**SRS**

There are 30 Subjects in 3 Yrs. Law Programme and 50 Subjects in 5 Yrs. Law Programme. 30 Subjects in Law are common for both the programmes. However, there were 20 non-law subjects which are spared over the first three years of BALLB programme.

We need the programming of attendance for I-Yr. BALLB and 1st Yr. LLB in two divisions i.e. *Section – A and Section – B.*

From II – BALLB to V Yr. BALLB one section for 5 Yrs. Programme and 2nd Yr. LLB to 3rd Yr. LLB. one section for 3 Yrs. Programme.

Every day 5 lectures for each course will be conducted from morning 08:30 AM to 02:00 PM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3 / 5 Yrs. LLB Programme Lecture Timings 1st Lec.** | **2nd Lec.** | **3rd Lec.** | **4th Lec.** | **5th Lec** |
| **08:30 AM** | **09:30 AM** | **10:30 AM** | **12:00 Noon** | **01:00 PM** |
| 6x1= 6 (5Yrs.)  4x1 = 4 (3Yrs.) | 6x1 = 6  4x1 = 4 | 6x1 = 6  4x1 = 4 | 6x1 = 6  4x1 = 4 | 6x1 = 6  4x1 = 4 |

We request you to design a programme for online attendance of the respective course students with a default of ABSENT and present will be filled while filling attendance.

The consolidated attendance of all the five subjects must be displayed as per the roll numbers of the respective class students Month Wise/ Day Wise / Week Wise be displayed by click of student roll number.

Example

G240001 - P P A P A

G240002 - A P P A A

G240003 - P A P P P

NG240012 - A P A P A

NG240013 - P P A A A

We are herewith enclosing the subjects of 3 and 5 Yrs. Law Courses and the sample attendance we are taking Lecture Wise (hourly) for your kind perusal and favorable consideration. It should also reflect the number of students present class / hour wise and consolidated statement of the same.

We solicit your cooperation in providing digital attendance as desired by Bar Counsil of India, New Delhi to full fill the mandatory obligation on the legal education centers.

**LLB – 3 Yr. Programme**

|  |  |
| --- | --- |
| **I – Semester** | **II – Semester** |
| Law of Contract I | Constitutional Law II |
| Constitutional Law I | Torts, Motor Vehicle Act and  Consumer Protection Act |
| Jurisprudence | Family Law II (Succession) |
| Family Law I (Family Relations) | Law of Contract II (Optional  Legal) |
| Banking Law Including  Negotiable Instruments Act | Social Research Methods  (Research methodology) |

|  |  |
| --- | --- |
| **III – Semester** | **IV – Semester** |
| Bhartiya Nyaya Sanhita | Bharatiya Nagarik Suraksha Sanhita |
| Property Law and Easement Act | Company Law |
| Labour Law I | Labour Law II |
| Administrative Law | Interpretation of Statutes  (Optional Legal) |
| Professional Ethics | ADR (Clinical Legal Edn.) |

|  |  |
| --- | --- |
| **V – Semester** | **VI – Semester** |
| Bharatiya Sakshya Adhiniyam | Civil Procedure Code and  Limitation Act |
| Principles of Taxation | Public International Law |
| Environmental Law | Intellectual Property Law |
| Land Laws including Tenure and  Tenancy Law | Insurance Law (Optional Legal) |
| Drafting, Pleading and Conveyance | Moot Court (Clinical Legal Edn.) |

**BALLB – 5 Yr. Programme**

|  |  |
| --- | --- |
| **I – Semester** | **II – Semester** |
| Principles of Political Science - I | Political Theory - II |
| Micro Economics - I | Macro Economics - II |
| Principles of Sociology - I | Theories of Sociology - II |
| General English | Local Language / Foreign Language  (Optional) |
| Legal Development | Constitutional Development |

|  |  |
| --- | --- |
| **III – Semester** | **IV – Semester** |
| Law and Literature | Legal Language and Legal Writing |
| Foundations of Political Obligations - III | International relations and obligations IV |
| Indian Economy - III | Economic Policies IV |
| Urban, Rural & Tribal Sociology of India - III | Contemporary issues of Sociology IV |
| Constitutional Law I | Constitutional Law II |

|  |  |
| --- | --- |
| **V – Semester** | **VI – Semester** |
| Local Self Government - V | Principles of Public Administration - VI |
| Jurisprudence | Torts, Motor Vehicles Act and Consumer Protection Act |
| Family Law I (Family Relations) | Family Law II (Succession) |
| Law of Contract I | Law of Contract II (Optional Legal) |
| Banking Law Including Negotiable Instruments Act | Social Research Methods  (Research methodology) |

|  |  |
| --- | --- |
| **VII – Semester** | **VIII – Semester** |
| Bhartiya Nyaya Sanhita | Bharatiya Nagarik Suraksha Sanhita |
| Property Law and Easement Act | Company Law |
| Labour Law I | Labour Law II |
| Administrative Law | Interpretation of Statutes  (Optional Legal) |
| Professional Ethics | ADR (Clinical Legal Edn.) |

|  |  |
| --- | --- |
| **IX – Semester** | **X – Semester** |
| Bharatiya Sakshya Adhiniyam | Civil Procedure Code and Limitation Act |
| Principles of Taxation | Public International Law |
| Environmental Law | Intellectual Property Law |
| Land Laws including Tenure and  Tenancy Law | Insurance Law (Optional Legal) |
| Drafting, Pleading and Conveyance | Moot Court (Clinical Legal Edn.) |

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

This was the SRS of our project now we’ve come so far here are the more details of our project like database records , codebase and problems and the goal is the complete this project as soon as possible ( sooner ) . we are building this project in Django ( DRF ) and React.js , MySQL , so here are the code so keep in mind the code so that we’ll build this project quickly without wasting time . we’ve named our project as program\_attendance and our app name is “app”

>>>>> app.models.py -> from django.db import models

from django.contrib.auth.models import User

from django.core.exceptions import ValidationError

from django.core.validators import RegexValidator

# Validate lecture time slots

LECTURE\_SLOTS = [

    ('08:30:00', '08:30 AM - 09:30 AM'),

    ('09:30:00', '09:30 AM - 10:30 AM'),

    ('10:30:00', '10:30 AM - 11:30 AM'),

    ('12:00:00', '12:00 PM - 01:00 PM'),

    ('13:00:00', '01:00 PM - 02:00 PM'),

]

class Program(models.Model):

    name = models.CharField(max\_length=50, unique=True)  # e.g., "BALLB 5 Yr", "LLB 3 Yr"

    duration\_years = models.PositiveIntegerField()  # 3 or 5

    def \_\_str\_\_(self):

        return self.name

    class Meta:

        verbose\_name = "Program"

        verbose\_name\_plural = "Programs"

class Section(models.Model):

    program = models.ForeignKey(Program, on\_delete=models.CASCADE, related\_name="sections")

    name = models.CharField(max\_length=10)  # e.g., "Section A", "Section B"

    year = models.PositiveIntegerField()  # 1, 2, 3, 4, 5

    def \_\_str\_\_(self):

        return f"{self.program} - Year {self.year} - {self.name}"

    class Meta:

        unique\_together = ('program', 'year', 'name')

        verbose\_name = "Section"

        verbose\_name\_plural = "Sections"

class Subject(models.Model):

    name = models.CharField(max\_length=100, unique=True)  # e.g., "Law of Contract I"

    is\_law\_subject = models.BooleanField(default=True)  # True for law, False for non-law

    semester = models.PositiveIntegerField()  # 1 to 10 (depending on program)

    def \_\_str\_\_(self):

        return self.name

    class Meta:

        verbose\_name = "Subject"

        verbose\_name\_plural = "Subjects"

class Teacher(models.Model):

    user = models.OneToOneField(User, on\_delete=models.CASCADE)  # Authentication

    first\_name = models.CharField(max\_length=100)

    last\_name = models.CharField(max\_length=100)

    email = models.EmailField(unique=True)

    phone = models.CharField(max\_length=15, null=True, blank=True)

    is\_admin = models.BooleanField(default = False)

    def \_\_str\_\_(self):

        return f"{self.first\_name} {self.last\_name}"

    class Meta:

        verbose\_name = "Teacher"

        verbose\_name\_plural = "Teachers"

class Student(models.Model):

    roll\_number = models.CharField(

        max\_length=10,

        unique=True,

        validators=[RegexValidator(r'^(G|NG)24\d{4}$', 'Roll number must be G24xxxx or NG24xxxx')]

    )  # e.g., G240001, NG240012

    first\_name = models.CharField(max\_length=50)

    last\_name = models.CharField(max\_length=50)

    email = models.EmailField(unique=True)

    phone = models.CharField(max\_length = 15)

    section = models.ForeignKey(Section, on\_delete=models.CASCADE, related\_name="students")

    subjects = models.ManyToManyField(Subject, related\_name="students", blank=True)

    def \_\_str\_\_(self):

        return f"{self.roll\_number} - {self.first\_name} {self.last\_name}"

    class Meta:

        verbose\_name = "Student"

        verbose\_name\_plural = "Students"

class Timetable(models.Model):

    DAY\_CHOICES = [

        ('Monday', 'Monday'),

        ('Tuesday', 'Tuesday'),

        ('Wednesday', 'Wednesday'),

        ('Thursday', 'Thursday'),

        ('Friday', 'Friday'),

        ('Saturday', 'Saturday'),

    ]

    LECTURE\_SLOTS = [

    ('08:30:00', '08:30 AM - 09:30 AM'),

    ('09:30:00', '09:30 AM - 10:30 AM'),

    ('10:30:00', '10:30 AM - 11:30 AM'),

    ('12:00:00', '12:00 PM - 01:00 PM'),

    ('13:00:00', '01:00 PM - 02:00 PM'),

    ]

    section = models.ForeignKey(Section, on\_delete=models.CASCADE, related\_name="timetable")

    subject = models.ForeignKey(Subject, on\_delete=models.CASCADE, related\_name="timetable")

    teacher = models.ForeignKey(Teacher, on\_delete=models.SET\_NULL, null=True, related\_name="timetable")

    day\_of\_week = models.CharField(max\_length=9, choices=DAY\_CHOICES)

    start\_time = models.TimeField(choices=LECTURE\_SLOTS)  # Fixed lecture slots

    semester\_start\_date = models.DateField()

    semester\_end\_date = models.DateField()

    def \_\_str\_\_(self):

        return f"{self.section} - {self.subject} ({self.day\_of\_week} {self.start\_time})"

    class Meta:

        unique\_together = ('section', 'subject', 'day\_of\_week', 'start\_time')

        verbose\_name = "Timetable"

        verbose\_name\_plural = "Timetables"

class Session(models.Model):

    STATUS\_CHOICES = [

        ('Scheduled', 'Scheduled'),

        ('Completed', 'Completed'),

        ('Cancelled', 'Cancelled'),

    ]

    timetable = models.ForeignKey(Timetable, on\_delete=models.CASCADE, related\_name="sessions")

    date = models.DateField()

    status = models.CharField(max\_length=10, choices=STATUS\_CHOICES, default='Scheduled')

    def \_\_str\_\_(self):

        return f"{self.timetable} on {self.date} ({self.status})"

    class Meta:

        unique\_together = ('timetable', 'date')

        verbose\_name = "Session"

        verbose\_name\_plural = "Sessions"

class Attendance(models.Model):

    STATUS\_CHOICES = [

        ('Present', 'Present'),

        ('Absent', 'Absent'),

    ]

    student = models.ForeignKey(Student, on\_delete=models.CASCADE, related\_name="attendance")

    session = models.ForeignKey(Session, on\_delete=models.CASCADE, related\_name="attendance")

    status = models.CharField(max\_length=7, choices=STATUS\_CHOICES, default='Absent')

    timestamp = models.DateTimeField(auto\_now=True)

    recorded\_by = models.ForeignKey(Teacher, on\_delete=models.SET\_NULL, null=True, related\_name="attendance\_records")

    def \_\_str\_\_(self):

        return f"{self.student} - {self.session} - {self.status}"

    class Meta:

        unique\_together = ('student', 'session')

        verbose\_name = "Attendance"

        verbose\_name\_plural = "Attendance"

class CalendarException(models.Model):

    date = models.DateField(unique=True)

    description = models.CharField(max\_length=100)

    def \_\_str\_\_(self):

        return f"{self.date} - {self.description}"

    class Meta:

        verbose\_name = "Calendar Exception"

        verbose\_name\_plural = "Calendar Exceptions"

records -> app\_program table records ->

|  |  |  |
| --- | --- | --- |
| 1 | BALLB 5 Yr | 5 |
| 2 | LLB 3 Yr | 3 |
| 5 | BALLB  5  Yr | 5 |

App\_section table ->

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Section A | 1 | 1 |
| 2 | Section B | 1 | 1 |
| 3 | Section A | 2 | 1 |
| 7 | Section A | 3 | 1 |
| 8 | Section A | 4 | 1 |
| 9 | Section A | 5 | 1 |
| 4 | Section A | 1 | 2 |
| 6 | Section B | 1 | 2 |
| 10 | Section A | 2 | 2 |
| 11 | Section A | 3 | 2 |
| 5 | Section A | 1 | 5 |
|  |  |  |  |

