API_V1

ADMIN.PY

```
from django.contrib import admin
            from api_v1.models import *


            # Register your models here.
            admin.site.register(student)
            admin.site.register(subjects)
            admin.site.register(attendance)
            admin.site.register(studentSubject)
```

APPS.PY

```
from django.apps import AppConfig




            class ApiV1Config(AppConfig):
            name = 'api_v1'


            def ready(self):
            import api_v1.signals
```

MIGRATION

MODELS.PY

```
from django.db import modelsfrom uuid import uuid4
def generateUUID(): return str(uuid4())
class record(models.Model): recordAdded = models.DateTimeField(auto_now_add=True)
recordModified = models.DateTimeField(auto_now=True)
class Meta: abstract = True
class student(record): studentId = models.CharField(primary_key=True,
default=generateUUID, editable=False,max_length=200) firstName =
models.CharField(max_length=200, blank=False, null=False) lastName =
models.CharField(max_length=200, blank=False, null=False) address =
models.CharField(max_length=200, blank=False, null=False) email =
models.CharField(max_length=200, blank=False, null=False) mobileNo =
models.CharField(max_length=15, blank=False, null=False) department =
models.CharField(max_length=100, blank=False, null=False)
def __str__(self): return str(self.studentId)

class subjects(record): subjectCode = models.AutoField(primary_key=True) title =
models.CharField(max_length=200, blank=False, null=False) #department =
models.ForeignKey(student,to_field="department",on_delete=models.CASCADE)
def __str__(self): return str(self.subjectCode)
class studentSubject(record): studentId =
models.ForeignKey(student,to_field="studentId",on_delete=models.CASCADE)
subjectCode =
models.ForeignKey(subjects,to_field="subjectCode",on_delete=models.CASCADE)
def __str__(self): return str(self.studentId)
class attendance(record): studentId =
models.ForeignKey(student,to_field="studentId",on_delete=models.CASCADE)
subjectCode = models.CharField(max_length=200, blank=False, null=False) lectureDate
= models.CharField(max_length=50, blank=False, null=False) status =
```

```
models.CharField(max_length=20, blank=False, null=False)
def __str__(self): return str(self.studentId)

SEND_MAIL.PY
import boto3
            from botocore.exceptions import ClientError


            def sendEmail(studentId,firstName,lastName):


            SENDER = "nus.jenkins@gmail.com"
            RECIPIENT_LIST =
["nus.jenkins@gmail.com","grishi2020@gmail.com","zaheernew@gmail.com"]


            SUBJECT = "Student Registered for Cloud Attendance System"


            # The email body for recipients with non-HTML email clients.
            BODY_TEXT = ("Hi "+firstName+" "+lastName+"\n\n"
            "You have been registred for attendance system. Please refer more
details,\n"+
            "Student ID : "+studentId+"\n"+"Password : Student@123\n\nBest
Regards,\nAdminstration"
            )


            # The character encoding for the email.
            CHARSET = "UTF-8"


            # Create a new SES resource and specify a region.
            client = boto3.client('ses',region_name='ap-southeast-1')


            # Try to send the email.
            try:
            #Provide the contents of the email.
            response = client.send_email(
            Destination={
            'ToAddresses': RECIPIENT_LIST,
            },
            Message={
            'Body': {
            'Text': {
            'Charset': CHARSET,
            'Data': BODY_TEXT,
            },
            },
            'Subject': {
            'Charset': CHARSET,
            'Data': SUBJECT,
            },
            },
            Source=SENDER,


            )
```

```
            # Display an error if something goes wrong.
            except ClientError as e:
            print(e.response['Error']['Message'])
            else:
            print("Email sent! Message ID:"),ALIZERS
            print(response['MessageId'])
SERIALIZER.PY
from rest_framework import serializers
            from .models import *


            class studentSerializer(serializers.ModelSerializer):
            class Meta:
            model = student
            fields= "__all__"


            class subjectSerializer(serializers.ModelSerializer):
            class Meta:
            model = subjects
            fields= "__all__"


            class attendanceSerializer(serializers.ModelSerializer):
            class Meta:
            model = attendance
            fields= ('id','studentId','subjectCode','status','lectureDate')




            class studentSubjectSerializer(serializers.ModelSerializer):
            class Meta:
            model = studentSubject
            fields= "__all__"
SIGNALS.PY
from django.contrib.auth.models import User
            from django.contrib.auth.models import Group
            from django.db.models.signals import post_save
            from django.dispatch import receiver
            from django.db.backends.signals import connection_created
            from .models import student
            from .send_mail import sendEmail


            @receiver(post_save,sender=student)
            def new_user_registration(sender, instance, created,**kwargs):
            if created:
            print("new user is created, therefore going to register it in Auth
Model and send mail via AWS")


            #Filter student object to get more attributes
            newStudent=student.objects.get(studentId=instance)


