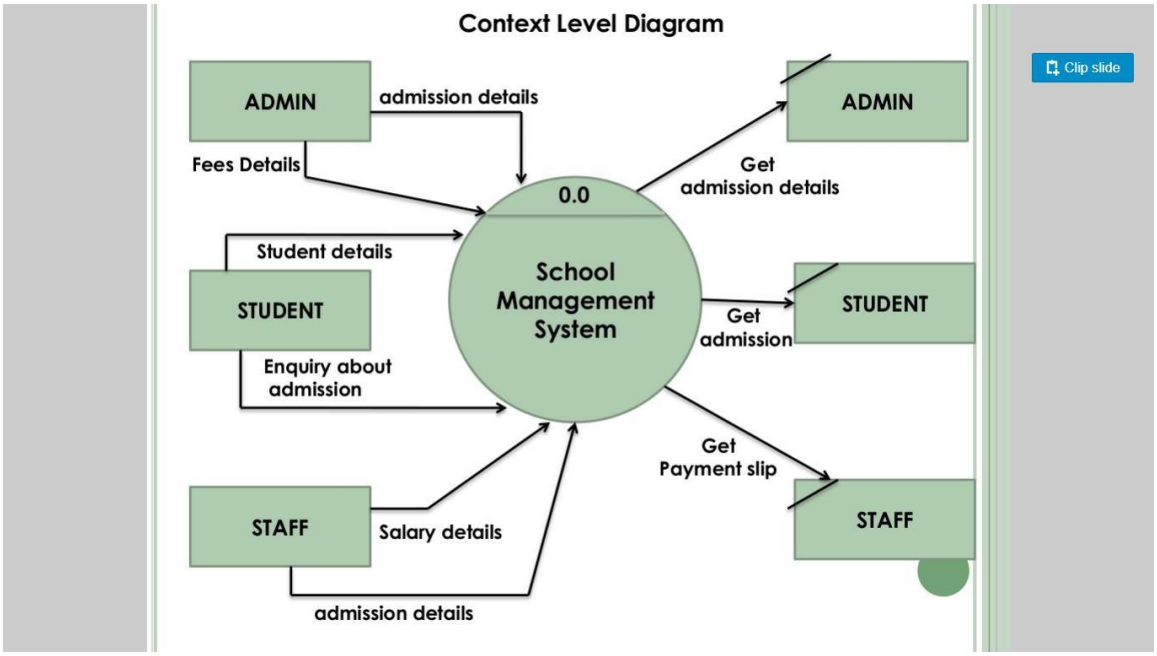
```
<!DOCTYPE html>
<html lang="en">
{% load static %}
   <head>
      <meta charset="utf-8">      <meta name="viewport" content="width=device-
width,   initialscale=1.0,   user-scalable=0">                    <title>MMIT
SIDDHARTHANAGAR</title>          <link rel="shortcut icon" href="{% static
'assets/img/favicon.png'   %}">                      <link   rel="stylesheet"
href="https://fonts.googleapis.com/css2?fami
ly=Poppins:ital,wght@0,500;0,600;0,700;1,400
&amp;display=swap">
      <link rel="stylesheet" href="{% static
'assets/plugins/bootstrap/css/bootstrap.min.
css' %}">
      <link rel="stylesheet" href="{% static
'assets/plugins/fontawesome/css/fontawesome. min.css' %}">
      <link            rel="stylesheet"           href="{%            static
'assets/plugins/fontawesome/css/all.min.css'
%}">
      <link rel="stylesheet" href="{% static
'assets/css/style.css' %}">
      <!--  datatable -->        <script
src="https://code.jquery.com/jquery1.11.1.min.js"></script>
       <link rel="stylesheet" href="{% static
'assets/plugins/datatables/datatables.min.cs s' %}">

   </head>
   <body>
<div class="main-wrapper">

       {% include 'includes/header.html' %}

         {% include 'includes/sidebar.html'
%}

<div class="page-wrapper">
    <div class="content container-fluid">
         {% block content %}
         {% endblock %}
          {% include 'includes/footer.html'
%}


      </div>
</div>
      <script src="{% static
```

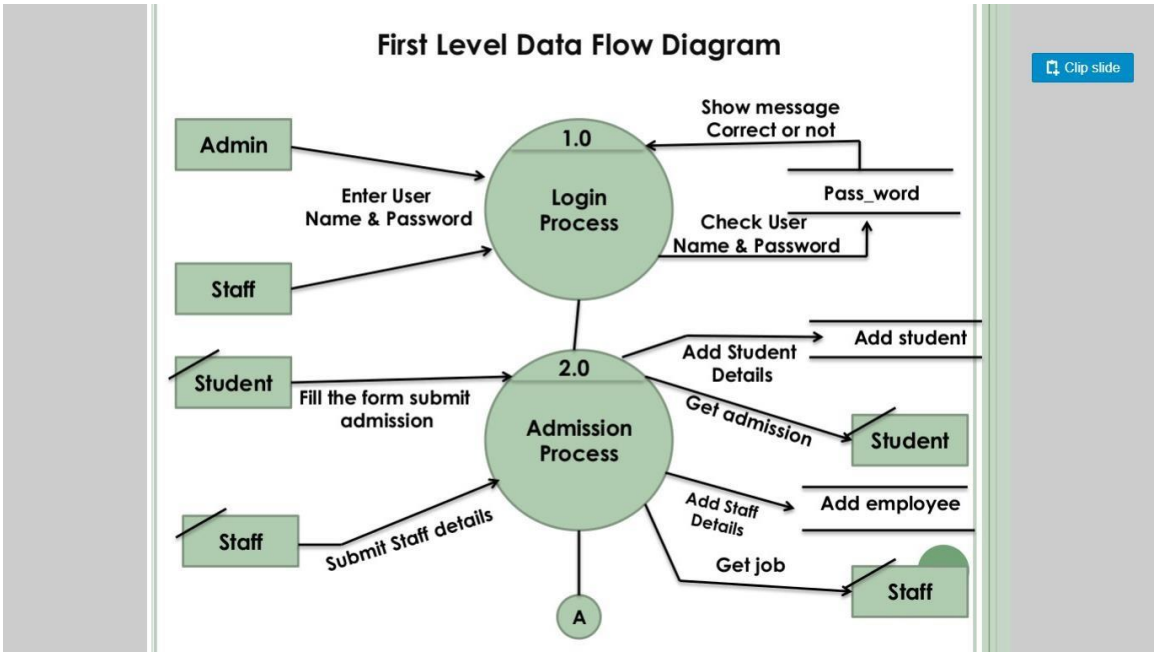
```
'assets/js/jquery-3.6.0.min.js'
%}"></script>
      <script src="{% static
'assets/js/popper.min.js' %}"></script>
      <script src="{% static
'assets/plugins/bootstrap/js/bootstrap.min.j s' %}"></script>
 <script src="{% static
'assets/plugins/slimscroll/jquery.slimscroll
.min.js' %}"></script>
      <script src="{% static
'assets/plugins/apexchart/apexcharts.min.js'
%}"></script>
      <script src="{% static
'assets/plugins/apexchart/chart-data.js'
%}"></script>
<script src="{% static 'assets/js/script.js' %}"></script>
script src="{% static 'assets/plugins/datatables/datatables.min.js
'%}"></script>       <script
src="https://cdn.datatables.net/1.10.4/js/jquery.dataTables.min.js">
</script>
  <script type="text/javascript">
    $(document).ready(function () {
        $('#table_id').dataTable();
    });
       </script>

   </body>

</html>
```

