-

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report

of

Internet Technologies (CSE 3162)

TITLE: Practice School Management System

SUBMITTED
BY

Vinayak Joshi (Reg. no. 210905270) Roll. No.43
Yashas Ranjan (Reg. no. 210905390) Roll. No. 61
Kaushal Singh (Reg. no. 210905404) Roll. No. 63
Shaswat Kumar (Reg.no.210905224) Roll.No.34

Under the Guidance of:
Ms. Rajashree Krishna and Dr. Shwetha Rai
Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal, Karnataka – 576104

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
04/04/2024

CERTIFICATE

This is to certify that the project titled Practice School Management System is a record of the Bonafede work done by Vinayak Joshi (Reg. No. 210905270), Yashas Ranjan (Reg. no. 210905390), Kaushal Singh (Reg. no. 210905404) and Shaswat Kumar (Reg. no. 210905224) submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2023-2024.

Name and Signature of Examiners:

1. Ms. Rajashree Krishna, Professor, CSE Dept

2. Dr. Shwetha Rai, Assistant Professor, CSE Dept.

TABLE OF CONTENTS

ABSTRACT:

The Practice School Management System is an innovative solution designed to streamline administrative tasks and enhance communication within educational institutions. This project aims to address the challenges faced by both students and teachers in managing projects and internships effectively. The SMS consists of separate portals for students and teachers, facilitating seamless interaction and collaboration. Students can apply for projects and internships through the student portal, while teachers can update available opportunities through the teacher portal.

The system ensures efficient management of projects and internships, thereby promoting academic and professional growth. This report provides an overview of the SMS project, detailing its objectives, methodology, implementation process, and conclusions drawn from its deployment. Through the SMS, educational institutions can foster a conducive environment for learning and development, empowering students and teachers alike.

This report provides a detailed exploration of the SMS project, outlining its objectives, methodology, and implementation process. Leveraging modern technologies and agile development methodologies, the SMS has been meticulously crafted to meet the diverse needs of educational institutions while ensuring scalability and flexibility for future enhancements.

CHAPTER 1: INTRODUCTION

The efficient management of resources, communication, and administrative tasks is paramount to the success of educational institutions. With the advent of technology, there has been a growing demand for innovative solutions that streamline these processes and enhance the overall learning experience. In response to these evolving needs, the Practice School Management System project emerges as a comprehensive solution designed to revolutionize the management practices within educational institutions.

Educational institutions, ranging from schools to universities, often grapple with numerous administrative challenges, including project and internship management, communication gaps between students and teachers, and the cumbersome task of resource allocation. Traditional methods of managing these tasks, such as manual record-keeping and disjointed communication channels, are no longer sustainable in today's digital age. Recognizing these challenges, the SMS project aims to bridge these gaps and transform the educational landscape by leveraging technology to streamline administrative processes and enhance collaboration.

The primary objective of the SMS project is to develop a user-friendly, scalable, and efficient platform that caters to the specific needs of students, teachers, and administrators within educational institutions.

The scope of the SMS project encompasses the development and deployment of a webbased platform that consists of separate portals for students and teachers. The student portal enables learners to explore, apply for, and track their involvement in various projects and internships, while the teacher portal empowers educators to manage and update available opportunities. Additionally, the SMS project encompasses the integration of features such as real-time notifications, progress tracking, and data analytics to provide stakeholders with valuable insights and enhance the overall user experience.

CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES

Problem Statement:

Educational institutions face various administrative challenges that hinder the efficient management of resources and communication between stakeholders. These challenges include:

- Manual and inefficient processes for project and internship management.
- Lack of centralized platform for students to explore and apply for opportunities.
- Communication gaps between students seeking projects/internships and teachers offering them.
- Difficulty in tracking and monitoring student involvement in projects and internships.

Objectives:

1. Streamlined Project and Internship Management: Develop a user-friendly platform that simplifies the process of posting, applying for, and managing projects and internships within educational institutions.