            #Add the user to Auth.models.User Table
```

```
            user =
User.objects.create_user(username=instance,email=newStudent.email,password='Student
@123',
            first_name=newStudent.firstName,last_name=newStudent.lastName)



            #Add the user to new group
            Studentgroup = Group.objects.get(name='Student')
            Studentgroup.user_set.add(user)


            #Send mail to user via AWS SES

      sendEmail(newStudent.studentId,newStudent.firstName,newStudent.lastName)


TEST.PY
from django.urls import path
            from api_v1 import views
            from django.conf.urls import url, include


            urlpatterns = [

      path("v1/student/<slug:studentId>",views.manageStudentRecord.as_view()),
            path("v1/student",views.CreateStudent.as_view()),



      #path("v1/subject/<slug:subjectCode>",views.manageStudentRecord.as_view()),
            path("v1/attendance/<int:id>",views.manageAttendanceRecord.as_view()),
            path("v1/attendance",views.CreateAttendance.as_view()),




            path("home",views.homePage),
            path("student_insert",views.student_insert),
            path("student_delete",views.student_delete),
            path("student_update",views.student_update),
            path("student_viewsingle",views.student_viewsingle),
            path("student_viewall",views.student_viewall),
            path("attendance_insert",views.attendance_insert),
            path("attendance_delete",views.attendance_delete),
            path("attendance_update",views.attendance_update),
            path("attendance_viewsingle",views.attendance_viewsingle),
            path("attendance_viewall",views.attendance_viewall),
            path("subjects_viewall",views.subjects_viewall),
            path("viewPersonal",views.viewPersonal),
            ]
URLS.PY
from django.urls import path
            from api_v1 import views
            from django.conf.urls import url, include


            urlpatterns = [
```

```
      path("v1/student/<slug:studentId>",views.manageStudentRecord.as_view()),
            path("v1/student",views.CreateStudent.as_view()),



      #path("v1/subject/<slug:subjectCode>",views.manageStudentRecord.as_view()),
            path("v1/attendance/<int:id>",views.manageAttendanceRecord.as_view()),
            path("v1/attendance",views.CreateAttendance.as_view()),




            path("home",views.homePage),
            path("student_insert",views.student_insert),
            path("student_delete",views.student_delete),
            path("student_update",views.student_update),
            path("student_viewsingle",views.student_viewsingle),
            path("student_viewall",views.student_viewall),
            path("attendance_insert",views.attendance_insert),
            path("attendance_delete",views.attendance_delete),
            path("attendance_update",views.attendance_update),
            path("attendance_viewsingle",views.attendance_viewsingle),
            path("attendance_viewall",views.attendance_viewall),
            path("subjects_viewall",views.subjects_viewall),
            path("viewPersonal",views.viewPersonal),.
            ]
VIEWS.PY
from django.shortcuts import render
            from rest_framework.generics import
ListAPIView,RetrieveUpdateDestroyAPIView, ListCreateAPIView, ListAPIView
            from .models import *
            from .serializers import *
            from django import http
            from django.contrib.auth.decorators import login_required




            # Create your views here.
            class manageStudentRecord(RetrieveUpdateDestroyAPIView):
            '''
            This view is used to manage records related to a Student
            '''


            serializer_class = studentSerializer
            lookup_url_kwarg = 'studentId'
            queryset = student.objects.all()


            class manageAttendanceRecord(RetrieveUpdateDestroyAPIView):
            '''
            This view is used to manage records related to a Attendance
            '''


            serializer_class = attendanceSerializer
            lookup_url_kwarg = 'id'
            queryset = attendance.objects.all()
```

```
            class manageSubjectRecord(RetrieveUpdateDestroyAPIView):
            '''
            This view is used to manage records related to a Subject
            '''


            serializer_class = subjectSerializer
            lookup_url_kwarg = 'subjectCode'
            queryset = subjects.objects.all()




            class CreateStudent(ListCreateAPIView):
            '''
            This view is used to create a record and retrieve all records to
student
            '''


            serializer_class = studentSerializer
            queryset = student.objects.all()


            class CreateAttendance(ListCreateAPIView):
            '''
            This view is used to create a record and retrieve all records related
to attandance
            '''


            serializer_class = attendanceSerializer
            queryset = attendance.objects.all()


            class CreateSubject(ListCreateAPIView):
            '''
            This view is used to create a record and retrieve all records related
to subjects
            '''


            serializer_class = subjectSerializer
            queryset = subjects.objects.all()


            class CreateStudentSubject(ListCreateAPIView):
            '''
            This view is used to create a record and retrieve all records related