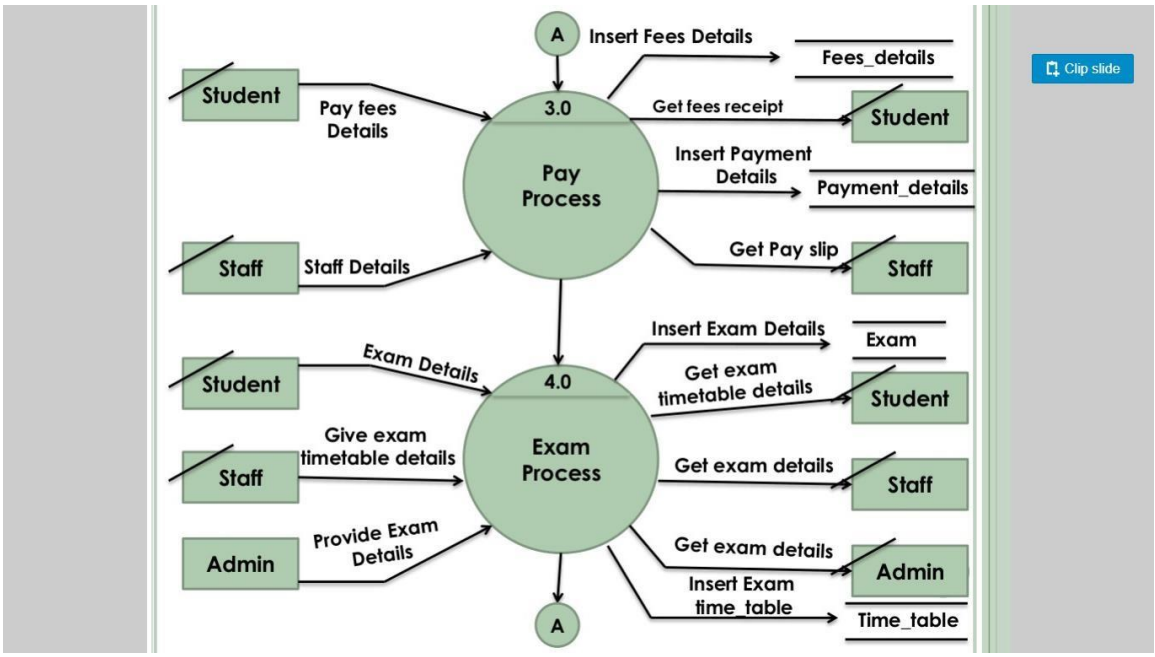
16.Future Enhancements & Conclusion

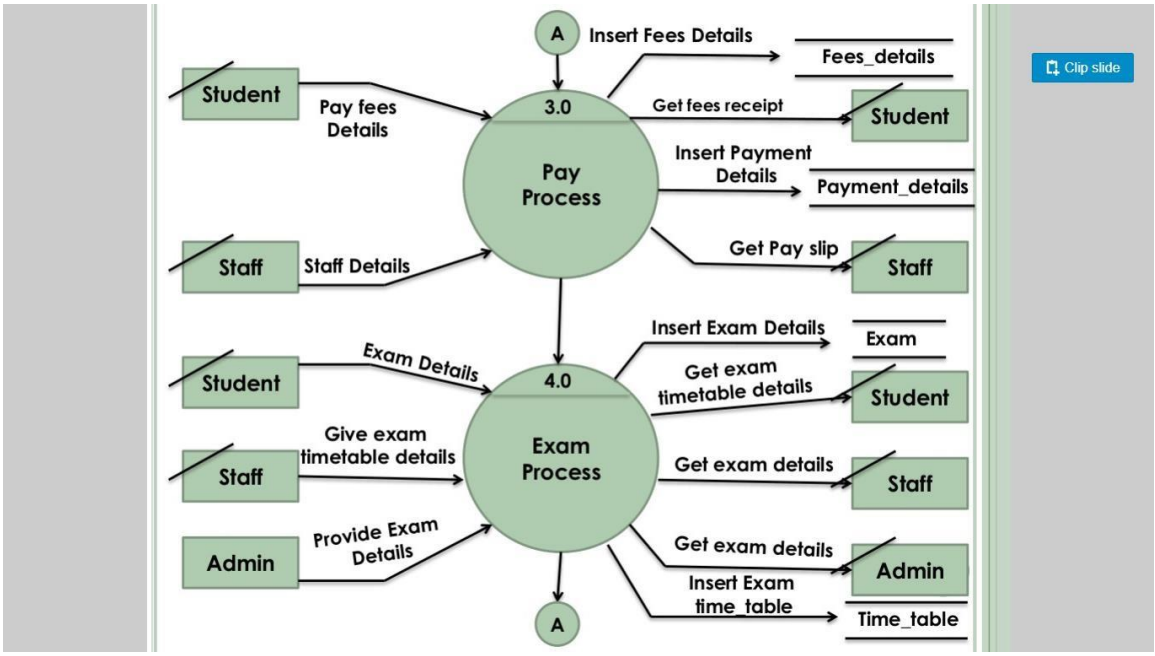
The project entitled Student Management system was completed successfully.The results obtained from the experiments and testing ensures that the proposed method is efficient and user-friendly. As compared to existing methods of managing the academic institutions, this project which yields centralized software makes the work administration and management easier and provides detailed information about the topic of users interest in just one mouse click. The educational institution can be provided with an easy-to-use user interface centralized software in which all services associated with the institution can interact with each other and share the data. As this is a Rest API hosted in the AWS Cloud server, the user will be able to access the resources from remote places. As the application is developed using micro-service architecture and agile methodology, in the future services can be added without having to make changes to the existing code.