2. Enhanced Communication Channels: Establish seamless communication channels between students and teachers to facilitate the exchange of project/internship opportunities, updates, and feedback.

3. Improved Accessibility and Transparency: Create a centralized repository of projects and internships accessible to all stakeholders, promoting transparency and equal opportunity for students.

4. Efficient Resource Allocation: Optimize resource allocation by providing administrators with insights into student involvement in projects and internships, enabling better planning and utilization of resources.

5. Enhanced User Experience: Prioritize the development of a user-friendly interface and intuitive features that enhance the overall experience for students, teachers, and administrators using the SMS platform.

6. Scalability and Flexibility: Design the SMS platform to be scalable and adaptable to the evolving needs of educational institutions, allowing for seamless integration of additional features and functionalities in the future.

CHAPTER 3: METHODOLOGY

Project Planning:

The methodology employed in the development of the Practice School Management System involved a systematic approach to project planning, execution, and evaluation. The project was divided into several phases, each with its specific objectives and deliverables:

- Requirement Analysis: Conducted extensive stakeholder interviews and surveys to gather requirements from students, teachers, and administrators regarding their needs and pain points in project and internship management.

- System Design: Based on the gathered requirements, designed the system architecture, database schema, and user interface wireframes to ensure the alignment of the SMS with the identified needs and objectives.

- Development: Utilized agile development methodologies, such as Scrum, to iteratively develop and refine the SMS platform. Regular sprint cycles allowed for continuous feedback and adaptation to changing requirements throughout the development process.

The development of the Practice School Management System followed a structured methodology encompassing project planning, technology selection, implementation, and evaluation. The project planning phase commenced with an in-depth requirement analysis, involving stakeholder interviews, surveys, and feedback sessions to identify the needs and preferences of students, teachers, and administrators. This phase laid the groundwork for subsequent activities by defining the scope, objectives, and key features of the SMS platform. Following requirement analysis, the system design phase focused on translating user needs into concrete design specifications, including architectural design and user interface design. Resource allocation and timeline planning were integral aspects of project planning to ensure the efficient utilization of resources and adherence to project deadlines. This involved assembling a multidisciplinary project team, estimating project effort, and developing a detailed project schedule to track progress effectively.

Throughout the development process, continuous evaluation and feedback were solicited from stakeholders to address issues, improve usability, and enhance overall user satisfaction. This iterative approach to development, coupled with proactive risk management and adherence to best practices, contributed to the successful implementation of the Practice School Management System, paving the way for enhanced communication, collaboration, and administrative efficiency within educational institutions.

Resource Allocation and Timeline Planning:

Resource allocation and timeline planning were crucial aspects of project planning to ensure the efficient utilization of resources and adherence to project deadlines. This involved:

- Identifying Project Team: Assembling a multidisciplinary team comprising developers, designers, testers, and project managers with the requisite skills and expertise to execute the SMS project.

- Estimating Project Effort: Estimating the time and effort required for each phase of the project, considering factors such as complexity of features, availability of resources, and potential risks.

- Developing Project Schedule: Creating a detailed project schedule outlining key milestones, deliverables, and dependencies to track progress and manage deadlines effectively.

Risk Management:

Proactive risk management was integral to the project planning process to anticipate and mitigate potential challenges that could impact project success. This involved:

- Identifying Risks: Conducting risk assessments to identify potential threats and vulnerabilities that could affect project scope, schedule, or quality.

- Risk Mitigation Strategies: Developing contingency plans and mitigation strategies to address identified risks, such as resource constraints, technology dependencies, or changes in project requirements.