App\_subject ->

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | Political Science I | 0 | 1 |
| 31 | Microeconomics I | 0 | 1 |
| 32 | Sociology I | 0 | 1 |
| 33 | General English | 0 | 1 |
| 34 | Local Language/Foreign Language | 0 | 1 |
| 35 | Legal Development | 0 | 1 |
| 36 | Political Theory II | 0 | 2 |
| 37 | Macroeconomics II | 0 | 2 |
| 38 | Sociology II | 0 | 2 |
| 39 | Constitutional Development | 0 | 2 |
| 40 | Law and Literature | 0 | 3 |
| 41 | Foundations of Political Obligations | 0 | 3 |
| 42 | Indian Economy | 0 | 3 |
| 43 | Urban, Rural & Tribal Sociology of India | 0 | 3 |
| 44 | Constitutional Law I | 1 | 3 |
| 45 | Legal Language and Legal Writing | 0 | 4 |
| 46 | International Relations and Obligations | 0 | 4 |
| 47 | Economic Policies | 0 | 4 |
| 48 | Contemporary Issues of Sociology | 0 | 4 |
| 49 | Constitutional Law II | 1 | 4 |
| 50 | Local Self Government | 0 | 5 |
| 51 | Jurisprudence | 1 | 5 |
| 52 | Torts & Consumer Protection Act | 1 | 5 |
| 53 | Family Law I | 1 | 5 |
| 54 | Law of Contract I | 1 | 5 |
| 55 | Banking Law & Negotiable Instruments Act | 1 | 5 |
| 56 | Principles of Public Administration | 0 | 6 |
| 57 | Family Law II | 1 | 6 |
| 58 | Law of Contract II | 1 | 6 |
| 59 | Social Research Methods | 0 | 6 |
| 60 | Bhartiya Nyaya Sanhita | 1 | 7 |
| 61 | Property Law and Easement Act | 1 | 7 |
| 62 | Labour Law I | 1 | 7 |
| 63 | Administrative Law | 1 | 7 |
| 64 | Professional Ethics | 1 | 7 |
| 65 | Bharatiya Nagarik Suraksha Sanhita | 1 | 8 |
| 66 | Company Law | 1 | 8 |
| 67 | Labour Law II | 1 | 8 |
| 68 | Interpretation of Statutes | 1 | 8 |
| 69 | Alternative Dispute Resolution (ADR) | 1 | 8 |
| 70 | Bharatiya Sakshya Adhiniyam | 1 | 9 |
| 71 | Civil Procedure Code & Limitation Act | 1 | 9 |
| 72 | Principles of Taxation | 1 | 9 |
| 73 | Environmental Law | 1 | 9 |
| 74 | Land Laws | 1 | 9 |
| 75 | Drafting, Pleading & Conveyance | 1 | 9 |
| 76 | Intellectual Property Law | 1 | 10 |
| 77 | Public International Law | 1 | 10 |
| 78 | Insurance Law | 1 | 10 |
| 79 | Moot Court & Clinical Legal Education | 1 | 10 |
| 80 | Law of Contract I | 1 | 1 |
| 81 | Constitutional Law I | 1 | 1 |
| 82 | Torts & Consumer Protection Act | 1 | 1 |
| 83 | Jurisprudence | 1 | 1 |
| 84 | Family Law I | 1 | 1 |
| 85 | Banking Law & Negotiable Instruments Act | 1 | 1 |
| 86 | Law of Contract II | 1 | 2 |
| 87 | Constitutional Law II | 1 | 2 |
| 88 | Family Law II | 1 | 2 |
| 89 | Motor Vehicle Act | 1 | 2 |
| 90 | Social Research Methods | 0 | 2 |
| 91 | Bhartiya Nyaya Sanhita | 1 | 3 |
| 92 | Property Law and Easement Act | 1 | 3 |
| 93 | Labour Law I | 1 | 3 |
| 94 | Administrative Law | 1 | 3 |
| 95 | Professional Ethics | 1 | 3 |
| 96 | Bharatiya Nagarik Suraksha Sanhita | 1 | 4 |
| 97 | Company Law | 1 | 4 |
| 98 | Labour Law II | 1 | 4 |
| 99 | Interpretation of Statutes | 1 | 4 |
| 100 | ADR (Clinical Legal Education) | 1 | 4 |
| 101 | Bharatiya Sakshya Adhiniyam | 1 | 5 |
| 102 | Civil Procedure Code & Limitation Act | 1 | 5 |
| 103 | Principles of Taxation | 1 | 5 |
| 104 | Environmental Law | 1 | 5 |
| 105 | Land Laws | 1 | 5 |
| 106 | Drafting, Pleading & Conveyance | 1 | 5 |
| 107 | Intellectual Property Law | 1 | 6 |
| 108 | Public International Law | 1 | 6 |
| 109 | Insurance Law | 1 | 6 |
| 110 | Moot Court & Clinical Legal Education | 1 | 6 |
|  |  |  |  |

App\_teacher ->

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | John | Doe | john.doe@example.com |  | 2 | 0 |
| 2 | Pranav | Sirsufale | pranav@gmail.com | 8482852301 | 3 | 0 |
| 3 | teacher | one | one@gmail.com |  | 4 | 0 |
| 4 | teacher | two | two@gmail.com |  | 5 | 0 |
| 5 | teacher | three | three@gmail.com |  | 6 | 0 |
| 6 | teacher | four | four@gmail.com |  | 7 | 0 |
| 7 | teacher | five | five@gmail.com |  | 8 | 0 |
| 8 | teacher | six | six@gmail.com |  | 9 | 0 |
| 9 | teacher | seven | seven@gmail.com |  | 10 | 0 |
| 10 | teacher | eight | eight@gmail.com |  | 11 | 0 |
| 11 | teacher | nine | nine@gmail.com |  | 12 | 0 |
| 12 | teacher | ten | ten@gmail.com |  | 13 | 0 |
| 13 | admin | admin\_last\_name | admin@gmail.com |  | 14 | 1 |
|  |  |  |  |  |  |  |

Auth\_user ->

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | pbkdf2\_sha256$870000$Me9YZI6YfmhjX5LJI0WBHt$W3xNBZ+tlmjsWwiziX7f07jSwR0bRoMOR57Hkls+fCA= |  | 0 | pranav |  |  |  | 0 | 1 | 2025-03-25 03:20:11.593406 |
| 2 | pbkdf2\_sha256$870000$50fSDLFD0fE5lURLDq9qVc$BNEr7PBQ8zvmLjjDUETH2F3B40uEw7B4GuvK6NjULQo= |  | 0 | john.doe |  |  |  | 0 | 1 | 2025-03-25 03:21:57.704575 |
| 3 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= | 2025-04-05 04:43:25.436272 | 0 | pranav.sirsufale |  |  |  | 0 | 1 | 2025-03-25 10:20:28.637956 |
| 4 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher1 |  |  | one@gmail.com | 0 | 1 | 2025-03-27 17:05:18.948975 |
| 5 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher2 |  |  | two@gmail.com | 0 | 1 | 2025-03-27 17:05:19.379372 |
| 6 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher3 |  |  | three@gmail.com | 0 | 1 | 2025-03-27 17:05:19.425674 |
| 7 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher4 |  |  | four@gmail.com | 0 | 1 | 2025-03-27 17:05:19.444986 |
| 8 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher5 |  |  | five@gmail.com | 0 | 1 | 2025-03-27 17:05:19.500480 |
| 9 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher6 |  |  | six@gmail.com | 0 | 1 | 2025-03-27 17:05:19.528860 |
| 10 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher7 |  |  | seven@gmail.com | 0 | 1 | 2025-03-27 17:05:19.545909 |
| 11 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher8 |  |  | eight@gmail.com | 0 | 1 | 2025-03-27 17:05:19.564738 |
| 12 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= | 2025-04-01 10:27:35.986448 | 0 | teacher9 |  |  | nine@gmail.com | 0 | 1 | 2025-03-27 17:05:19.595600 |
| 13 | pbkdf2\_sha256$870000$XYnI0TJYmgkOeCXoW4HUTD$/LyDuIJ77OD8cfEvd65VBayMQDfBZyqK0u8YfLQR5M4= |  | 0 | teacher10 |  |  | ten@gmail.com | 0 | 1 | 2025-03-27 17:05:19.613329 |
| 14 | pbkdf2\_sha256$870000$X7EOnRDXkTdYSQUgd627z2$wG8Mh0ph0O6YOnWzCNuP/a/nyaofD22DHmLnjTTAAOI= | 2025-03-31 06:49:06.310620 | 0 | admin |  |  | admin@gmail.com | 0 | 1 | 2025-03-30 18:07:31.745190 |
|  |  |  |  |  |  |  |  |  |  |  |

App\_student ->

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | G240001 | Alice | Smith | alice@example.com | 1 |  |
| 4 | G240002 | Ishita | Verma | ishita.verma@example.com | 2 | 9876543211 |
| 5 | G240003 | Rohan | Iyer | rohan.iyer@example.com | 1 | 9876543212 |
| 6 | G240004 | Priya | Chopra | priya.chopra@example.com | 2 | 9876543213 |
| 7 | G240005 | Vivaan | Malhotra | vivaan.malhotra@example.com | 1 | 9876543214 |
| 8 | G240006 | Aditya | Kapoor | aditya.kapoor@example.com | 2 | 9876543215 |
| 9 | G240007 | Kabir | Menon | kabir.menon@example.com | 1 | 9876543216 |
| 10 | G240008 | Swati | Bose | swati.bose@example.com | 2 | 9876543217 |
| 11 | G240009 | Ananya | Chopra | ananya.chopra@example.com | 1 | 9876543218 |
| 12 | G240010 | Dev | Iyer | dev.iyer@example.com | 2 | 9876543219 |
| 13 | G240011 | Harsh | Verma | harsh.verma@example.com | 1 | 9876543220 |
| 14 | G240012 | Ritika | Patel | ritika.patel@example.com | 2 | 9876543221 |
| 15 | G240013 | Pooja | Singh | pooja.singh@example.com | 1 | 9876543222 |
| 16 | G240014 | Tanya | Kapoor | tanya.kapoor@example.com | 2 | 9876543223 |
| 17 | G240015 | Simran | Sharma | simran.sharma@example.com | 1 | 9876543224 |
| 18 | G240016 | Vikram | Malhotra | vikram.malhotra@example.com | 2 | 9876543225 |
| 19 | G240017 | Neha | Bose | neha.bose@example.com | 1 | 9876543226 |
| 20 | G240018 | Ishaan | Menon | ishaan.menon@example.com | 2 | 9876543227 |
| 21 | G240019 | Arjun | Chopra | arjun.chopra@example.com | 1 | 9876543228 |
| 22 | G240020 | Devika | Patel | devika.patel@example.com | 2 | 9876543229 |
| 23 | NG240001 | Aarav | Kapoor | aarav.kapoor@example.com | 4 | 9876543230 |
| 24 | NG240002 | Rohan | Bose | rohan.bose@example.com | 6 | 9876543231 |
| 25 | NG240003 | Vivaan | Singh | vivaan.singh@example.com | 4 | 9876543232 |
| 26 | NG240004 | Kabir | Malhotra | kabir.malhotra@example.com | 6 | 9876543233 |
| 27 | NG240005 | Neha | Sharma | neha.sharma@example.com | 4 | 9876543234 |
| 28 | NG240006 | Ananya | Patel | ananya.patel@example.com | 6 | 9876543235 |
| 29 | NG240007 | Dev | Chopra | dev.chopra@example.com | 4 | 9876543236 |
| 30 | NG240008 | Swati | Iyer | swati.iyer@example.com | 6 | 9876543237 |
| 34 | NG240009 | Harsh | Verma | harshs.verma@example.com | 4 | 9876543238 |
| 35 | NG240010 | Pooja | Menon | pooja.menon@example.com | 6 | 9876543239 |
| 36 | NG240011 | Ishita | Sharma | ishita.sharma@example.com | 4 | 9876543240 |
| 37 | NG240012 | Simran | Kapoor | simran.kapoor@example.com | 6 | 9876543241 |
| 38 | NG240013 | Tanya | Bose | tanya.bose@example.com | 4 | 9876543242 |
| 39 | NG240014 | Ritika | Chopra | ritika.chopra@example.com | 6 | 9876543243 |
| 40 | NG240015 | Vikram | Patel | vikram.patel@example.com | 4 | 9876543244 |
| 44 | NG240016 | Ishaan | Menon | ishaanw.menon@example.com | 6 | 9876543245 |
| 45 | NG240017 | Priya | Malhotra | priya.malhotra@example.com | 4 | 9876543246 |
| 46 | NG240018 | Aditya | Singh | aditya.singh@example.com | 6 | 9876543247 |
| 47 | NG240019 | Devika | Verma | devika.verma@example.com | 4 | 9876543248 |
| 48 | NG240020 | Arjun | Iyer | arjun.iyer@example.com | 6 | 9876543249 |
|  |  |  |  |  |  |  |

App\_timetable ->

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | Monday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 1 | 30 | 1 |
| 5 | Thursday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 1 | 30 | 1 |
| 6 | Saturday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 1 | 30 | 1 |
| 8 | Monday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 31 | 12 |
| 9 | Friday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 31 | 12 |
| 10 | Saturday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 31 | 12 |
| 11 | Monday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 85 | 2 |
| 12 | Wednesday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 85 | 2 |
| 13 | Saturday | 08:30:00.000000 | 2025-03-01 | 2025-08-31 | 2 | 85 | 2 |
|  |  |  |  |  |  |  |  |