17. References



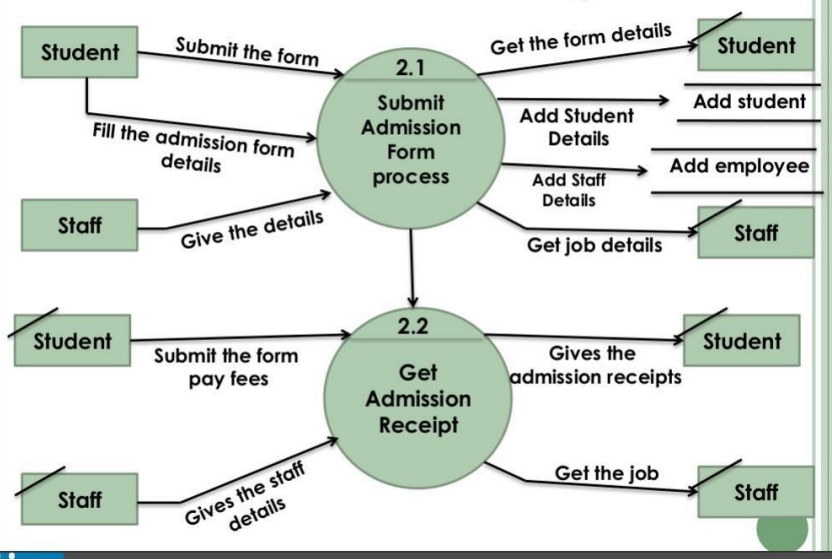



A
Insert Fees Details
Fees_details
Student
Pay fees Details
3.0
Pay Process
Get fees receipt
Student
Insert Payment Details
Payment_details
Staff
Staff Details
Get Pay slip
Staff
Insert Exam Details
Exam
Student
Exam Details
4.0
Get exam timetable details
Student
Give exam timetable details
Exam Process
Staff
Get exam details
Staff
Admin
Provide Exam Details
Get exam details
Admin
Insert Exam time_table
Time_table
A
Clip slide


A
Insert Fees Details
Fees_details
Student
Pay fees Details
3.0
Pay Process
Get fees receipt
Student
Insert Payment Details
Payment_details
Staff
Staff Details
Get Pay slip
Staff
Insert Exam Details
Exam
Student
Exam Details
4.0
Get exam timetable details
Student
Give exam timetable details
Exam Process
Staff
Get exam details
Staff
Admin
Provide Exam Details
Get exam details
Admin
Insert Exam time_table
Time_table
A
Clip slide