CHAPTER 4: IMPLEMENTATION

Account:

Forms.py:

```
from django import forms from django.contrib.auth.forms import
UserCreationForm, AuthenticationForm, UserChangeForm from
django.contrib.auth.models import User
from .models import StudentProfile, TeacherProfile

class StudentRegistrationForm(UserCreationForm):
registration_number = forms.CharField(max_length=20)
address = forms.CharField(widget=forms.Textarea)
phone = forms.CharField(max_length=15)     department =
forms.CharField(max_length=100)     batch =
forms.CharField(max_length=20)

    class Meta:
        model = User        fields = ['first_name', 'last_name', 'username',
'email', 'password1', 'password2', 'registration_number', 'address', 'phone',
'department', 'batch']

class TeacherRegistrationForm(UserCreationForm):
teacher_id = forms.CharField(max_length=20)     address
= forms.CharField(widget=forms.Textarea)     phone =
forms.CharField(max_length=15)
    department = forms.CharField(max_length=100)
    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'username', 'email', 'password1',
'password2', 'teacher_id', 'address', 'phone', 'department']

class StudentProfileForm(UserChangeForm):
class Meta:
        model = StudentProfile
        fields = ['registration_number', 'address', 'phone', 'department',
'batch']

class TeacherProfileForm(UserChangeForm):
class Meta:          model = TeacherProfile
        fields = ['teacher_id', 'address', 'phone', 'department']
```

models.py:

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

class StudentProfile(models.Model):      user =
models.OneToOneField(User, on_delete=models.CASCADE)
registration_number = models.CharField(max_length=20)
    address = models.TextField()     phone =
models.CharField(max_length=15)     department =
models.CharField(max_length=100)     batch =
models.CharField(max_length=20)
```

```
    def update_profile(self, **kwargs):
for field, value in kwargs.items():
            setattr(self, field, value)
self.save()

    def __str__(self):
        return self.user.username + ' (Student)'

class TeacherProfile(models.Model): user =
models.OneToOneField(User, on_delete=models.CASCADE)
teacher_id = models.CharField(max_length=20) address =
models.TextField()
    phone = models.CharField(max_length=15)
department = models.CharField(max_length=100)

    def update_profile(self, **kwargs):
for field, value in kwargs.items():
            setattr(self, field, value)
self.save()

    def __str__(self):
        return self.user.username + ' (Teacher)'
```

urls.py

```
from django.urls import path
from . import views
```

```
urlpatterns = [     #Profile Page      path('',
views.profile_view, name='profile_view'),
#Generic Registeration Page
    path('register/', views.register_view, name='register_common'),

    #Called on basis of what is selected above
path('register/student/',  views.register_student,
name='register_student'),
    path('register/teacher/', views.register_teacher,
name='register_teacher'),

    #Generic Login/Logout Page     path('login/',
views.login_view, name='login_view'),     path('logout/',
views.logout_view, name='logout_view'),

    #Generic Edit if needed      path('edit/',
views.edit_profile_view, name='edit_profile_view')
]
```

Views.py:

```
from django.shortcuts import render, redirect from
django.contrib.auth import login, logout, authenticate from
django.contrib.auth.forms import
AuthenticationForm from django.contrib.auth.models import
Permission from django.contrib.auth.decorators import
login_required
```

```
from .models import StudentProfile, TeacherProfile  from .forms import
StudentRegistrationForm,                    TeacherRegistrationForm,
StudentProfileForm, TeacherProfileForm

def register_view(request):      return
render(request, 'registration.html')  def
register_student(request):     if
request.method == 'POST':         form =
StudentRegistrationForm(request.POST)
if form.is_valid():
user_profile = form.save()
student_profile = StudentProfile(
            user               = user_profile,
registration_number    =
form.cleaned_data.get("registration_number"),
            address            = form.cleaned_data.get("address"),
phone              = form.cleaned_data.get("phone"),               department
= form.cleaned_data.get("department"),               batch              =
form.cleaned_data.get("batch")
         )
student_profile.save()
permissions = Permission.objects.filter(
content_type__app_label='dashboard',              codename__in=[
               'can_apply_to_internship',
               'can_view_applied_internships',
            ]
         )
         for permission in permissions:
user_profile.user_permissions.add(permission)
```