Session ->

|  |  |  |  |
| --- | --- | --- | --- |
| 22 | 2025-03-03 | Scheduled | 4 |
| 23 | 2025-03-10 | Scheduled | 4 |
| 24 | 2025-03-17 | Scheduled | 4 |
| 25 | 2025-03-24 | Scheduled | 4 |
| 26 | 2025-03-31 | Scheduled | 4 |
| 27 | 2025-04-07 | Scheduled | 4 |
| 28 | 2025-04-14 | Scheduled | 4 |
| 29 | 2025-04-21 | Scheduled | 4 |
| 30 | 2025-04-28 | Scheduled | 4 |
| 31 | 2025-05-05 | Scheduled | 4 |
| 32 | 2025-05-12 | Scheduled | 4 |
| 33 | 2025-05-19 | Scheduled | 4 |
| 34 | 2025-05-26 | Scheduled | 4 |
| 35 | 2025-06-02 | Scheduled | 4 |
| 36 | 2025-06-09 | Scheduled | 4 |
| 37 | 2025-06-16 | Scheduled | 4 |
| 38 | 2025-06-23 | Scheduled | 4 |
| 39 | 2025-06-30 | Scheduled | 4 |
| 40 | 2025-07-07 | Scheduled | 4 |
| 41 | 2025-07-14 | Scheduled | 4 |
| 42 | 2025-07-21 | Scheduled | 4 |
| 43 | 2025-07-28 | Scheduled | 4 |
| 44 | 2025-08-04 | Scheduled | 4 |
| 45 | 2025-08-11 | Scheduled | 4 |
| 46 | 2025-08-18 | Scheduled | 4 |
| 47 | 2025-08-25 | Scheduled | 4 |
| 48 | 2025-03-06 | Scheduled | 5 |
| 49 | 2025-03-13 | Scheduled | 5 |
| 50 | 2025-03-20 | Scheduled | 5 |
| 51 | 2025-03-27 | Scheduled | 5 |
| 52 | 2025-04-03 | Scheduled | 5 |
| 53 | 2025-04-10 | Scheduled | 5 |
| 54 | 2025-04-17 | Scheduled | 5 |
| 55 | 2025-04-24 | Scheduled | 5 |
| 56 | 2025-05-01 | Scheduled | 5 |
| 57 | 2025-05-08 | Scheduled | 5 |
| 58 | 2025-05-15 | Scheduled | 5 |
| 59 | 2025-05-22 | Scheduled | 5 |
| 60 | 2025-05-29 | Scheduled | 5 |
| 61 | 2025-06-05 | Scheduled | 5 |
| 62 | 2025-06-12 | Scheduled | 5 |
| 63 | 2025-06-19 | Scheduled | 5 |
| 64 | 2025-06-26 | Scheduled | 5 |
| 65 | 2025-07-03 | Scheduled | 5 |
| 66 | 2025-07-10 | Scheduled | 5 |
| 67 | 2025-07-17 | Scheduled | 5 |
| 68 | 2025-07-24 | Scheduled | 5 |
| 69 | 2025-07-31 | Scheduled | 5 |
| 70 | 2025-08-07 | Scheduled | 5 |
| 71 | 2025-08-14 | Scheduled | 5 |
| 72 | 2025-08-21 | Scheduled | 5 |
| 73 | 2025-08-28 | Scheduled | 5 |
| 74 | 2025-03-01 | Scheduled | 6 |
| 75 | 2025-03-08 | Scheduled | 6 |
| 76 | 2025-03-15 | Scheduled | 6 |
| 77 | 2025-03-22 | Scheduled | 6 |
| 78 | 2025-03-29 | Scheduled | 6 |
| 79 | 2025-04-05 | Scheduled | 6 |
| 80 | 2025-04-12 | Scheduled | 6 |
| 81 | 2025-04-19 | Scheduled | 6 |
| 82 | 2025-04-26 | Scheduled | 6 |
| 83 | 2025-05-03 | Scheduled | 6 |
| 84 | 2025-05-10 | Scheduled | 6 |
| 85 | 2025-05-17 | Scheduled | 6 |
| 86 | 2025-05-24 | Scheduled | 6 |
| 87 | 2025-05-31 | Scheduled | 6 |
| 88 | 2025-06-07 | Scheduled | 6 |
| 89 | 2025-06-14 | Scheduled | 6 |
| 90 | 2025-06-21 | Scheduled | 6 |
| 91 | 2025-06-28 | Scheduled | 6 |
| 92 | 2025-07-05 | Scheduled | 6 |
| 93 | 2025-07-12 | Scheduled | 6 |
| 94 | 2025-07-19 | Scheduled | 6 |
| 95 | 2025-07-26 | Scheduled | 6 |
| 96 | 2025-08-02 | Scheduled | 6 |
| 97 | 2025-08-09 | Scheduled | 6 |
| 98 | 2025-08-16 | Scheduled | 6 |
| 99 | 2025-08-23 | Scheduled | 6 |
| 100 | 2025-08-30 | Scheduled | 6 |
| 101 | 2025-03-03 | Scheduled | 8 |
| 102 | 2025-03-10 | Scheduled | 8 |
| 103 | 2025-03-17 | Scheduled | 8 |
| 104 | 2025-03-24 | Scheduled | 8 |
| 105 | 2025-03-31 | Scheduled | 8 |
| 106 | 2025-04-07 | Scheduled | 8 |
| 107 | 2025-04-14 | Scheduled | 8 |
| 108 | 2025-04-21 | Scheduled | 8 |
| 109 | 2025-04-28 | Scheduled | 8 |
| 110 | 2025-05-05 | Scheduled | 8 |
| 111 | 2025-05-12 | Scheduled | 8 |
| 112 | 2025-05-19 | Scheduled | 8 |
| 113 | 2025-05-26 | Scheduled | 8 |
| 114 | 2025-06-02 | Scheduled | 8 |
| 115 | 2025-06-09 | Scheduled | 8 |
| 116 | 2025-06-16 | Scheduled | 8 |
| 117 | 2025-06-23 | Scheduled | 8 |
| 118 | 2025-06-30 | Scheduled | 8 |
| 119 | 2025-07-07 | Scheduled | 8 |
| 120 | 2025-07-14 | Scheduled | 8 |
| 121 | 2025-07-21 | Scheduled | 8 |
| 122 | 2025-07-28 | Scheduled | 8 |
| 123 | 2025-08-04 | Scheduled | 8 |
| 124 | 2025-08-11 | Scheduled | 8 |
| 125 | 2025-08-18 | Scheduled | 8 |
| 126 | 2025-08-25 | Scheduled | 8 |
| 127 | 2025-03-07 | Scheduled | 9 |
| 128 | 2025-03-14 | Scheduled | 9 |
| 129 | 2025-03-21 | Scheduled | 9 |
| 130 | 2025-03-28 | Scheduled | 9 |
| 131 | 2025-04-04 | Scheduled | 9 |
| 132 | 2025-04-11 | Scheduled | 9 |
| 133 | 2025-04-18 | Scheduled | 9 |
| 134 | 2025-04-25 | Scheduled | 9 |
| 135 | 2025-05-02 | Scheduled | 9 |
| 136 | 2025-05-09 | Scheduled | 9 |
| 137 | 2025-05-16 | Scheduled | 9 |
| 138 | 2025-05-23 | Scheduled | 9 |
| 139 | 2025-05-30 | Scheduled | 9 |
| 140 | 2025-06-06 | Scheduled | 9 |
| 141 | 2025-06-13 | Scheduled | 9 |
| 142 | 2025-06-20 | Scheduled | 9 |
| 143 | 2025-06-27 | Scheduled | 9 |
| 144 | 2025-07-04 | Scheduled | 9 |
| 145 | 2025-07-11 | Scheduled | 9 |
| 146 | 2025-07-18 | Scheduled | 9 |
| 147 | 2025-07-25 | Scheduled | 9 |
| 148 | 2025-08-01 | Scheduled | 9 |
| 149 | 2025-08-08 | Scheduled | 9 |
| 150 | 2025-08-15 | Scheduled | 9 |
| 151 | 2025-08-22 | Scheduled | 9 |
| 152 | 2025-08-29 | Scheduled | 9 |
| 153 | 2025-03-01 | Scheduled | 10 |
| 154 | 2025-03-08 | Scheduled | 10 |
| 155 | 2025-03-15 | Scheduled | 10 |
| 156 | 2025-03-22 | Scheduled | 10 |
| 157 | 2025-03-29 | Scheduled | 10 |
| 158 | 2025-04-05 | Scheduled | 10 |
| 159 | 2025-04-12 | Scheduled | 10 |
| 160 | 2025-04-19 | Scheduled | 10 |
| 161 | 2025-04-26 | Scheduled | 10 |
| 162 | 2025-05-03 | Scheduled | 10 |
| 163 | 2025-05-10 | Scheduled | 10 |
| 164 | 2025-05-17 | Scheduled | 10 |
| 165 | 2025-05-24 | Scheduled | 10 |
| 166 | 2025-05-31 | Scheduled | 10 |
| 167 | 2025-06-07 | Scheduled | 10 |
| 168 | 2025-06-14 | Scheduled | 10 |
| 169 | 2025-06-21 | Scheduled | 10 |
| 170 | 2025-06-28 | Scheduled | 10 |
| 171 | 2025-07-05 | Scheduled | 10 |
| 172 | 2025-07-12 | Scheduled | 10 |
| 173 | 2025-07-19 | Scheduled | 10 |
| 174 | 2025-07-26 | Scheduled | 10 |
| 175 | 2025-08-02 | Scheduled | 10 |
| 176 | 2025-08-09 | Scheduled | 10 |
| 177 | 2025-08-16 | Scheduled | 10 |
| 178 | 2025-08-23 | Scheduled | 10 |
| 179 | 2025-08-30 | Scheduled | 10 |
| 180 | 2025-03-03 | Completed | 11 |
| 181 | 2025-03-10 | Scheduled | 11 |
| 182 | 2025-03-17 | Scheduled | 11 |
| 183 | 2025-03-24 | Scheduled | 11 |
| 184 | 2025-03-31 | Scheduled | 11 |
| 185 | 2025-04-07 | Scheduled | 11 |
| 186 | 2025-04-14 | Scheduled | 11 |
| 187 | 2025-04-21 | Scheduled | 11 |
| 188 | 2025-04-28 | Scheduled | 11 |
| 189 | 2025-05-05 | Scheduled | 11 |
| 190 | 2025-05-12 | Scheduled | 11 |
| 191 | 2025-05-19 | Scheduled | 11 |
| 192 | 2025-05-26 | Scheduled | 11 |
| 193 | 2025-06-02 | Scheduled | 11 |
| 194 | 2025-06-09 | Scheduled | 11 |
| 195 | 2025-06-16 | Scheduled | 11 |
| 196 | 2025-06-23 | Scheduled | 11 |
| 197 | 2025-06-30 | Scheduled | 11 |
| 198 | 2025-07-07 | Scheduled | 11 |
| 199 | 2025-07-14 | Scheduled | 11 |
| 200 | 2025-07-21 | Scheduled | 11 |
| 201 | 2025-07-28 | Scheduled | 11 |
| 202 | 2025-08-04 | Scheduled | 11 |
| 203 | 2025-08-11 | Scheduled | 11 |
| 204 | 2025-08-18 | Scheduled | 11 |
| 205 | 2025-08-25 | Scheduled | 11 |
| 206 | 2025-03-05 | Scheduled | 12 |
| 207 | 2025-03-12 | Scheduled | 12 |
| 208 | 2025-03-19 | Scheduled | 12 |
| 209 | 2025-03-26 | Scheduled | 12 |
| 210 | 2025-04-02 | Completed | 12 |
| 211 | 2025-04-09 | Scheduled | 12 |
| 212 | 2025-04-16 | Scheduled | 12 |
| 213 | 2025-04-23 | Scheduled | 12 |
| 214 | 2025-04-30 | Scheduled | 12 |
| 215 | 2025-05-07 | Scheduled | 12 |
| 216 | 2025-05-14 | Scheduled | 12 |
| 217 | 2025-05-21 | Scheduled | 12 |
| 218 | 2025-05-28 | Scheduled | 12 |
| 219 | 2025-06-04 | Scheduled | 12 |
| 220 | 2025-06-11 | Scheduled | 12 |
| 221 | 2025-06-18 | Scheduled | 12 |
| 222 | 2025-06-25 | Scheduled | 12 |
| 223 | 2025-07-02 | Scheduled | 12 |
| 224 | 2025-07-09 | Scheduled | 12 |
| 225 | 2025-07-16 | Scheduled | 12 |
| 226 | 2025-07-23 | Scheduled | 12 |
| 227 | 2025-07-30 | Scheduled | 12 |
| 228 | 2025-08-06 | Scheduled | 12 |
| 229 | 2025-08-13 | Scheduled | 12 |
| 230 | 2025-08-20 | Scheduled | 12 |
| 231 | 2025-08-27 | Scheduled | 12 |
| 232 | 2025-03-01 | Completed | 13 |
| 233 | 2025-03-08 | Scheduled | 13 |
| 234 | 2025-03-15 | Scheduled | 13 |
| 235 | 2025-03-22 | Scheduled | 13 |
| 236 | 2025-03-29 | Scheduled | 13 |
| 237 | 2025-04-05 | Scheduled | 13 |
| 238 | 2025-04-12 | Scheduled | 13 |
| 239 | 2025-04-19 | Scheduled | 13 |
| 240 | 2025-04-26 | Scheduled | 13 |
| 241 | 2025-05-03 | Scheduled | 13 |
| 242 | 2025-05-10 | Scheduled | 13 |
| 243 | 2025-05-17 | Scheduled | 13 |
| 244 | 2025-05-24 | Scheduled | 13 |
| 245 | 2025-05-31 | Scheduled | 13 |
| 246 | 2025-06-07 | Scheduled | 13 |
| 247 | 2025-06-14 | Scheduled | 13 |
| 248 | 2025-06-21 | Scheduled | 13 |
| 249 | 2025-06-28 | Scheduled | 13 |
| 250 | 2025-07-05 | Scheduled | 13 |
| 251 | 2025-07-12 | Scheduled | 13 |
| 252 | 2025-07-19 | Scheduled | 13 |
| 253 | 2025-07-26 | Scheduled | 13 |
| 254 | 2025-08-02 | Scheduled | 13 |
| 255 | 2025-08-09 | Scheduled | 13 |
| 256 | 2025-08-16 | Scheduled | 13 |
| 257 | 2025-08-23 | Scheduled | 13 |
| 258 | 2025-08-30 | Scheduled | 13 |
|  |  |  |  |