to StudentSubjects
            '''


            serializer_class = studentSubjectSerializer
            queryset = studentSubject.objects.all()
```

```
@login_required
def homePage(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "index.htm")


if group.name=="Student":
print("User ID : "+str(request.user))
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)
else:
return render(request, "index.htm")


'''
Student Record Management
'''


@login_required
def student_insert(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "student_insert.htm")


if group.name=="Student":
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)
else:
return render(request, "student_insert.htm")


@login_required
def student_update(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "student_update.htm")


if group.name=="Student":
return render(request, "my_record.htm")
else:
return render(request, "student_update.htm")
```

```
@login_required
def student_delete(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "student_delete.htm")


if group.name=="Student":
return render(request, "my_record.htm")
else:
return render(request, "student_delete.htm")


@login_required
def student_viewsingle(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "student_viewsingle.htm")


if group.name=="Student":
return render(request, "my_record.htm")
else:
return render(request, "student_viewsingle.htm")


@login_required
def student_viewall(request):


try:
group = request.user.groups.get(name="Student")
except:
personalRecords = student.objects.all()
context = {"allrecord": personalRecords}
return render(request, "student_viewall.htm",context)


if group.name=="Student":
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)
else:
personalRecords = student.objects.all()
context = {"allrecord": personalRecords}
return render(request, "student_viewall.htm",context)



'''
Attendance
```

```
'''


@login_required
def attendance_insert(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "attendance_insert.htm")


if group.name=="Student":
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)


@login_required
def attendance_update(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "attendance_update.htm")


if group.name=="Student":
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)


@login_required
def attendance_delete(request):


try:
group = request.user.groups.get(name="Student")
except:
return render(request, "attendance_delete.htm")


if group.name=="Student":
personalData = student.objects.get(studentId=request.user)
context = {"student": personalData}
return render(request, "my_record.htm",context)


@login_required
def attendance_viewsingle(request):


try:
group = request.user.groups.get(name="Student")
except:
```

```
            try:
            student=(request.POST['studentIdValue'])
            personalData = attendance.objects.filter(studentId=student)
            context = {"allrecord": personalData}
            return render(request, "attendance_viewsingle.htm",context)


            except:
            return render(request, "attendance_viewsingle.htm")


            if group.name=="Student":
            personalData = student.objects.get(studentId=request.user)
            context = {"student": personalData}
            return render(request, "my_record.htm",context)


            @login_required
            def attendance_viewall(request):


            try:
            group = request.user.groups.get(name="Student")
            except:
            attendanceRecords = attendance.objects.all()
            context = {"allrecord": attendanceRecords}
            return render(request, "attendance_viewall.htm",context)


            if group.name=="Student":
            personalData = student.objects.get(studentId=request.user)
            context = {"student": personalData}
            return render(request, "my_record.htm",context)


            '''
            Personal Student Record
            '''


            @login_required
            def viewPersonal(request):


            try:
            group = request.user.groups.get(name="Student")
            except:
            return render(request, "index.htm")


            if group.name=="Student":
            attendanceRecords =
attendance.objects.filter(studentId=str(request.user))
            context = {"allrecord": attendanceRecords}
            return render(request, "my_recordall.htm",context)
            else:
            return render(request, "index.htm")
```

```
            '''
            Subjects
            '''


            @login_required
            def subjects_viewall(request):


            try:
            group = request.user.groups.get(name="Student")
            except:
            return render(request, "subjects_viewall.htm")


            if group.name=="Student":
            personalData = student.objects.get(studentId=request.user)
            context = {"student": personalData}
            return render(request, "my_record.htm",context)
```

MANAGE.PY

```
#!/usr/bin/env python
            import os
            import sys


            if __name__ == '__main__':
            os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'Attendance_System.settings')
            try:
            from django.core.management import execute_from_command_line
            except ImportError as exc:
            raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
            ) from exc
            execute_from_command_line(sys.argv)
```