```
            login(request, user_profile)            return
redirect('dashboard_view')        else:            print(form.errors)
return render(request, 'student_registration.html', {'form': form,
'error': form.errors})     else:
        form = StudentRegistrationForm()
    return render(request, 'student_registration.html', {'form': form})

def register_teacher(request):     if request.method ==
'POST':          form =
TeacherRegistrationForm(request.POST)
if form.is_valid():
            user_profile = form.save()
teacher_profile = TeacherProfile(
user        = user_profile,
                teacher_id  = form.cleaned_data.get("teacher_id"),
address     = form.cleaned_data.get("address"),                phone
= form.cleaned_data.get("phone"),                department  =
form.cleaned_data.get("department"),

            )
teacher_profile.save()
permissions = Permission.objects.filter(
content_type__app_label='dashboard',                codename__in=[
'can_post_internship',
                    'can_edit_internship',
                    'can_delete_internship',
                    'can_view_internship_applications',
                ]
            )
            for permission in permissions:
                user_profile.user_permissions.add(permission)

            login(request, user_profile)            return
redirect('dashboard_view')         else:             return render(request,
'teacher_registration.html', {'form': form,  'error': "Something went
wrong."})     else:
        form = TeacherRegistrationForm()
    return render(request, 'teacher_registration.html', {'form': form})

@login_required def
edit_profile_view(request):      if
hasattr(request.user, 'studentprofile'):
profile = request.user.studentprofile
ProfileForm = StudentProfileForm
editTemplate = "edit_student_profile.html"
elif hasattr(request.user, 'teacherprofile'):
profile = request.user.teacherprofile
ProfileForm = TeacherProfileForm
editTemplate =
"edit_teacher_profile.html"     else:
return redirect('logout')

    if request.method == 'POST':
        form = ProfileForm(request.POST, instance=profile)        if
form.is_valid():            form.save()           return redirect('profile_view')
else:          return     render(request,  editTemplate,     {'form':     form,
 'error':
"Something went wrong.", "profile" : profile})
else:
        form = ProfileForm(instance=profile)
    return render(request, editTemplate, {'form': form, 'profile':profile})

@login_required def profile_view(request):
if hasattr(request.user, 'studentprofile'):
profile = request.user.studentprofile
elif hasattr(request.user, 'teacherprofile'):
profile =
request.user.teacherprofile     else:
return redirect("login_view")

    return render(request, 'profile.html', {'profile': profile})



def login_view(request):      if
request.user.is_authenticated:
        if       hasattr(request.user,  'studentprofile')  or
hasattr(request.user, 'teacherprofile'):            return
redirect('profile_view')         else:             return
redirect('logout_view')
if request.method == 'POST':          form =
AuthenticationForm(data=request.POST)
if form.is_valid():              username = form.cleaned_data.get('username')
password = form.cleaned_data.get('password')              user    =
authenticate(request,  username=username, password=password)
if user is not None:                login(request, user)                next_url =
request.GET.get('next')                if next_url:                    return
redirect(next_url)                else:                    return
redirect('dashboard_view')            else:                return render(request,
'login.html', {'form': form, 'error': 'Invalid
username or password'})     else:
form = AuthenticationForm()
    return render(request, 'login.html', {'form': form})

def logout_view(request):
    logout(request)      return
redirect('login_view')
```

Dashboard:

Forms.py:

```
from django import forms
from .models import Internship, Application

class InternshipForm(forms.ModelForm):
class Meta:          model = Internship
        fields = ['title', 'description', 'salary', 'is_active']

class
ApplicationForm(forms.ModelForm):     class
Meta:         model = Application         fields =
[]
```

models.py:

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

class        Internship(models.Model):                              title        =
models.CharField(max_length=100)     description = models.TextField()
salary        =       models.DecimalField(max_digits=10,
decimal_places=2)     teacher = models.ForeignKey(User,
on_delete=models.CASCADE)  # Teacher who posted the internship
is_active = models.BooleanField(default=True)


    class Meta:
        permissions = [
            ("can_post_internship", "Can post internship"),
            ("can_edit_internship", "Can edit internships"),
            ("can_delete_internship", "Can delete internships"),
            ("can_apply_to_internship", "Can apply to internships"),
            ("can_view_applied_internships", "Can view all applied
internships"),
            ("can_view_internship_applications", "Can view all applications
to internships"),
        ]
    def __str__(self):
return self.title
 class
Application(models.Model):      internship     =
models.ForeignKey(Internship, on_delete=models.CASCADE)
student = models.ForeignKey(User, on_delete=models.CASCADE)  #
```

```
Student who applied for the internship      applied_at =
models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ('internship', 'student')  # Ensure a student can
apply to an internship only once

    def __str__(self):
        return f"{self.student.username} applied for {self.internship.title}"
```

urls.py:

```
from django.urls import path
from . import views

urlpatterns = [      path(", views.dashboard_view,
name='dashboard_view'),

    path('internship/post', views.post_internship, name='post_internship'),
path('internship/edit/<int:internship_id>/',     views.edit_internship,
name='edit_internship'),
    path('internship/delete/<int:internship_id>/',
views.delete_internship, name='delete_internship'),
    path('internship/apply/<int:internship_id>/',
 views.apply_internship, name='apply_internship'),
    path('internship/applied/',         views.applied_internships,
name='applied_internships'),
path('internship/applications/<int:internship_id>/',
views.internship_applications, name='internship_applications')
]
```

Views.py:

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required,
permission_required

from .models import Internship, Application from
.forms import InternshipForm, ApplicationForm


@login_required def
dashboard_view(request):     if
hasattr(request.user, 'studentprofile'):
internships = Internship.objects.all()
        applied_internships = [ application.internship for application in
Application.objects.filter(student=request.user)]        applied_ids =
[internship.id for internship in internships if
internship in applied_internships]
        unavailable_ids = [internship.id for internship in internships if
internship.is_active == False]
```

```
        return  render(request,  'student_dashboard.html',  {'internships':
internships, 'applied_ids':applied_ids, 'unavailable_ids':unavailable_ids})

    elif hasattr(request.user, 'teacherprofile'):        internships =
Internship.objects.all()                        return     render(request,
'teacher_dashboard.html',        {'internships':        internships,
"user":request.user})    else:        return redirect('logout_view')

@login_required
@permission_required('dashboard.can_post_internshi
p', raise_exception=True) def
post_internship(request):     if request.method ==
'POST':          form =
InternshipForm(request.POST)
if form.is_valid():
internship =
form.save(commit=False)
internship.teacher = request.user
internship.save()            return
redirect('dashboard_view')      else:
form = InternshipForm()
    return render(request, 'post_internship.html', {'form': form})

@login_required
@permission_required('dashboard.can_edit_internshi p',
raise_exception=True) def edit_internship(request, internship_id):
internship = get_object_or_404(Internship, id=internship_id)     if
request.user.is_authenticated and request.user ==
internship.teacher:         if request.method == 'POST':              form
= InternshipForm(request.POST, instance=internship)
if form.is_valid():
form.save()                  return
redirect('dashboard_view')         else:              form =
InternshipForm(instance=internship)         return
render(request, 'edit_internship.html', {'form': form, 'internship':
internship})
else:          return
redirect('dashboard_view')

@login_required
@permission_required('dashboard.can_delete_internshi p',
raise_exception=True) def delete_internship(request,
internship_id):      internship = get_object_or_404(Internship,
id=internship_id)     if request.user.is_authenticated and
request.user == internship.teacher:         if request.method ==
'POST':
            internship.delete()
            return redirect('dashboard_view')         else:            return
 render(request,  'delete_internship.html',      {'internship':  internship})
else:          return
redirect('dashboard_view')