App\_attendance ->

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Present | 2025-03-31 13:21:41.405355 | 210 | 4 | 2 |
| 2 | Present | 2025-03-31 13:21:41.422428 | 210 | 6 | 2 |
| 3 | Present | 2025-03-31 13:21:41.437027 | 210 | 8 | 2 |
| 4 | Present | 2025-03-31 13:21:41.453021 | 210 | 10 | 2 |
| 5 | Present | 2025-03-31 13:21:41.471972 | 210 | 12 | 2 |
| 6 | Present | 2025-03-31 13:21:41.482969 | 210 | 14 | 2 |
| 7 | Present | 2025-03-31 13:21:41.491958 | 210 | 16 | 2 |
| 8 | Present | 2025-03-31 13:21:41.500142 | 210 | 18 | 2 |
| 9 | Present | 2025-03-31 13:21:41.510134 | 210 | 20 | 2 |
| 10 | Present | 2025-03-31 13:21:41.520139 | 210 | 22 | 2 |
| 11 | Present | 2025-03-31 13:21:22.094293 | 232 | 4 | 2 |
| 12 | Absent | 2025-03-31 13:21:22.117264 | 232 | 6 | 2 |
| 13 | Present | 2025-03-31 13:21:22.125505 | 232 | 8 | 2 |
| 14 | Present | 2025-03-31 13:21:22.148070 | 232 | 10 | 2 |
| 15 | Present | 2025-03-31 13:21:22.156687 | 232 | 12 | 2 |
| 16 | Present | 2025-03-31 13:21:22.166301 | 232 | 14 | 2 |
| 17 | Present | 2025-03-31 13:21:22.175712 | 232 | 16 | 2 |
| 18 | Present | 2025-03-31 13:21:22.185677 | 232 | 18 | 2 |
| 19 | Present | 2025-03-31 13:21:22.194379 | 232 | 20 | 2 |
| 20 | Present | 2025-03-31 13:21:22.202288 | 232 | 22 | 2 |
| 21 | Present | 2025-03-31 13:21:30.273337 | 180 | 4 | 2 |
| 22 | Absent | 2025-03-31 13:21:30.296273 | 180 | 6 | 2 |
| 23 | Present | 2025-03-31 13:21:30.304733 | 180 | 8 | 2 |
| 24 | Absent | 2025-03-31 13:21:30.313723 | 180 | 10 | 2 |
| 25 | Present | 2025-03-31 13:21:30.323020 | 180 | 12 | 2 |
| 26 | Absent | 2025-03-31 13:21:30.332014 | 180 | 14 | 2 |
| 27 | Absent | 2025-03-31 13:21:30.347048 | 180 | 16 | 2 |
| 28 | Present | 2025-03-31 13:21:30.365353 | 180 | 18 | 2 |
| 29 | Present | 2025-03-31 13:21:30.396276 | 180 | 20 | 2 |
| 30 | Absent | 2025-03-31 13:21:30.410457 | 180 | 22 | 2 |
|  |  |  |  |  |  |

App\_calendarexception ->

>>>> serializer.py ->

from rest\_framework import serializers

from .models import Teacher,Student, Subject , Program , Timetable , Session, Attendance , Section , CalendarException

class TeacherSerializer(serializers.ModelSerializer):

    class Meta:

        model = Teacher

        fields = ['id' , 'first\_name' , 'last\_name' , 'email','phone' , 'is\_admin']

class StudentSerializer(serializers.ModelSerializer):

    section = serializers.StringRelatedField()

    subjects = serializers.PrimaryKeyRelatedField(many = True , read\_only = True)

    class Meta:

        model = Student

        fields = ['id','roll\_number','first\_name','last\_name','email','phone','section','subjects']

class ProgramSerializer(serializers.ModelSerializer):

    class Meta:

        model = Program

        fields = ['id' , 'name' , 'duration']

class SubjectSerializer(serializers.ModelSerializer):

    teacher = TeacherSerializer(read\_only = True)

    class Meta:

        model = Subject

        fields = ['id' , 'name' ,'is\_law\_subject', 'semester','teacher']

class DailyScheduleSerializer(serializers.Serializer):

    day\_of\_week = serializers.ChoiceField(choices= Timetable.DAY\_CHOICES)

    subject = serializers.PrimaryKeyRelatedField(queryset=Subject.objects.all())

    start\_time = serializers.ChoiceField(choices=Timetable.LECTURE\_SLOTS)

class CalendarExceptionSerializer(serializers.ModelSerializer):

    class Meta:

        model = CalendarException

        fields = ['id','date','description']

class SectionSerializer(serializers.ModelSerializer):

    program = serializers.SlugRelatedField(slug\_field='name', read\_only=True)

    available\_semesters = serializers.SerializerMethodField()

    def get\_available\_semesters(self, obj):

        # Get subjects available for this section's program and year

        program\_duration = obj.program.duration\_years \* 2  # Max semesters

        start\_semester = (obj.year - 1) \* 2 + 1

        end\_semester = min(obj.year \* 2, program\_duration)

        subjects = Subject.objects.filter(semester\_\_gte=start\_semester, semester\_\_lte=end\_semester)

        return sorted(set(subject.semester for subject in subjects))

    class Meta:

        model = Section

        fields = ['id', 'name', 'year', 'program', 'available\_semesters']

class TimetableSerializer(serializers.ModelSerializer):

    section = SectionSerializer()

    subject = SubjectSerializer()

    teacher = TeacherSerializer()

    semester = serializers.IntegerField(source='subject.semester', read\_only=True)

    class Meta:

        model = Timetable

        fields = ['id', 'section', 'teacher', 'subject', 'semester', 'day\_of\_week', 'start\_time', 'semester\_start\_date', 'semester\_end\_date']

class SessionSerializer(serializers.ModelSerializer):

    timetable = TimetableSerializer(read\_only=True)

    class Meta:

        model = Session

        fields = ['id','timetable','date','status']

class AttendanceSerializer(serializers.ModelSerializer):

    student = StudentSerializer(read\_only=True)

    session = SessionSerializer(read\_only=True)

    recorded\_by = TeacherSerializer(read\_only=True)

    class Meta:

        model = Attendance

        fields = ['id','student','session','status','timestamp','recorded\_by']

        read\_only\_fields = ['timestamp' , 'recorded\_by']

class TimetableCreateSerializer(serializers.Serializer):

    section = serializers.PrimaryKeyRelatedField(queryset=Section.objects.all())

    teacher = serializers.PrimaryKeyRelatedField(queryset=Teacher.objects.all())

    semester = serializers.IntegerField()

    daily\_schedules = serializers.ListField(

        child=serializers.DictField(child=serializers.CharField())

    )

    def validate\_daily\_schedules(self, value):

        for schedule in value:

            required\_fields = ['day\_of\_week', 'subject', 'start\_time']

            if not all(field in schedule for field in required\_fields):

                raise serializers.ValidationError(f"Each schedule must include {required\_fields}")

            if schedule['day\_of\_week'] not in ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']:

                raise serializers.ValidationError("Invalid day of week")

            if schedule['start\_time'] not in [slot[0] for slot in LECTURE\_SLOTS]:

                raise serializers.ValidationError(f"Start time must be one of {[slot[0] for slot in LECTURE\_SLOTS]}")

        return value

    def validate(self, data):

        section = data['section']

        semester = data['semester']

        program\_duration = section.program.duration\_years \* 2

        start\_semester = (section.year - 1) \* 2 + 1

        end\_semester = min(section.year \* 2, program\_duration)

        if semester < start\_semester or semester > end\_semester:

            raise serializers.ValidationError(f"Semester {semester} is not valid for {section} (valid range: {start\_semester}-{end\_semester})")

        for schedule in data['daily\_schedules']:

            subject = Subject.objects.get(id=schedule['subject'])

            if subject.semester != semester:

                raise serializers.ValidationError(f"Subject {subject.name} (Semester {subject.semester}) does not match selected semester {semester}")

        return data

>>>>> views.py ->

from django.shortcuts import render

from rest\_framework.views import APIView

from django.http import HttpResponse

from django.utils import timezone

from rest\_framework import viewsets, generics, status , serializers

from rest\_framework.permissions import IsAuthenticated

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view , permission\_classes

from rest\_framework.exceptions import ValidationError

from app.permissions import IsAdmin

from django.db.models import Count, Q

from datetime import datetime, timedelta

from .models import Session , Program, Attendance, Teacher, Student, Subject, Timetable, CalendarException , Section

from .serializers import (

    SessionSerializer, AttendanceSerializer , TimetableCreateSerializer, TeacherSerializer, StudentSerializer,

    TimetableSerializer, CalendarExceptionSerializer, ProgramSerializer , SubjectSerializer , SectionSerializer

)

import logging

import traceback

import sys

import jwt

# Logging setup

logger = logging.getLogger(\_\_name\_\_)

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)s %(message)s', stream=sys.stdout)

# Create your views here.

def home(request):

    return HttpResponse('hello world this is home')

class AdminCRUDViewSet(viewsets.ModelViewSet):

    permission\_classes = [IsAuthenticated, IsAdmin]

    def get\_queryset(self):

        return self.queryset

class TeacherViewSet(AdminCRUDViewSet):

    queryset = Teacher.objects.all()

    serializer\_class = TeacherSerializer

class StudentViewSet(AdminCRUDViewSet):

    queryset = Student.objects.all()

    serializer\_class = StudentSerializer

class ProgramViewSet(AdminCRUDViewSet):

    queryset = Program.objects.all()

    serializer\_class = ProgramSerializer

class SubjectViewSet(AdminCRUDViewSet):

    queryset = Subject.objects.all()

    serializer\_class = SubjectSerializer

class SessionViewSet(AdminCRUDViewSet):

    queryset = Session.objects.all()

    serializer\_class = SessionSerializer

class AdminAttendanceStatsView(viewsets.ViewSet):

    permission\_classes = [IsAuthenticated, IsAdmin]

    def list(self, request):

        period = request.query\_params.get('period', 'semester')

        section\_id = request.query\_params.get('section', None)

        specific\_date = request.query\_params.get('date', None)  # For daily stats

        logger.debug(f"Fetching all attendance stats for period: {period}, section: {section\_id}, date: {specific\_date}")

        today = timezone.now().date()

        if period == 'daily' and specific\_date:

            start\_date = end\_date = datetime.strptime(specific\_date, '%Y-%m-%d').date()

        elif period == 'weekly':

            start\_date = today - timedelta(days=today.weekday())

            end\_date = start\_date + timedelta(days=6)

        elif period == 'monthly':

            start\_date = today.replace(day=1)

            end\_date = (start\_date + timedelta(days=31)).replace(day=1) - timedelta(days=1)

        else:  # semester

            timetables = Timetable.objects.all()

            start\_date = min(t.semester\_start\_date for t in timetables) if timetables else today

            end\_date = max(t.semester\_end\_date for t in timetables) if timetables else today

        stats\_query = Attendance.objects.filter(session\_\_date\_\_gte=start\_date, session\_\_date\_\_lte=end\_date)

        if section\_id:

            stats\_query = stats\_query.filter(session\_\_timetable\_\_section\_id=section\_id)

        stats = (

            stats\_query

            .values('student\_\_id', 'student\_\_first\_name', 'student\_\_last\_name', 'student\_\_roll\_number', 'recorded\_by\_\_first\_name', 'recorded\_by\_\_last\_name')

            .annotate(

                total\_sessions=Count('session'),

                present=Count('session', filter=Q(status='Present')),

                absent=Count('session', filter=Q(status='Absent'))

            )