School Management System

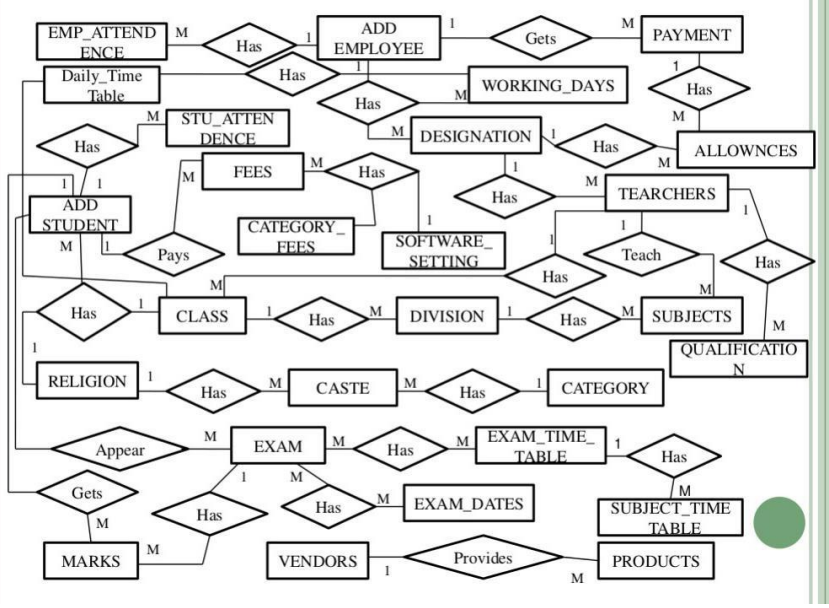
Second Level Data Flow Diagram



Clip slide

15 of 78

Entity Relationship Diagram



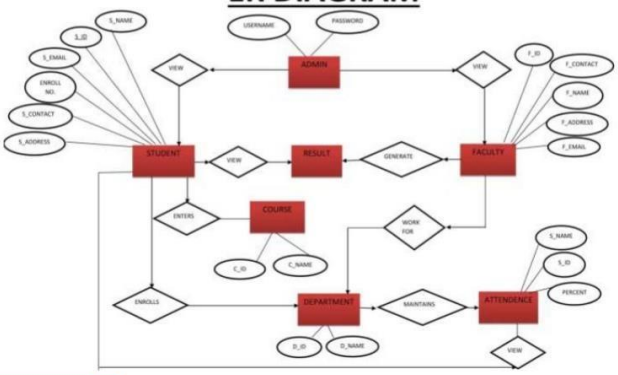
Clip slide

Student Management System best PPT

Clip slide

SYSTEM DESIGN

ER DIAGRAM



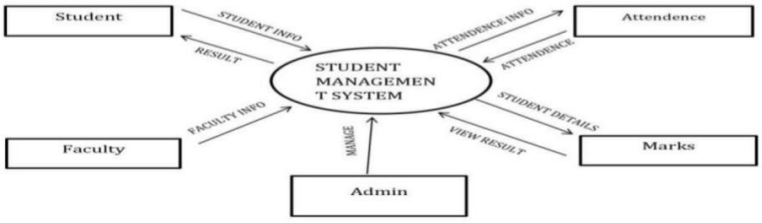
7 of 21

Clip slide

DATA FLOW DIAGRAM

- ***CONTEXT LEVEL DFD***

- ***LEVEL 1 DFD***

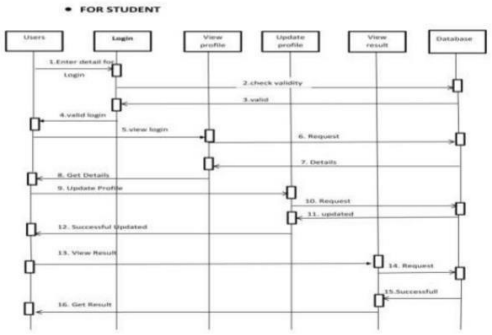


Student Management System best PPT