@login_required
@permission_required('dashboard.can_apply_to_internshi p',
raise_exception=True) def apply_internship(request,
internship_id):      internship = get_object_or_404(Internship,
id=internship_id)
```

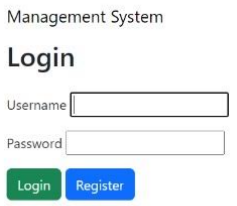
```
if request.method == 'POST':          form =
ApplicationForm(request.POST)
if form.is_valid():              application =
form.save(commit=False)
application.internship = internship             application.student
= request.user
            application.save()

            internship.is_active = False
internship.save()

            return redirect('dashboard_view')
else:          form = ApplicationForm()     return
render(request, 'apply_internship.html', {'form': form,
'internship':  internship})

@login_required
@permission_required('dashboard.can_view_applied_internshi ps',
raise_exception=True) def applied_internships(request):     if
request.user.is_authenticated:          applications =
Application.objects.filter(student=request.user)          return
render(request,  'applied_internships.html',  {'applications':
applications})
else:          return
redirect('dashboard_view')

@login_required
@permission_required('dashboard.can_view_internship_applicatio ns',
raise_exception=True) def internship_applications(request,
internship_id):      applications =
Application.objects.filter(internship_id=internship_id)
internship = Internship.objects.get(id=internship_id)

    return render(request, 'internship_applications.html', {'applications':
applications, 'internship':internship})
```

PracticeSchoolManagementSystem:

Urls.py:

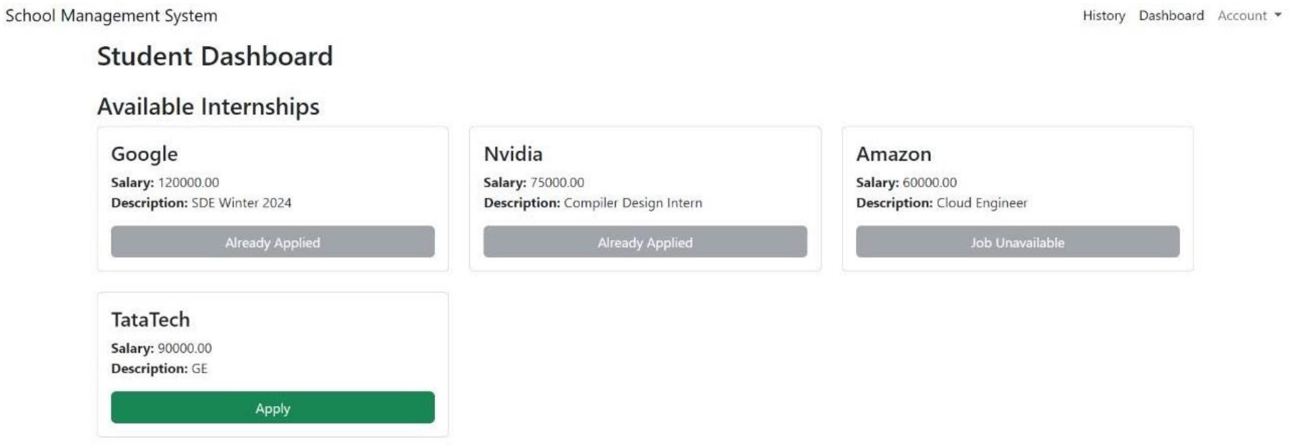
```
from django.contrib import admin
from django.urls import path, include