        )

        response = [

            {

                'student\_id': stat['student\_\_id'],

                'name': f"{stat['student\_\_first\_name']} {stat['student\_\_last\_name']}",

                'roll\_number': stat['student\_\_roll\_number'],

                'total\_sessions': stat['total\_sessions'],

                'present': stat['present'],

                'absent': stat['absent'],

                'attendance\_percentage': round((stat['present'] / stat['total\_sessions']) \* 100, 2) if stat['total\_sessions'] > 0 else 0,

                'recorded\_by': f"{stat['recorded\_by\_\_first\_name']} {stat['recorded\_by\_\_last\_name']}"

            }

            for stat in stats

        ]

        return Response({

            'period': period,

            'section': section\_id,

            'start\_date': start\_date.isoformat(),

            'end\_date': end\_date.isoformat(),

            'stats': response

        })

class TeacherAttendanceStatsView(APIView):

    permission\_classes = [IsAuthenticated]

    def get(self, request):

        try:

            teacher = Teacher.objects.get(user=request.user)

            period = request.query\_params.get('period', 'semester')

            start\_date\_param = request.query\_params.get('start\_date', None)

            end\_date\_param = request.query\_params.get('end\_date', None)

            logger.debug(f"Fetching attendance stats for teacher {teacher.id}, period: {period}, start\_date: {start\_date\_param}, end\_date: {end\_date\_param}")

            timetables = Timetable.objects.filter(teacher=teacher)

            session\_ids = Session.objects.filter(timetable\_\_in=timetables).values\_list('id', flat=True)

            students = Student.objects.filter(section\_\_in=timetables.values('section')).distinct()

            today = timezone.now().date()

            if start\_date\_param and end\_date\_param:

                start\_date = datetime.strptime(start\_date\_param, '%Y-%m-%d').date()

                end\_date = datetime.strptime(end\_date\_param, '%Y-%m-%d').date()

            elif period == 'weekly':

                start\_date = today - timedelta(days=today.weekday())

                end\_date = start\_date + timedelta(days=6)

            elif period == 'monthly':

                start\_date = today.replace(day=1)

                end\_date = (start\_date + timedelta(days=31)).replace(day=1) - timedelta(days=1)

            else:  # semester

                if not timetables.exists():

                    start\_date = today

                    end\_date = today

                else:

                    start\_date = min(t.semester\_start\_date for t in timetables)

                    end\_date = max(t.semester\_end\_date for t in timetables)

            attendance\_stats = (

                Attendance.objects.filter(session\_\_id\_\_in=session\_ids, student\_\_in=students)

                .filter(session\_\_date\_\_gte=start\_date, session\_\_date\_\_lte=end\_date)

                .values('student\_\_id', 'student\_\_first\_name', 'student\_\_last\_name', 'student\_\_roll\_number')

                .annotate(

                    total\_sessions=Count('session'),

                    present=Count('session', filter=Q(status='Present')),

                    absent=Count('session', filter=Q(status='Absent'))

                )

            )

            stats = [

                {

                    'student\_id': stat['student\_\_id'],

                    'name': f"{stat['student\_\_first\_name']} {stat['student\_\_last\_name']}",

                    'roll\_number': stat['student\_\_roll\_number'],

                    'total\_sessions': stat['total\_sessions'],

                    'present': stat['present'],

                    'absent': stat['absent'],

                    'attendance\_percentage': round((stat['present'] / stat['total\_sessions']) \* 100, 2) if stat['total\_sessions'] > 0 else 0

                }

                for stat in attendance\_stats

            ]

            return Response({

                'period': period,

                'start\_date': start\_date.isoformat(),

                'end\_date': end\_date.isoformat(),

                'stats': stats

            }, status=200)

        except Teacher.DoesNotExist:

            logger.error("Teacher not found for user: %s", request.user)

            return Response({"error": "Teacher not found"}, status=404)

        except Exception as e:

            logger.error("Error fetching attendance stats: %s", str(e))

            return Response({"error": str(e)}, status=500)

class StudentViewSet(viewsets.ReadOnlyModelViewSet):

    queryset = Student.objects.all()

    serializer\_class = StudentSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        return Student.objects.filter(section\_\_timetable\_\_teacher\_\_user = self.request.user ).distinct()

class SessionViewSet(viewsets.ModelViewSet):

    queryset = Session.objects.all()

    serializer\_class = SessionSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        return Session.objects.filter(timetable\_\_teacher\_\_user=self.request.user)

class AttendanceViewSet(viewsets.ModelViewSet):

    queryset = Attendance.objects.all()

    serializer\_class = AttendanceSerializer

    permission\_classes = [IsAuthenticated]

    def perform\_create(self, serializer):

        serializer.save(recorded\_by=Teacher.objects.get(user=self.request.user))

'''

class TimetableViewSet(viewsets.ModelViewSet):

    queryset = Timetable.objects.all()

    permission\_classes = [IsAuthenticated]

    def get\_serializer\_class(self):

        if self.action in ['create', 'update', 'partial\_update']:

            return TimetableCreateSerializer

        return TimetableSerializer

    def get\_queryset(self):

        logger.debug("Fetching queryset for user: %s", self.request.user)

        return Timetable.objects.filter(teacher\_\_user=self.request.user)

    def create(self, request, \*args, \*\*kwargs):

        logger.debug("POST request received at /api/timetables/")

        logger.debug("Request data: %s", request.data)

        try:

            mutable\_data = request.data.copy()

            teacher = Teacher.objects.get(user=request.user)

            mutable\_data['teacher'] = teacher.id

            serializer = self.get\_serializer(data=mutable\_data)

            serializer.is\_valid(raise\_exception=True)

            timetable\_data = self.perform\_create(serializer)

            logger.info("Timetable created successfully")

            return Response(timetable\_data, status=status.HTTP\_201\_CREATED)

        except ValidationError as e:

            logger.error("Serializer validation failed: %s", str(e))

            return Response({"detail": str(e)}, status=status.HTTP\_400\_BAD\_REQUEST)

        except Exception as e:

            logger.error("Unexpected error: %s\n%s", str(e), traceback.format\_exc())

            return Response({"detail": f"Server error: {str(e)}"}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

    def perform\_create(self, serializer):

        validated\_data = serializer.validated\_data

        teacher = validated\_data['teacher']

        daily\_schedules = validated\_data['daily\_schedules']

        section = validated\_data['section']

        if not daily\_schedules:

            raise ValidationError("At least one daily schedule is required.")

        for day in ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']:

            existing = Timetable.objects.filter(section=section, day\_of\_week=day).count()

            new\_for\_day = len([s for s in daily\_schedules if s['day\_of\_week'] == day])

            if existing + new\_for\_day > 5:

                raise ValidationError(f"Cannot schedule more than 5 lectures on {day} for this section.")

            for schedule in [s for s in daily\_schedules if s['day\_of\_week'] == day]:

                if Timetable.objects.filter(

                    teacher=teacher,

                    day\_of\_week=day,

                    start\_time=schedule['start\_time'],

                    semester\_start\_date=validated\_data['semester\_start\_date'],

                    semester\_end\_date=validated\_data['semester\_end\_date']

                ).exists():

                    raise ValidationError(f"Teacher already scheduled on {day} at {schedule['start\_time']} for this semester.")

        day\_of\_week\_map = {'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3, 'Friday': 4, 'Saturday': 5}

        created\_timetables = []

        for schedule in daily\_schedules:

            timetable = Timetable.objects.create(

                section=section,

                teacher=teacher,

                subject=Subject.objects.get(id=schedule['subject']),

                day\_of\_week=schedule['day\_of\_week'],

                start\_time=schedule['start\_time'],

                semester\_start\_date=validated\_data['semester\_start\_date'],

                semester\_end\_date=validated\_data['semester\_end\_date']

            )

            created\_timetables.append(timetable)

            start\_date = validated\_data['semester\_start\_date']

            end\_date = validated\_data['semester\_end\_date']

            current\_date = start\_date

            target\_day = day\_of\_week\_map[timetable.day\_of\_week]

            while current\_date <= end\_date:

                if current\_date.weekday() == target\_day:

                    Session.objects.get\_or\_create(

                        timetable=timetable,

                        date=current\_date,

                        defaults={'status': 'Scheduled'}

                    )

                current\_date += timedelta(days=1)

        # Serialize the created timetables

        timetable\_serializer = TimetableSerializer(created\_timetables, many=True)

        return timetable\_serializer.data

    def update(self, request, \*args, \*\*kwargs):

        instance = self.get\_object()

        serializer = TimetableSerializer(instance, data=request.data, partial=True)  # Use TimetableSerializer for updates

        serializer.is\_valid(raise\_exception=True)

        self.perform\_update(serializer)

        return Response(serializer.data)

    def perform\_update(self, serializer):

        instance = serializer.instance

        Session.objects.filter(timetable=instance).delete()

        serializer.save()

        day\_of\_week\_map = {'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3, 'Friday': 4, 'Saturday': 5}

        start\_date = instance.semester\_start\_date

        end\_date = instance.semester\_end\_date

        current\_date = start\_date

        target\_day = day\_of\_week\_map[instance.day\_of\_week]

        while current\_date <= end\_date:

            if current\_date.weekday() == target\_day:

                Session.objects.get\_or\_create(

                    timetable=instance,

                    date=current\_date,

                    defaults={'status': 'Scheduled'}

                )

            current\_date += timedelta(days=1)

    def destroy(self, request, \*args, \*\*kwargs):

        instance = self.get\_object()

        Session.objects.filter(timetable=instance).delete()

        self.perform\_destroy(instance)

        return Response(status=status.HTTP\_204\_NO\_CONTENT)

'''

class TimetableViewSet(viewsets.ModelViewSet):

    queryset = Timetable.objects.all()

    permission\_classes = [IsAuthenticated]

    def get\_serializer\_class(self):

        if self.action in ['create']:

            return TimetableCreateSerializer

        return TimetableSerializer  # Includes semester\_start\_date, semester\_end\_date for updates

    def get\_queryset(self):

        return Timetable.objects.filter(teacher\_\_user=self.request.user)

    def perform\_create(self, serializer):

        validated\_data = serializer.validated\_data

        teacher = validated\_data['teacher']

        daily\_schedules = validated\_data['daily\_schedules']

        section = validated\_data['section']

        if not daily\_schedules:

            raise ValidationError("At least one daily schedule is required.")

        day\_of\_week\_map = {'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3, 'Friday': 4, 'Saturday': 5}

        created\_timetables = []

        default\_start = datetime.now().date()

        default\_end = default\_start + timedelta(days=180)  # 6 months default

        for schedule in daily\_schedules:

            timetable = Timetable.objects.create(

                section=section,

                teacher=teacher,

                subject=Subject.objects.get(id=schedule['subject']),

                day\_of\_week=schedule['day\_of\_week'],

                start\_time=schedule['start\_time'],

                semester\_start\_date=default\_start,

                semester\_end\_date=default\_end

            )

            created\_timetables.append(timetable)

            start\_date = default\_start

            end\_date = default\_end

            current\_date = start\_date

            target\_day = day\_of\_week\_map[timetable.day\_of\_week]

            while current\_date <= end\_date:

                if current\_date.weekday() == target\_day:

                    Session.objects.get\_or\_create(

                        timetable=timetable,

                        date=current\_date,

                        defaults={'status': 'Scheduled'}

                    )

                current\_date += timedelta(days=1)

        timetable\_serializer = TimetableSerializer(created\_timetables, many=True)

        return Response(timetable\_serializer.data, status=status.HTTP\_201\_CREATED)

class TeacherViewSet(viewsets.ReadOnlyModelViewSet):

    queryset = Teacher.objects.all()

    serializer\_class = TeacherSerializer

    permission\_classes = [IsAuthenticated]

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def teacher\_info(request):

    try:

        teacher = Teacher.objects.get(user=request.user)

        token = request.headers.get('Authorization', '').split(' ')[1]

        decoded = jwt.decode(token, options={"verify\_signature": False})

        return Response({

            'name': f"{teacher.first\_name} {teacher.last\_name}",

            'last\_login': request.user.last\_login.isoformat() if request.user.last\_login else None,

            'token\_expiry': datetime.fromtimestamp(decoded['exp']).isoformat() if 'exp' in decoded else 'Unknown',