Sequence Diagram

Press Esc to exit full screen

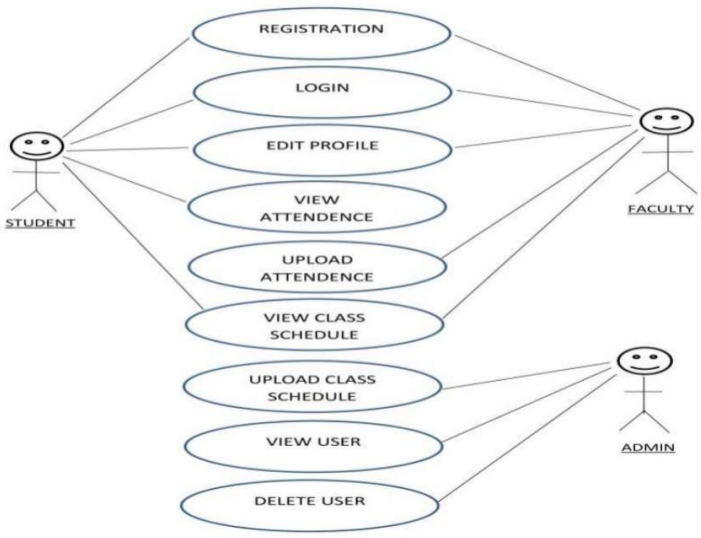
Clip slide

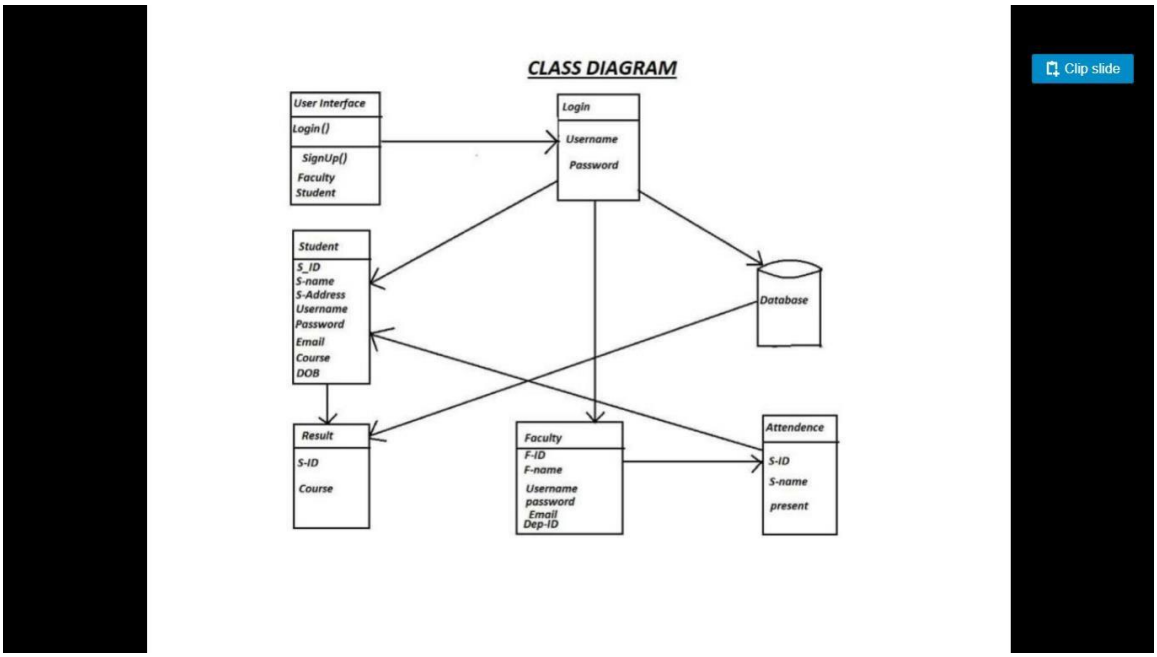


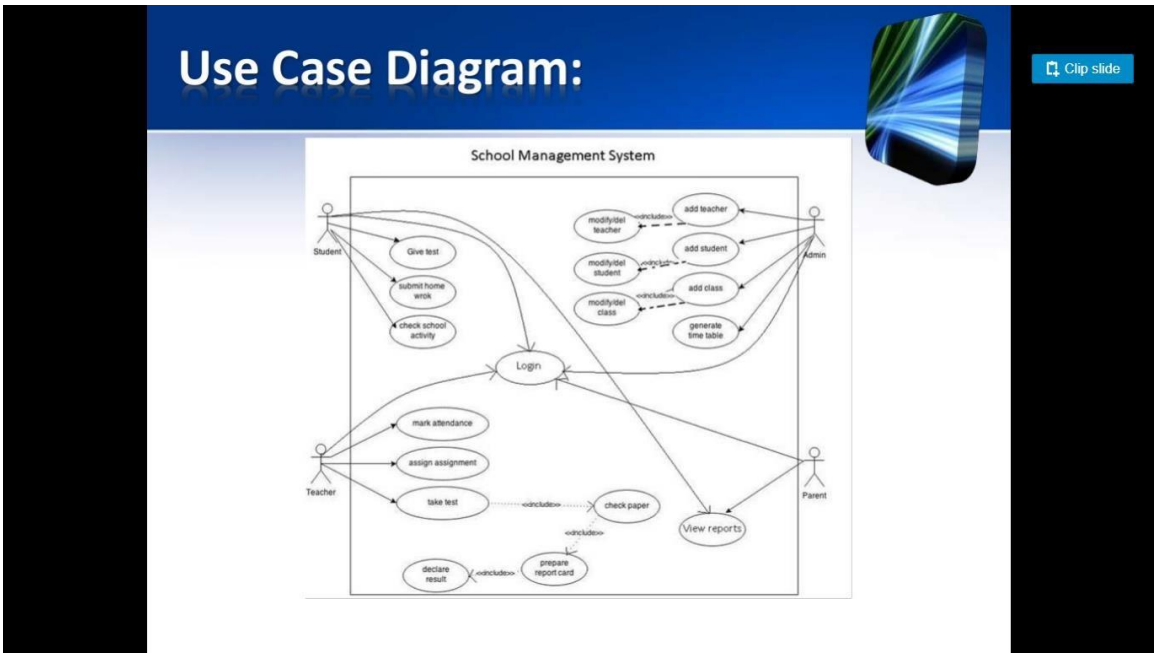
9 of 21

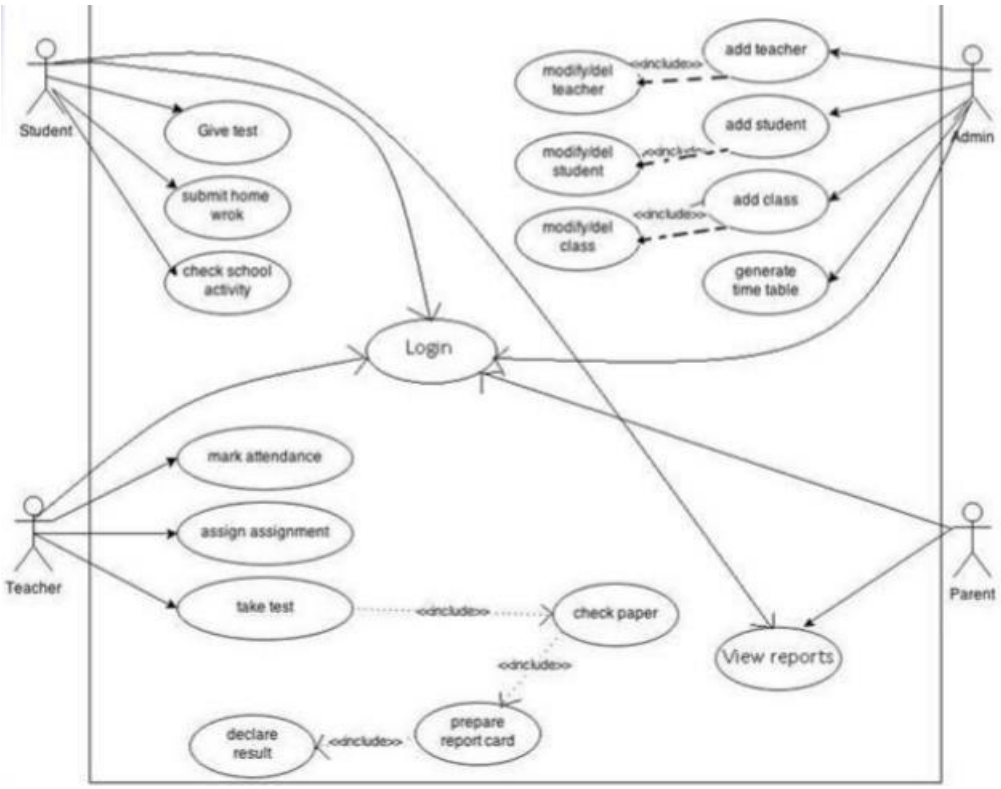
Clip slide

USE CASE DIAGRAM

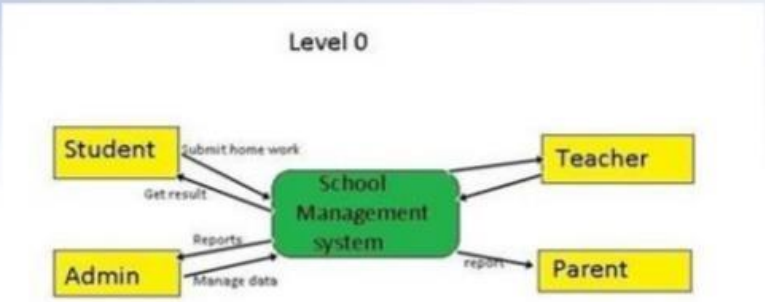


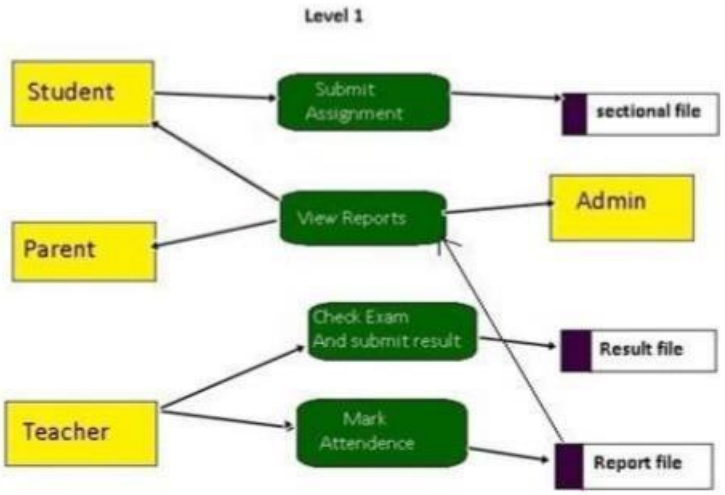

CLASS DIAGRAM
User Interface
Login()
SignUp()
Faculty
Student
Login
Username
Password
Student
S_ID
S-name
S-Address
Username
Password
Email
Course
DOB
Database
Result
S-ID
Course
Faculty
F-ID
F-name
Username
password
Email
Dep-ID
Attendence
S-ID
S-name
present
Clip slide


Use Case Diagram:
School Management System
Student
Give test
submit home wrok
check school activity
modify/del teacher
add teacher
add student
modify/del student
add class
modify/del class
generate time table
Admin
Login
mark attendance
assign assignment
take test
check paper
View reports
Teacher
Parent
declare result
prepare report card
Clip slide