from django.shortcuts import redirect

urlpatterns = [
    path('', lambda req: redirect('login_view')), # Redirect to login on no
path     path('admin/', admin.site.urls),     path('account/',
include('account.urls')),
    path('accounts/', include('account.urls')), # For compatibility with
login_required
    path('dashboard/', include('dashboard.urls')),
]
```

SCREENSHOTS:



Management System

Login



School Management System

Student Dashboard

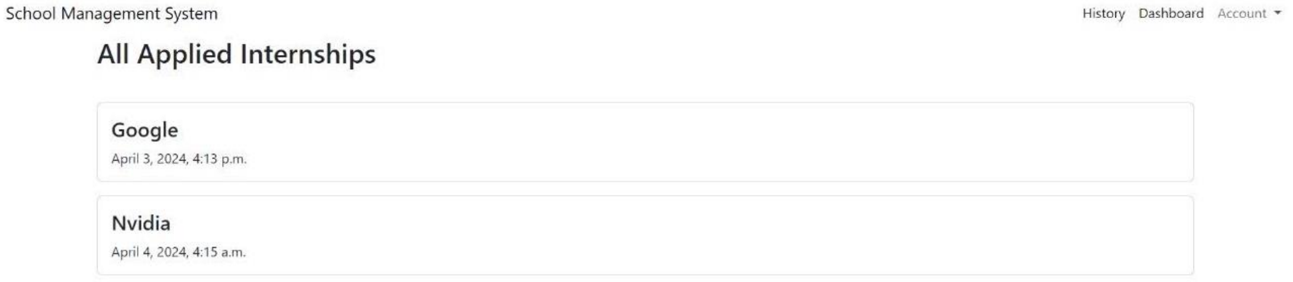
Available Internships

Google
Salary: 120000.00
Description: SDE Winter 2024

Nvidia
Salary: 75000.00
Description: Compiler Design Intern

Amazon
Salary: 60000.00
Description: Cloud Engineer

TataTech
Salary: 90000.00
Description: GE

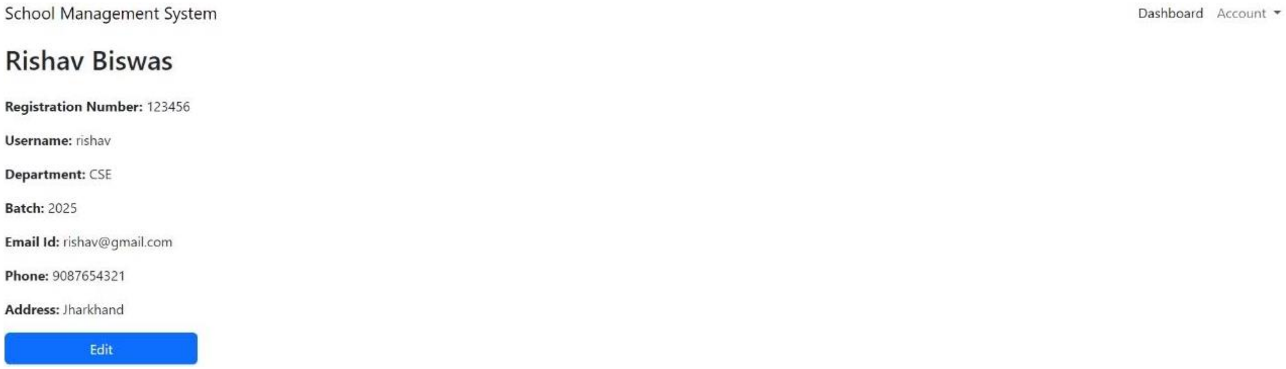


School Management System

All Applied Internships

Google
April 3, 2024, 4:13 p.m.

Nvidia
April 4, 2024, 4:15 a.m.



Rishav Biswas

Registration Number: 123456

Username: rishav

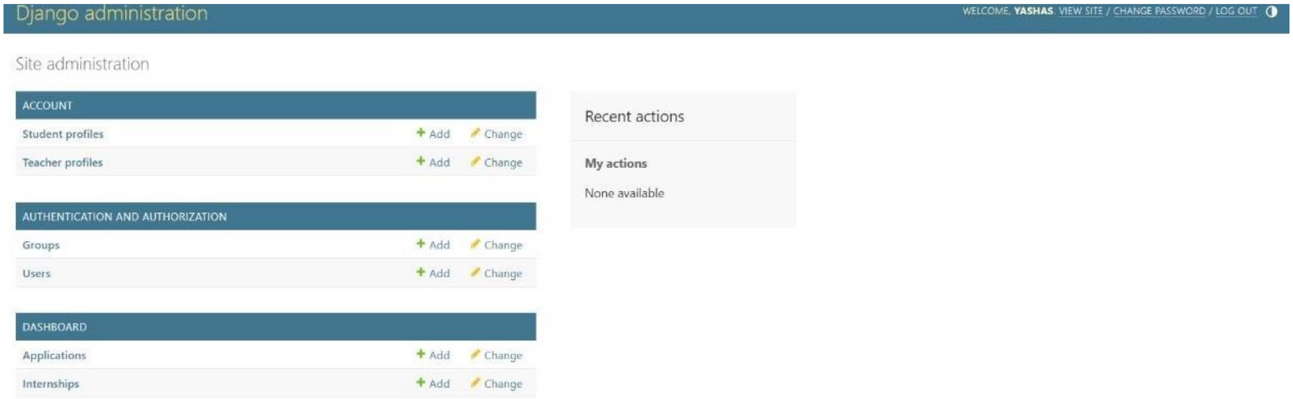
Department: CSE

Batch: 2025

Email Id: rishav@gmail.com

Phone: 9087654321

Address: Jharkhand



Site administration

ACCOUNT

Student profiles + Add ✎ Change

Teacher profiles + Add ✎ Change

AUTHENTICATION AND AUTHORIZATION

Groups + Add ✎ Change

Users + Add ✎ Change

DASHBOARD

Applications + Add ✎ Change

Internships + Add ✎ Change

Recent actions

My actions

None available

CHAPTER 5: CONCLUSION

The Practice School Management System project represents a significant step towards modernizing administrative practices within educational institutions and fostering a culture of innovation and efficiency. Throughout the course of this project, a comprehensive platform has been developed to address the administrative challenges faced by students, teachers, and administrators in managing projects and internships effectively.

The objectives set forth at the beginning of the project have been successfully achieved through a systematic approach to development and implementation. The SMS platform provides a user-friendly interface for students to explore, apply for, and track their involvement in various projects and internships. Similarly, teachers have been empowered to manage and update available opportunities, fostering a dynamic learning environment that bridges theoretical knowledge with practical application.