            'is\_admin': teacher.is\_admin

        })

    except Teacher.DoesNotExist:

        return Response({'detail': 'Teacher not found'}, status=status.HTTP\_404\_NOT\_FOUND)

class TeacherCalendarView(generics.ListAPIView):

    serializer\_class = SessionSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        teacher = Teacher.objects.get(user=self.request.user)

        queryset = Session.objects.filter(timetable\_\_teacher=teacher).order\_by('date')

        section\_id = self.request.query\_params.get('section\_id')

        if section\_id:

            queryset = queryset.filter(timetable\_\_section\_id=section\_id)

        return queryset

class MarkAttendanceView(generics.GenericAPIView):

    serializer\_class = AttendanceSerializer

    permission\_classes = [IsAuthenticated]

    def get(self, request, session\_id):

        try:

            session = Session.objects.get(id=session\_id)

            teacher = Teacher.objects.get(user=request.user)

            if session.timetable.teacher != teacher:

                return Response({"error": "Not authorized to mark this session"}, status=status.HTTP\_403\_FORBIDDEN)

            students = Student.objects.filter(section=session.timetable.section)

            existing\_attendance = Attendance.objects.filter(session=session).select\_related('student')

            attendance\_data = [

                {'student\_id': a.student.id, 'status': a.status} for a in existing\_attendance

            ] if existing\_attendance.exists() else []

            return Response({

                "session": SessionSerializer(session).data,

                "students": StudentSerializer(students, many=True).data,

                "attendance": attendance\_data

            }, status=status.HTTP\_200\_OK)

        except Session.DoesNotExist:

            logger.error(f"Session {session\_id} not found")

            return Response({"error": "Session not found"}, status=status.HTTP\_404\_NOT\_FOUND)

        except Teacher.DoesNotExist:

            logger.error(f"Teacher not found for user: {request.user}")

            return Response({"error": "Teacher not found"}, status=status.HTTP\_404\_NOT\_FOUND)

        except Exception as e:

            logger.error(f"Error in MarkAttendanceView GET: {str(e)}")

            return Response({"error": str(e)}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

    def post(self, request, session\_id):

        try:

            session = Session.objects.get(id=session\_id)

            teacher = Teacher.objects.get(user=request.user)

            if session.timetable.teacher != teacher:

                return Response({"error": "Not authorized to mark this session"}, status=status.HTTP\_403\_FORBIDDEN)

            attendance\_data = request.data.get('attendance', [])

            if not attendance\_data:

                return Response({"error": "Attendance data required"}, status=status.HTTP\_400\_BAD\_REQUEST)

            for entry in attendance\_data:

                student = Student.objects.get(id=entry['student\_id'])

                if student.section != session.timetable.section:

                    continue

                Attendance.objects.update\_or\_create(

                    student=student,

                    session=session,

                    defaults={'status': entry['status'], 'recorded\_by': teacher}

                )

            session.status = 'Completed'

            session.save()

            return Response({"message": "Attendance marked/updated successfully"}, status=status.HTTP\_200\_OK)

        except Session.DoesNotExist:

            return Response({"error": "Session not found"}, status=status.HTTP\_404\_NOT\_FOUND)

        except Student.DoesNotExist:

            return Response({"error": "Invalid student ID"}, status=status.HTTP\_400\_BAD\_REQUEST)

        except Exception as e:

            logger.error(f"Error in MarkAttendanceView POST: {str(e)}")

            return Response({"error": str(e)}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

class HolidayListCreateView(generics.ListCreateAPIView):

    queryset = CalendarException.objects.all()

    serializer\_class = CalendarExceptionSerializer

    permission\_classes = [IsAuthenticated]

    def perform\_create(self, serializer):

        holiday = serializer.save()

        Session.objects.filter(date=holiday.date).update(status='Cancelled')

class AttendanceStatsView(generics.GenericAPIView):

    permission\_classes = [IsAuthenticated]

    def get(self, request, roll\_number):

        try:

            student = Student.objects.get(roll\_number=roll\_number)

            section = student.section

            sessions = Session.objects.filter(

                timetable\_\_section=section, status='Completed'

            )

            total\_sessions = sessions.count()

            attendance = Attendance.objects.filter(

                student=student, session\_\_in=sessions

            ).aggregate(

                present=Count('id', filter=Q(status='Present')),

                absent=Count('id', filter=Q(status='Absent'))

            )

            monthly\_stats = Attendance.objects.filter(

                student=student, session\_\_in=sessions

            ).extra(

                select={'month': "EXTRACT(MONTH FROM session\_date)"}

            ).values('month').annotate(

                present=Count('id', filter=Q(status='Present')),

                absent=Count('id', filter=Q(status='Absent'))

            )

            attendance\_list = [

                'P' if a.status == 'Present' else 'A'

                for a in Attendance.objects.filter(student=student, session\_\_in=sessions).order\_by('session\_\_date')

            ]

            return Response({

                'student': StudentSerializer(student).data,

                'total\_sessions': total\_sessions,

                'present': attendance['present'] or 0,

                'absent': attendance['absent'] or 0,

                'percentage': (attendance['present'] / total\_sessions \* 100) if total\_sessions > 0 else 0,

                'monthly\_stats': list(monthly\_stats),

                'consolidated': ' '.join(attendance\_list[:5])

            })

        except Student.DoesNotExist:

            return Response({"error": "Student not found"}, status=status.HTTP\_404\_NOT\_FOUND)

class ClassHourlyStatsView(generics.GenericAPIView):

    permission\_classes = [IsAuthenticated]

    def get(self, request, session\_id):

        try:

            session = Session.objects.get(id=session\_id)

            if session.timetable.teacher != Teacher.objects.get(user=request.user):

                return Response({"error": "Not authorized"}, status=status.HTTP\_403\_FORBIDDEN)

            attendance = Attendance.objects.filter(session=session).aggregate(

                present=Count('id', filter=Q(status='Present')),

                absent=Count('id', filter=Q(status='Absent'))

            )

            total\_students = Student.objects.filter(section=session.timetable.section).count()

            return Response({

                'session': SessionSerializer(session).data,

                'present': attendance['present'] or 0,

                'absent': attendance['absent'] or 0,

                'total\_students': total\_students

            })

        except Session.DoesNotExist:

            return Response({"error": "Session not found"}, status=status.HTTP\_404\_NOT\_FOUND)

class SectionViewSet(viewsets.ReadOnlyModelViewSet):

    queryset = Section.objects.all()

    serializer\_class = SectionSerializer #serializers.ModelSerializer(Section, fields='\_\_all\_\_')

    permission\_classes = [IsAuthenticated]

class SubjectViewSet(viewsets.ModelViewSet):

    queryset = Subject.objects.all()

    serializer\_class = SubjectSerializer

    permission\_classes = [IsAuthenticated]

logger.debug('SubjectsForSectionView loaded - TEST DEPLOYMENT')

'''

class SubjectsForSectionView(APIView):

    permission\_classes = [IsAuthenticated]

    def get(self, request):

        logger.debug("TEST: Endpoint hit")

        section\_id = request.query\_params.get('section\_id')

        semester = request.query\_params.get('semester')

        logger.debug(f"TEST: Request received: section\_id={section\_id}, semester={semester}")

        if not section\_id or not semester:

            logger.warning("TEST: Missing parameters")

            return Response({"error": "section\_id and semester are required"}, status=status.HTTP\_400\_BAD\_REQUEST)

        try:

            section = Section.objects.get(id=int(section\_id))

            semester\_int = int(semester)

            logger.debug(f"TEST: Section: {section}, Program: {section.program}, Year: {section.year}, Semester: {semester\_int}")

        except (ValueError, Section.DoesNotExist):

            logger.error(f"TEST: Invalid input: section\_id={section\_id}, semester={semester}")

            return Response({"error": "Invalid section\_id or semester"}, status=status.HTTP\_400\_BAD\_REQUEST)

        program\_duration = section.program.duration\_years \* 2

        start\_semester = (section.year - 1) \* 2 + 1

        end\_semester = min(section.year \* 2, program\_duration)

        logger.debug(f"TEST: Validation range: {start\_semester} <= {semester\_int} <= {end\_semester}")

        if semester\_int < start\_semester or semester\_int > end\_semester:

            logger.warning(f"TEST: Semester {semester\_int} out of range for {section}")

            return Response({"error": f"Semester {semester\_int} is not valid for {section} (valid range: {start\_semester}-{end\_semester})"}, status=status.HTTP\_400\_BAD\_REQUEST)

        subjects = Subject.objects.filter(semester=semester\_int)

        logger.debug(f"TEST: Subjects: {list(subjects.values('id', 'name', 'semester'))}")

        serializer = SubjectSerializer(subjects, many=True)

        return Response(serializer.data, status=status.HTTP\_200\_OK)

'''

class SubjectsForSectionView(APIView):

    permission\_classes = [IsAuthenticated]

    def get(self, request):

        section\_id = request.query\_params.get('section\_id')

        semester = request.query\_params.get('semester')

        logger.debug(f"Request: section\_id={section\_id}, semester={semester}")

        if not section\_id or not semester:

            return Response({"error": "section\_id and semester are required"}, status=status.HTTP\_400\_BAD\_REQUEST)

        try:

            section = Section.objects.get(id=int(section\_id))

            semester\_int = int(semester)

            logger.debug(f"Section: {section}, Semester: {semester\_int}")

        except (ValueError, Section.DoesNotExist):

            logger.error(f"Invalid input: section\_id={section\_id}, semester={semester}")

            return Response({"error": "Invalid section\_id or semester"}, status=status.HTTP\_400\_BAD\_REQUEST)

        subjects = Subject.objects.filter(semester=semester\_int)

        logger.debug(f"Subjects: {list(subjects.values('id', 'name', 'semester'))}")

        if not subjects.exists():

            return Response({"message": "No subjects found", "data": []}, status=status.HTTP\_200\_OK)

        serializer = SubjectSerializer(subjects, many=True)

        return Response(serializer.data, status=status.HTTP\_200\_OK)

'''

# @api\_view(['GET'])

# @permission\_classes([IsAuthenticated])

# def get\_sections(request):

    try:

        teacher = Teacher.objects.get(user=request.user)

        # Fetch sections where the teacher is assigned via timetables, distinct

        sections = Section.objects.filter(timetable\_\_teacher=teacher).distinct()

        serializer = SectionSerializer(sections, many=True)

        # Enhance data with semester context (optional, if semester is stored elsewhere)

        return Response(serializer.data)

    except Teacher.DoesNotExist:

        logger.error(f"Teacher not found for user: {request.user}")

        return Response({"error": "Teacher not found"}, status=status.HTTP\_404\_NOT\_FOUND)

    except Exception as e:

        logger.error(f"Error in get\_sections: {str(e)}")

        return Response({"error": str(e)}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

'''

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def get\_sections(request):

    try:

        # Fetch all sections (not just those tied to the teacher)

        sections = Section.objects.all()

        serializer = SectionSerializer(sections, many=True)

        return Response(serializer.data)

    except Exception as e:

        logger.error(f"Error in get\_sections: {str(e)}")

        return Response({"error": str(e)}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def get\_subjects\_for\_section(request):

    section\_id = request.query\_params.get('section\_id')

    semester\_start\_date = request.query\_params.get('semester\_start\_date')

    if not section\_id or not semester\_start\_date:

        return Response({"error": "Section ID and semester start date are required"}, status=status.HTTP\_400\_BAD\_REQUEST)

    try:

        section = Section.objects.get(id=section\_id)

        start\_date = datetime.strptime(semester\_start\_date, '%Y-%m-%d').date()

        # Infer semester: Jan-Jun = odd (1, 3, 5), Jul-Dec = even (2, 4, 6)

        semester = (section.year \* 2 - 1) if start\_date.month <= 6 else (section.year \* 2)

        # Filter subjects by semester and optionally by section-specific logic if Subject model has a section relation

        subjects = Subject.objects.filter(semester=semester)

        serializer = SubjectSerializer(subjects, many=True)

        return Response(serializer.data)

    except Section.DoesNotExist:

        return Response({"error": "Section not found"}, status=status.HTTP\_404\_NOT\_FOUND)

    except Exception as e:

        logger.error(f"Error in get\_subjects\_for\_section: {str(e)}")

        return Response({"error": str(e)}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def get\_subjects(request):

    subjects = Subject.objects.all()

    serializer = SubjectSerializer(subjects, many=True)

    return Response(serializer.data)

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def get\_time\_slots(request):

    time\_slots = [slot[0] for slot in Timetable.LECTURE\_SLOTS]

    return Response(time\_slots)

>>>> urls.py ->

from django.urls import path,include

from . import views

from rest\_framework.routers import DefaultRouter

router = DefaultRouter()

router.register(r'teachers', views.TeacherViewSet)

router.register(r'students', views.StudentViewSet)

router.register(r'programs' , views.ProgramViewSet)

router.register(r'subjects', views.SubjectViewSet)

router.register(r'timetables', views.TimetableViewSet)

router.register(r'sessions',views.SessionViewSet)

router.register(r'admin-attendance-stats', views.AdminAttendanceStatsView , basename= 'admin-attendance-stats')

urlpatterns = [

    # path('', views.home , name = 'home')

    path('', include(router.urls)),

    path('calendar/', views.TeacherCalendarView.as\_view(), name='teacher\_calendar'),

    path('mark-attendance/<int:session\_id>/', views.MarkAttendanceView.as\_view(), name='mark\_attendance'),

    path('holidays/', views.HolidayListCreateView.as\_view(), name='holiday\_list\_create'),

    path('attendance-stats/<str:roll\_number>/', views.AttendanceStatsView.as\_view(), name='attendance\_stats'),

    path('teacher-info/', views.teacher\_info, name='teacher\_info'),

    path('teacher-attendance-stats/', views.TeacherAttendanceStatsView.as\_view(), name='teacher\_attendance\_stats'),

    path('hourly-stats/<int:session\_id>/', views.ClassHourlyStatsView.as\_view(), name='hourly\_stats'),

    path('sections/', views.get\_sections, name='get\_sections'),

    path('subjects/', views.get\_subjects, name='get\_subjects'),

    path('time-slots/', views.get\_time\_slots, name='get\_time\_slots'),

    path('subjects-for-section/', views.get\_subjects\_for\_section, name='get\_subjects\_for\_section'),

    ]