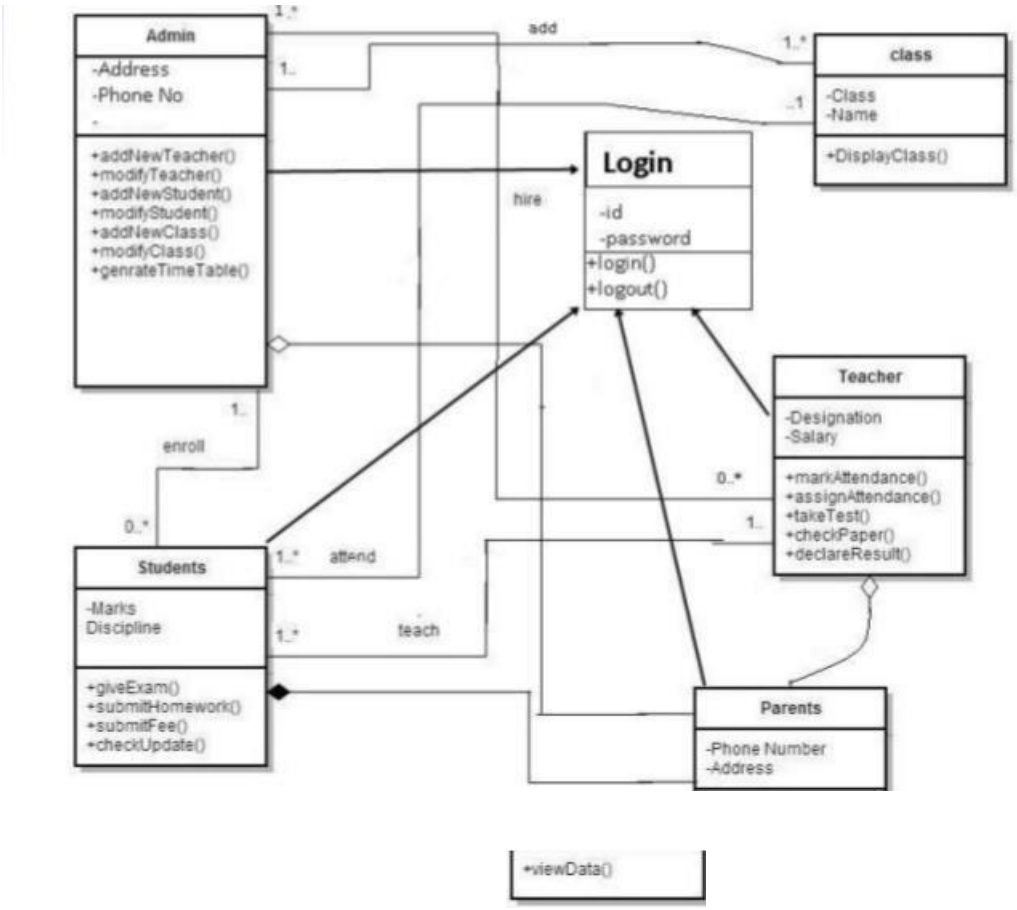


Data flow diagram

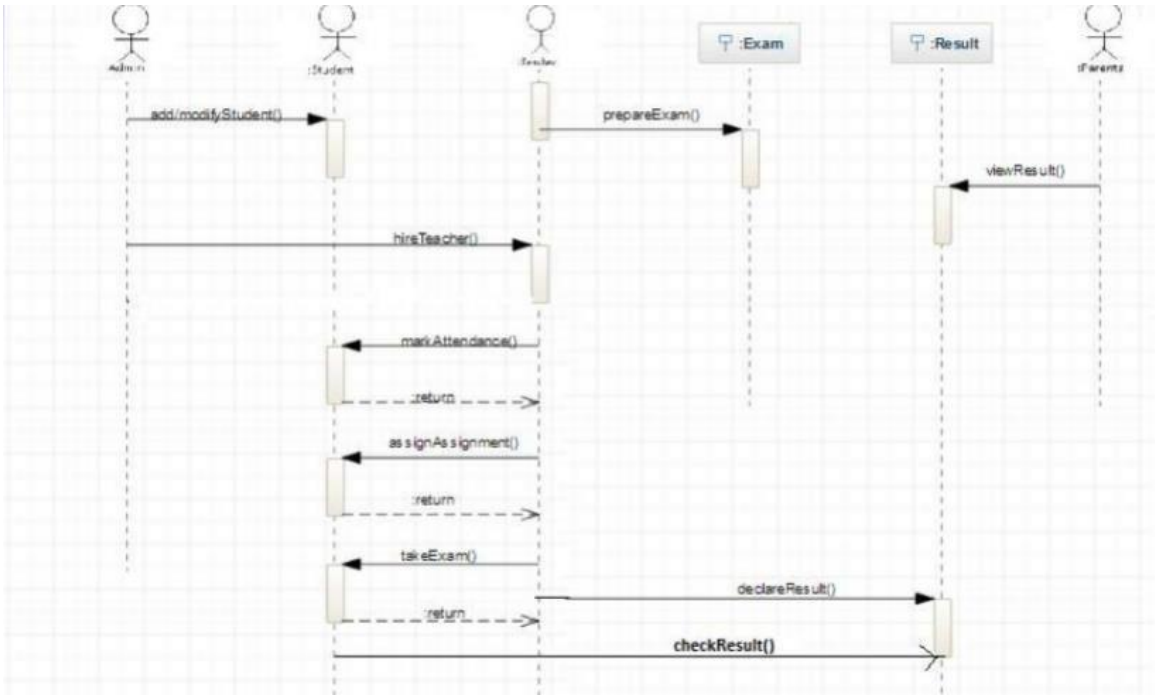




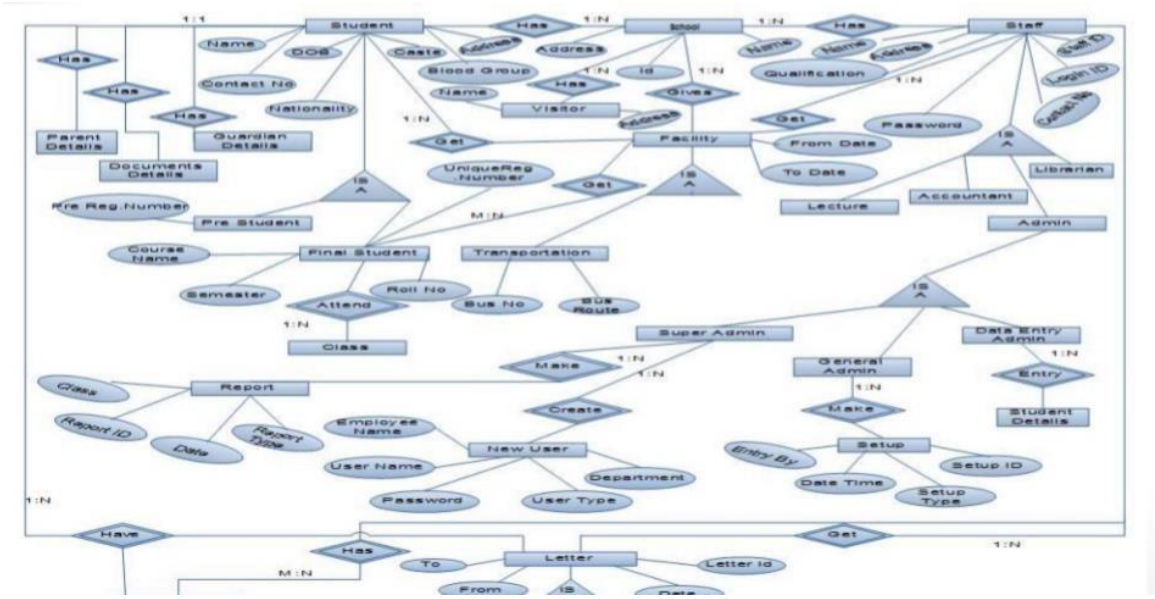
Class Diagram:

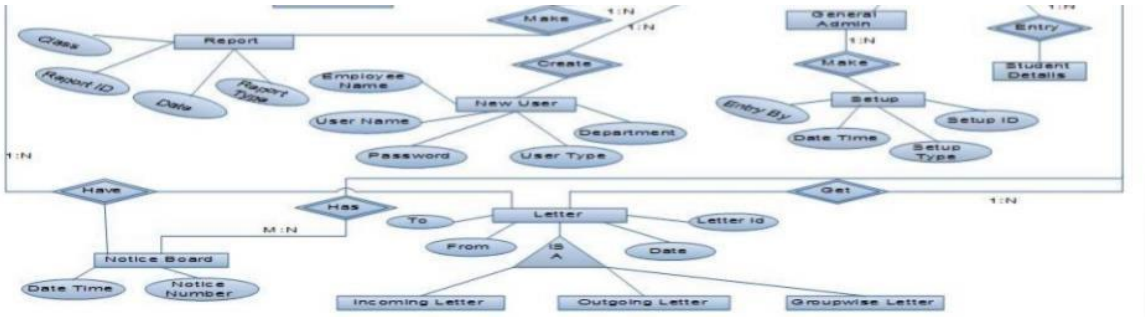


Sequence dia:

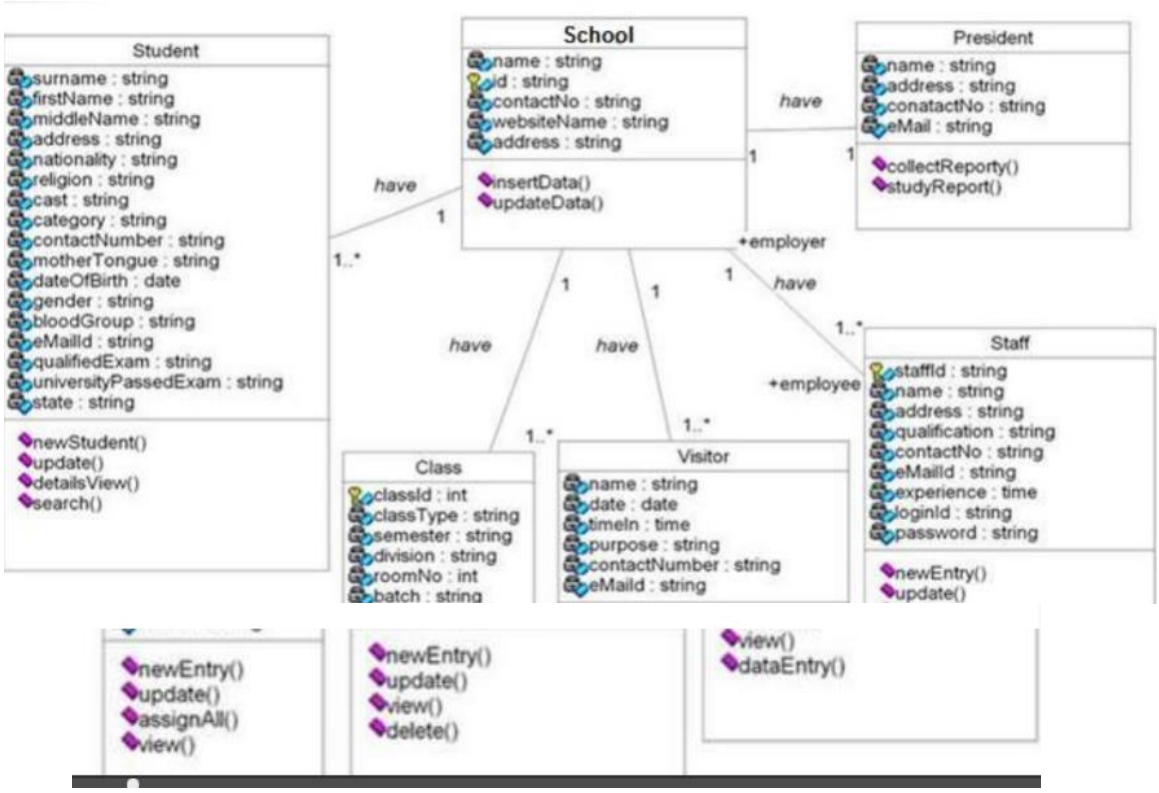

:Exam
:Result
add/modifyStudent()
prepareExam()
viewResult()
hireTeacher()
markAttendance()
:return
assignAssignment()
:return
takeExam()
declareResult()
:return
checkResult()

ERD:

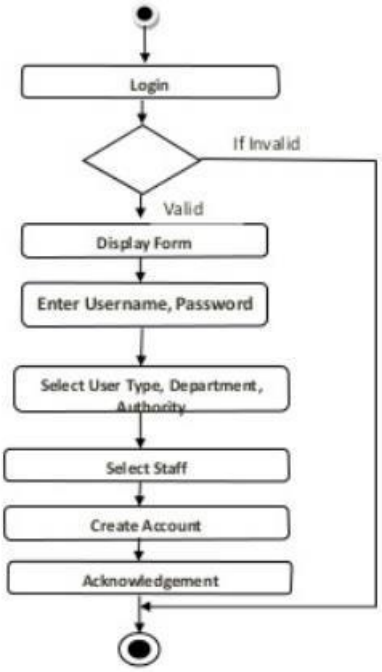

Student
School
Staff
Parent Details
Guardian Details
Documents Details
Pre Student
Final Student
Transportation
Facility
Visitor
Lecture
Accountant
Librarian
Admin
Super Admin
General Admin
Data Entry Admin
Class
Report
New User
Letter
Setup
Student Details

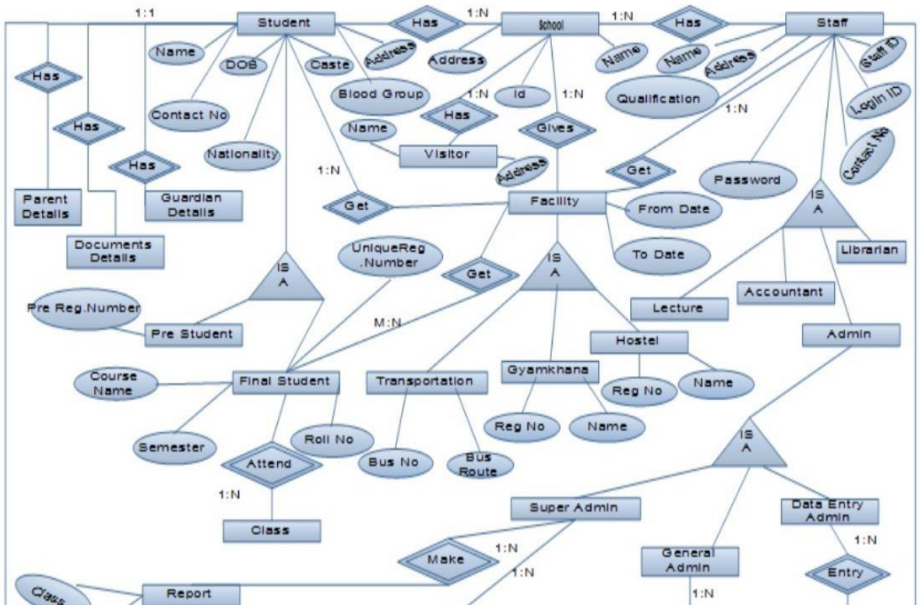


Class Dia:



Activity Diagram:


Login
If Invalid
Valid
Display Form
Enter Username, Password
Select User Type, Department, Authority
Select Staff
Create Account
Acknowledgement


Student
School
Staff
Has
Name
DOB
Caste
Address
Blood Group
Contact No
Nationality
Parent Details
Guardian Details
Documents Details
Pre Reg.Number
Pre Student
Course Name
Final Student
Semester
Roll No
Attend
Class
Bus No
Transportation
Bus Route
Id
Qualification
Login ID
Staff ID
Password
Contact No
Visitor
Gives
Get
Facility
From Date
To Date
UniqueReg .Number
IS A
Librarian
Accountant
Lecture
Admin
Hostel
Gymkhana
Reg No
Super Admin
Data Entry Admin
General Admin
Entry
Make
Report
1:1
1:N
M:N