By streamlining communication channels, enhancing transparency, and optimizing resource allocation, the SMS project has laid the groundwork for a more collaborative and efficient educational ecosystem. The scalability and flexibility of the SMS platform ensures its adaptability to the evolving needs of educational institutions, allowing for seamless integration of additional features and functionalities in the future.

Furthermore, the successful deployment of the SMS platform underscores the importance of stakeholder engagement and continuous feedback throughout the development process. By prioritizing user satisfaction and addressing issues in realtime, the SMS project has demonstrated a commitment to delivering a solution that meets the diverse needs of its users.

In conclusion, the Practice School Management System project represents a testament to the power of technology in transforming educational management practices and fostering academic excellence. As educational institutions continue to evolve and embrace digital innovation, the SMS project stands as a beacon of progress, empowering students, teachers, and administrators to thrive in an everchanging educational landscape.

CHAPTER 6: REFERENCE

The resources that have been used in the creation of this project are:

- Brown, C., & Williams, D. (Year). "Streamlining Administrative Processes in Educational Institutions." International Conference on Information Systems, 123-135.

- Anderson, R. (Year). "Agile Development Methodologies: Principles and Practices." Communications of the ACM, 55(6), 71-78.

- MongoDB Documentation. Retrieved from https://docs.mongodb.com/

- Git Documentation. Retrieved from https://git-scm.com/doc

- Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide), 6th Edition.

- IEEE Computer Society. IEEE Standard for Software and System Test Documentation.