>>>> permissions.py ->

from rest\_framework.permissions import BasePermission

from app.models import Teacher

class IsAdmin(BasePermission):

    def has\_permission(self,request,view):

        if not request.user.is\_authenticated:

            return False

        try:

            teacher = Teacher.objects.get(user = request.user)

            return teacher.is\_admin

        except Teacher.DoesNotExist:

            return False

>>>>> programme\_attendance.urls.py ->

from django.contrib import admin

from django.urls import path , include

from rest\_framework\_simplejwt.views import TokenObtainPairView , TokenRefreshView

urlpatterns = [

    path("admin/", admin.site.urls),

    path('api/',include('app.urls')),

    path('api/token/',TokenObtainPairView.as\_view() , name = 'token\_obtain\_pair'),

    path('api/token?refresh/',TokenRefreshView.as\_view() , name = 'token\_refresh')

]

Frontend >>>> for forendend we used react for stylying we are using tailwind css

>>> App.jsx ->

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Login from './components/Login';

import Calendar from './components/Calendar';

import AttendanceForm from './components/AttendanceForm';

import TimetableForm from './components/TimetableForm';

import AttendanceStats from './components/AttendanceStats';

import AdminDashboard from './components/AdminDashboard';

import AdminCRUD from './components/AdminCRUD';

import AdminAttendanceStats from './components/AdminAttendanceStats';

import ProfileIcon from './components/ProfileIcon';

function App() {

  return (

    <Router>

      <div className="min-h-screen bg-gray-100">

        <nav className="bg-blue-600 p-4 flex justify-between items-center">

          <h1 className="text-white text-lg font-bold">Attendance System</h1>

          <ProfileIcon />

        </nav>

        <Routes>

          <Route path="/" element={<Login />} />

          <Route path="/calendar" element={<Calendar />} />

          <Route path="/attendance/:sessionId" element={<AttendanceForm onClose={() => window.history.back()} />} />

          <Route path="/timetable" element={<TimetableForm onClose={() => window.history.back()} />} />

          <Route path="/attendance-stats" element={<AttendanceStats />} />

          <Route path="/admin" element={<AdminDashboard />} />

          <Route path="/admin/teachers" element={<AdminCRUD resource="teachers" />} />

          <Route path="/admin/students" element={<AdminCRUD resource="students" />} />

          <Route path="/admin/programs" element={<AdminCRUD resource="programs" />} />

          <Route path="/admin/subjects" element={<AdminCRUD resource="subjects" />} />

          <Route path="/admin/timetables" element={<AdminCRUD resource="timetables" />} />

          <Route path="/admin/sessions" element={<AdminCRUD resource="sessions" />} />

          <Route path="/admin/attendance-stats" element={<AdminAttendanceStats />} />

        </Routes>

      </div>

    </Router>

  );

}

export default App;

/\*

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Login from './components/Login';

import Calendar from './components/Calendar';

import AttendanceForm from './components/AttendanceForm';

import TimetableForm from './components/TimetableForm';

import AttendanceStats from './components/AttendanceStats';

import ProfileIcon from './components/ProfileIcon';

function App() {

  return (

    <Router>

      <div className="min-h-screen bg-gray-100">

        <nav className="bg-blue-600 p-4 flex justify-between items-center">

          <h1> Manikchand Pahade Law College, Aurangabad  </h1>

          <h2 className="text-white text-lg font-bold">Attendance System</h2>

          <ProfileIcon />

        </nav>

        <Routes>

          <Route path="/" element={<Login />} />

          <Route path="/calendar" element={<Calendar />} />

          <Route path="/attendance/:sessionId" element={<AttendanceForm onClose={() => window.history.back()} />} />

          <Route path="/timetable" element={<TimetableForm onClose={() => window.history.back()} />} />

          <Route path="/attendance-stats" element={<AttendanceStats />} />

        </Routes>

      </div>

    </Router>

  );

}

export default App;

\*/

>>>> package.json ->

{

  "name": "frontend",

  "private": true,

  "version": "0.0.0",

  "type": "module",

  "scripts": {

    "dev": "vite",

    "build": "vite build",

    "lint": "eslint .",

    "preview": "vite preview"

  },

  "dependencies": {

    "@fullcalendar/daygrid": "^6.1.15",

    "@fullcalendar/react": "^6.1.15",

    "@fullcalendar/timegrid": "^6.1.15",

    "@tailwindcss/vite": "^4.0.16",

    "axios": "^1.8.4",

    "frontend": "file:",

    "react": "^19.0.0",

    "react-dom": "^19.0.0",

    "react-router-dom": "^7.4.0",

    "tailwindcss": "^4.0.16"

  },

  "devDependencies": {

    "@eslint/js": "^9.21.0",

    "@types/react": "^19.0.10",

    "@types/react-dom": "^19.0.4",

    "@vitejs/plugin-react": "^4.3.4",

    "eslint": "^9.21.0",

    "eslint-plugin-react-hooks": "^5.1.0",

    "eslint-plugin-react-refresh": "^0.4.19",

    "globals": "^15.15.0",

    "vite": "^6.2.0"

  }

}

>> AdminAttendanceStats.jsx ->

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function AdminAttendanceStats() {

  const [stats, setStats] = useState([]);

  const [period, setPeriod] = useState('semester');

  const [section, setSection] = useState('');

  const [date, setDate] = useState('');

  const [error, setError] = useState('');

  const [startDate, setStartDate] = useState('');

  const [endDate, setEndDate] = useState('');

  useEffect(() => {

    const fetchStats = async () => {

      const token = localStorage.getItem('access\_token');

      if (!token) {

        setError('Please log in first');

        return;

      }

      let url = `http://localhost:8000/api/admin-attendance-stats/?period=${period}`;

      if (section) url += `&section=${section}`;

      if (period === 'daily' && date) url += `&date=${date}`;

      try {

        const response = await axios.get(url, {

          headers: { Authorization: `Bearer ${token}` },

        });

        setStats(response.data.stats);

        setStartDate(response.data.start\_date);

        setEndDate(response.data.end\_date);

        setError('');

      } catch (err) {

        console.error('Failed to fetch stats:', err);

        setError(err.response?.data?.error || 'Failed to load attendance stats');

      }

    };

    fetchStats();

  }, [period, section, date]);

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">All Attendance Statistics</h2>

      {error && <p className="text-red-600 mb-4">{error}</p>}

      <div className="mb-4 flex space-x-4">

        <div>

          <label className="mr-2 text-gray-800">View by:</label>

          <select

            value={period}

            onChange={(e) => setPeriod(e.target.value)}

            className="p-2 border rounded-md bg-white text-gray-800 focus:outline-none focus:ring-2 focus:ring-blue-500"

          >

            <option value="daily">Daily</option>

            <option value="weekly">Weekly</option>

            <option value="monthly">Monthly</option>

            <option value="semester">Semester</option>

          </select>

        </div>

        <div>

          <label className="mr-2 text-gray-800">Section ID:</label>

          <input

            type="number"

            value={section}

            onChange={(e) => setSection(e.target.value)}

            className="p-2 border rounded-md text-gray-800"

            placeholder="Optional"

          />

        </div>

        {period === 'daily' && (

          <div>

            <label className="mr-2 text-gray-800">Date:</label>

            <input

              type="date"

              value={date}

              onChange={(e) => setDate(e.target.value)}

              className="p-2 border rounded-md text-gray-800"

              required

            />

          </div>

        )}

      </div>

      <p className="text-gray-800 mb-4">Period: {startDate} to {endDate}</p>

      {stats.length > 0 ? (

        <div className="overflow-x-auto">

          <table className="w-full bg-white rounded-lg shadow-md">

            <thead>

              <tr className="bg-blue-600 text-white">

                <th className="p-2 text-left">Student Name</th>

                <th className="p-2 text-left">Roll Number</th>

                <th className="p-2 text-left">Total Sessions</th>

                <th className="p-2 text-left">Present</th>

                <th className="p-2 text-left">Absent</th>

                <th className="p-2 text-left">Attendance %</th>

                <th className="p-2 text-left">Recorded By</th>

              </tr>

            </thead>

            <tbody>

              {stats.map((stat) => (

                <tr key={stat.student\_id} className="border-t">

                  <td className="p-2 text-gray-800">{stat.name}</td>

                  <td className="p-2 text-gray-800">{stat.roll\_number}</td>

                  <td className="p-2 text-gray-800">{stat.total\_sessions}</td>

                  <td className="p-2 text-gray-800">{stat.present}</td>

                  <td className="p-2 text-gray-800">{stat.absent}</td>

                  <td className="p-2 text-gray-800">{stat.attendance\_percentage}%</td>

                  <td className="p-2 text-gray-800">{stat.recorded\_by}</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      ) : (

        <p className="text-gray-600">No attendance stats available for this period.</p>

      )}

    </div>

  );

}

export default AdminAttendanceStats;

>>>> AdminCRUD.jsx ->

import React, { useState, useEffect } from "react";

import axios from "axios";

import { useParams, useNavigate } from "react-router-dom";

function AdminCRUD({ resource }) {

  const [items, setItems] = useState([]);

  const [error, setError] = useState("");

  const [formData, setFormData] = useState({});

  const [editingId, setEditingId] = useState(null);

  const navigate = useNavigate();

  useEffect(() => {

    const fetchItems = async () => {

      const token = localStorage.getItem("access\_token");

      try {

        const response = await axios.get(

          `http://localhost:8000/api/${resource}/`,

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

        setItems(response.data);

      } catch (err) {

        setError("Failed to load data");

      }

    };

    fetchItems();

  }, [resource]);

  const handleSubmit = async (e) => {

    e.preventDefault();

    const token = localStorage.getItem("access\_token");

    try {

      if (editingId) {

        await axios.put(

          `http://localhost:8000/api/${resource}/${editingId}/`,

          formData,

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

      } else {

        await axios.post(`http://localhost:8000/api/${resource}/`, formData, {

          headers: { Authorization: `Bearer ${token}` },

        });

      }

      setFormData({});

      setEditingId(null);

      const response = await axios.get(

        `http://localhost:8000/api/${resource}/`,

        {

          headers: { Authorization: `Bearer ${token}` },

        }

      );

      setItems(response.data);

    } catch (err) {

      setError("Failed to save data");

    }

  };

  const handleEdit = (item) => {

    setFormData(item);

    setEditingId(item.id);

  };

  const handleDelete = async (id) => {

    const token = localStorage.getItem("access\_token");

    try {

      await axios.delete(`http://localhost:8000/api/${resource}/${id}/`, {

        headers: { Authorization: `Bearer ${token}` },

      });

      setItems(items.filter((item) => item.id !== id));

    } catch (err) {

      setError("Failed to delete item");

    }

  };

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">

        Manage {resource.charAt(0).toUpperCase() + resource.slice(1)}

      </h2>

      {error && <p className="text-red-600 mb-4">{error}</p>}

      <form

        onSubmit={handleSubmit}

        className="mb-6 bg-white p-4 rounded-lg shadow-md"

      >

        {Object.keys(formData).map(

          (key) =>

            key !== "id" && (

              <div key={key} className="mb-2">

                <label className="block text-gray-800">

                  {key.replace("\_", " ")}:

                </label>

                <input

                  type="text"

                  value={formData[key] || ""}

                  onChange={(e) =>

                    setFormData({ ...formData, [key]: e.target.value })

                  }

                  className="w-full p-2 border rounded-md text-gray-800"

                />

              </div>

            )

        )}

        <button

          type="submit"

          className="bg-blue-600 text-white py-2 px-4 rounded-md hover:bg-blue-700"

        >

          {editingId ? "Update" : "Create"}

        </button>

        {editingId && (

          <button

            type="button"

            onClick={() => {

              setFormData({});

              setEditingId(null);

            }}

            className="ml-2 bg-gray-600 text-white py-2 px-4 rounded-md hover:bg-gray-700"

          >

            Cancel

          </button>

        )}

      </form>

      <table className="w-full bg-white rounded-lg shadow-md">

        <thead>

          <tr className="bg-blue-600 text-white">

            {items.length > 0 &&

              Object.keys(items[0]).map(

                (key) =>

                  key !== "id" && (

                    <th key={key} className="p-2 text-left">

                      {key.replace("\_", " ")}

                    </th>

                  )

              )}

            <th className="p-2 text-left">Actions</th>

          </tr>

        </thead>

        <tbody>

          {items.map((item) => (

            <tr key={item.id} className="border-t">

              {Object.keys(item).map(

                (key) =>

                  key !== "id" && (

                    <td key={key} className="p-2 text-gray-800">

                      {JSON.stringify(item[key])}

                    </td>

                  )

              )}

              <td className="p-2">

                <button

                  onClick={() => handleEdit(item)}

                  className="text-blue-600 hover:underline mr-2"

                >

                  Edit

                </button>

                <button

                  onClick={() => handleDelete(item.id)}

                  className="text-red-600 hover:underline"

                >

                  Delete

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

}

export default AdminCRUD;

>> AdminDashboard.jsx ->

import React, { useState, useEffect } from "react";

import axios from "axios";

import { Link, useNavigate } from "react-router-dom";

function AdminDashboard() {

  const [isAdmin, setIsAdmin] = useState(false);

  const [error, setError] = useState("");

  const navigate = useNavigate();

  useEffect(() => {

    const checkAdmin = async () => {

      const token = localStorage.getItem("access\_token");

      if (!token) {

        setError("Please log in first");

        navigate("/");

        return;

      }

      try {

        const response = await axios.get(

          "http://localhost:8000/api/teacher-info/",

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

        setIsAdmin(response.data.is\_admin);

        if (!response.data.is\_admin) {

          setError("You are not authorized to access this page");

          navigate("/calendar");

        }

      } catch (err) {

        setError("Failed to verify admin status");

        navigate("/");

      }

    };

    checkAdmin();

  }, [navigate]);

  if (error) {

    return (

      <div className="p-6 bg-gray-100 min-h-screen text-red-600">{error}</div>

    );

  }

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">Admin Dashboard</h2>

      {isAdmin && (

        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Teachers

            </h3>

            <Link

              to="/admin/teachers"

              className="text-blue-600 hover:underline"

            >

              View/Edit Teachers

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Students

            </h3>

            <Link

              to="/admin/students"

              className="text-blue-600 hover:underline"

            >

              View/Edit Students

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Programs

            </h3>

            <Link

              to="/admin/programs"

              className="text-blue-600 hover:underline"

            >

              View/Edit Programs

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Subjects

            </h3>

            <Link

              to="/admin/subjects"

              className="text-blue-600 hover:underline"

            >

              View/Edit Subjects

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Timetables

            </h3>

            <Link to="/timetable" className="text-blue-600 hover:underline">

              Create Timetable

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Sessions

            </h3>

            <Link

              to="/admin/sessions"

              className="text-blue-600 hover:underline"

            >

              View/Edit Sessions

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              View All Stats

            </h3>

            <Link

              to="/admin/attendance-stats"

              className="text-blue-600 hover:underline"

            >

              View Attendance Stats

            </Link>

          </div>

        </div>

      )}

    </div>

  );

}

export default AdminDashboard;

>>> AttendanceForm.jsx ->

import React, { useState, useEffect } from "react";

import axios from "axios";

import { Link, useNavigate } from "react-router-dom";

function AdminDashboard() {

  const [isAdmin, setIsAdmin] = useState(false);

  const [error, setError] = useState("");

  const navigate = useNavigate();

  useEffect(() => {

    const checkAdmin = async () => {

      const token = localStorage.getItem("access\_token");

      if (!token) {

        setError("Please log in first");

        navigate("/");

        return;

      }

      try {

        const response = await axios.get(

          "http://localhost:8000/api/teacher-info/",

          {

            headers: { Authorization: `Bearer ${token}` },

          }

        );

        setIsAdmin(response.data.is\_admin);

        if (!response.data.is\_admin) {

          setError("You are not authorized to access this page");

          navigate("/calendar");

        }

      } catch (err) {

        setError("Failed to verify admin status");

        navigate("/");

      }

    };

    checkAdmin();

  }, [navigate]);

  if (error) {

    return (

      <div className="p-6 bg-gray-100 min-h-screen text-red-600">{error}</div>

    );

  }

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">Admin Dashboard</h2>

      {isAdmin && (

        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Teachers

            </h3>

            <Link

              to="/admin/teachers"

              className="text-blue-600 hover:underline"

            >

              View/Edit Teachers

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Students

            </h3>

            <Link

              to="/admin/students"

              className="text-blue-600 hover:underline"

            >

              View/Edit Students

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Programs

            </h3>

            <Link

              to="/admin/programs"

              className="text-blue-600 hover:underline"

            >

              View/Edit Programs

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Subjects

            </h3>

            <Link

              to="/admin/subjects"

              className="text-blue-600 hover:underline"

            >

              View/Edit Subjects

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Timetables

            </h3>

            <Link to="/timetable" className="text-blue-600 hover:underline">

              Create Timetable

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              Manage Sessions

            </h3>

            <Link

              to="/admin/sessions"

              className="text-blue-600 hover:underline"

            >

              View/Edit Sessions

            </Link>

          </div>

          <div className="bg-white p-4 rounded-lg shadow-md">

            <h3 className="text-lg font-semibold mb-2 text-gray-800">

              View All Stats

            </h3>

            <Link

              to="/admin/attendance-stats"

              className="text-blue-600 hover:underline"

            >

              View Attendance Stats

            </Link>

          </div>

        </div>

      )}

    </div>

  );

}

export default AdminDashboard;

>>>> AttendanceStats.jsx ->

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function AttendanceStats() {

  const [stats, setStats] = useState([]);

  const [period, setPeriod] = useState('semester');

  const [startDate, setStartDate] = useState('');

  const [endDate, setEndDate] = useState('');

  const [customRange, setCustomRange] = useState(false);

  const [error, setError] = useState('');

  const [responseStartDate, setResponseStartDate] = useState('');

  const [responseEndDate, setResponseEndDate] = useState('');

  useEffect(() => {

    const fetchStats = async () => {

      const token = localStorage.getItem('access\_token');

      if (!token) {

        setError('Please log in first');

        return;

      }

      let url = `http://localhost:8000/api/teacher-attendance-stats/?period=${period}`;

      if (customRange && startDate && endDate) {

        url += `&start\_date=${startDate}&end\_date=${endDate}`;

      }

      try {

        const response = await axios.get(url, {

          headers: { Authorization: `Bearer ${token}` },

        });

        setStats(response.data.stats);

        setResponseStartDate(response.data.start\_date);

        setResponseEndDate(response.data.end\_date);

        setError('');

      } catch (err) {

        console.error('Failed to fetch stats:', err);

        setError(err.response?.data?.error || 'Failed to load attendance stats');

      }

    };

    fetchStats();

  }, [period, startDate, endDate, customRange]);

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">Attendance Statistics</h2>

      {error && <p className="text-red-600 mb-4">{error}</p>}

      <div className="mb-4 flex flex-col space-y-4">

        <div className="flex space-x-4">

          <label className="mr-2 text-gray-800">View by:</label>

          <select

            value={period}

            onChange={(e) => {

              setPeriod(e.target.value);

              setCustomRange(false); // Reset custom range when changing period

            }}

            className="p-2 border rounded-md bg-white text-gray-800 focus:outline-none focus:ring-2 focus:ring-blue-500"

          >

            <option value="weekly">Weekly</option>

            <option value="monthly">Monthly</option>

            <option value="semester">Semester</option>

            <option value="custom">Custom Range</option>

          </select>

        </div>

        {period === 'custom' && (

          <div className="flex space-x-4">

            <div>

              <label className="mr-2 text-gray-800">Start Date:</label>

              <input

                type="date"

                value={startDate}

                onChange={(e) => {

                  setStartDate(e.target.value);

                  setCustomRange(true);

                }}

                className="p-2 border rounded-md text-gray-800"

              />

            </div>

            <div>

              <label className="mr-2 text-gray-800">End Date:</label>

              <input

                type="date"

                value={endDate}

                onChange={(e) => {

                  setEndDate(e.target.value);

                  setCustomRange(true);

                }}

                className="p-2 border rounded-md text-gray-800"

              />

            </div>

          </div>

        )}

      </div>

      <p className="text-gray-800 mb-4">Period: {responseStartDate} to {responseEndDate}</p>

      {stats.length > 0 ? (

        <div className="overflow-x-auto">

          <table className="w-full bg-white rounded-lg shadow-md">

            <thead>

              <tr className="bg-blue-600 text-white">

                <th className="p-2 text-left">Student Name</th>

                <th className="p-2 text-left">Roll Number</th>

                <th className="p-2 text-left">Total Sessions</th>

                <th className="p-2 text-left">Present</th>

                <th className="p-2 text-left">Absent</th>

                <th className="p-2 text-left">Attendance %</th>

              </tr>

            </thead>

            <tbody>

              {stats.map((stat) => (

                <tr key={stat.student\_id} className="border-t">

                  <td className="p-2 text-gray-800">{stat.name}</td>

                  <td className="p-2 text-gray-800">{stat.roll\_number}</td>

                  <td className="p-2 text-gray-800">{stat.total\_sessions}</td>

                  <td className="p-2 text-gray-800">{stat.present}</td>

                  <td className="p-2 text-gray-800">{stat.absent}</td>

                  <td className="p-2 text-gray-800">{stat.attendance\_percentage}%</td>

                </tr>

              ))}

            </tbody>

          </table>

        </div>

      ) : (

        <p className="text-gray-600">No attendance stats available for this period.</p>

      )}

    </div>

  );

}

export default AttendanceStats;

>>>> Calendar.jsx ->

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { Link } from 'react-router-dom';

function Calendar() {

  const [sessions, setSessions] = useState([]);

  const [timetables, setTimetables] = useState([]);

  const [error, setError] = useState('');

  const [showTimetableForm, setShowTimetableForm] = useState(false);

  const [editTimetableId, setEditTimetableId] = useState(null);

  const [sections, setSections] = useState([]);

  const [subjects, setSubjects] = useState([]);

  const [timeSlots, setTimeSlots] = useState([]);

  const [selectedSection, setSelectedSection] = useState('');

  const [selectedSemester, setSelectedSemester] = useState('');

  const [timetableData, setTimetableData] = useState({

    section: '',

    semester: '',

    daily\_schedules: [{ day\_of\_week: 'Monday', subject: '', start\_time: '08:30:00' }],

    semester\_start\_date: '',

    semester\_end\_date: '',

  });

  useEffect(() => {

    fetchSessions();

    fetchTimetables();

    fetchOptions();

  }, [selectedSection]);

  const fetchSessions = async () => {

    const token = localStorage.getItem('access\_token');

    if (!token) { setError('Please log in first'); return; }

    try {

      const url = selectedSection ? `http://localhost:8000/api/calendar/?section\_id=${selectedSection}` : 'http://localhost:8000/api/calendar/';

      const response = await axios.get(url, { headers: { Authorization: `Bearer ${token}` } });

      setSessions(response.data);

      setError('');

    } catch (err) {

      console.error('Failed to fetch sessions:', err);

      setError('Failed to load sessions');

    }

  };

  const fetchTimetables = async () => {

    const token = localStorage.getItem('access\_token');

    if (!token) { setError('Please log in first'); return; }

    try {

      const response = await axios.get('http://localhost:8000/api/timetables/', { headers: { Authorization: `Bearer ${token}` } });

      setTimetables(response.data);

    } catch (err) {

      console.error('Failed to fetch timetables:', err);

      setError('Failed to load timetables');

    }

  };

  const fetchOptions = async () => {

    const token = localStorage.getItem('access\_token');

    if (!token) { setError('Please log in first'); return; }

    try {

      const [sectionsRes, timeSlotsRes] = await Promise.all([

        axios.get('http://localhost:8000/api/sections/', { headers: { Authorization: `Bearer ${token}` } }),

        axios.get('http://localhost:8000/api/time-slots/', { headers: { Authorization: `Bearer ${token}` } }),

      ]);

      setSections(sectionsRes.data);

      setTimeSlots(timeSlotsRes.data);

      if (timetableData.section && timetableData.semester) {

        fetchSubjects(timetableData.section, timetableData.semester);

      }

      setError('');

    } catch (err) {

      console.error('Failed to fetch options:', err);

      setError('Failed to load options: ' + (err.response?.data?.error || 'Unknown error'));

    }

  };

  const fetchSubjects = async (sectionId, semester) => {

    const token = localStorage.getItem('access\_token');

    try {

      console.log(`Fetching subjects: section\_id=${sectionId}, semester=${semester}`);

      const response = await axios.get('http://localhost:8000/api/subjects-for-section/', {

        params: { section\_id: sectionId, semester: semester },

        headers: { Authorization: `Bearer ${token}` },

      });

      const data = Array.isArray(response.data) ? response.data : response.data.data || [];

      console.log('Subjects received:', data);

      setSubjects(data);

      setError('');

    } catch (err) {

      console.error('Failed to fetch subjects:', err.response?.data || err.message);

      setError('Failed to load subjects: ' + (err.response?.data?.error || err.response?.data?.message || 'Unknown error'));

      setSubjects([]);

    }

  };

  const handleSectionSemesterChange = (e) => {

    const [sectionId, semester] = e.target.value.split('-');

    console.log(`Fetching subjects for sectionId=${sectionId}, semester=${semester}`);

    setTimetableData({ ...timetableData, section: sectionId, semester: semester });

    setSelectedSemester(semester);

    if (sectionId && semester) {

      fetchSubjects(sectionId, semester);

    }

  };

  const handleDateChange = (field) => (e) => {

    setTimetableData({ ...timetableData, [field]: e.target.value });

  };

  const handleTimetableSubmit = async (e) => {

    e.preventDefault();

    const token = localStorage.getItem('access\_token');

    try {

      const data = {

        section: timetableData.section,

        semester: parseInt(timetableData.semester),

        daily\_schedules: timetableData.daily\_schedules,

        ...(editTimetableId ? {

          semester\_start\_date: timetableData.semester\_start\_date,

          semester\_end\_date: timetableData.semester\_end\_date

        } : {})

      };

      if (editTimetableId) {

        await axios.put(`http://localhost:8000/api/timetables/${editTimetableId}/`, {

          section: data.section,

          subject: data.daily\_schedules[0].subject,

          day\_of\_week: data.daily\_schedules[0].day\_of\_week,

          start\_time: data.daily\_schedules[0].start\_time,

          semester\_start\_date: data.semester\_start\_date,

          semester\_end\_date: data.semester\_end\_date,

        }, { headers: { Authorization: `Bearer ${token}` } });

      } else {

        await axios.post('http://localhost:8000/api/timetables/', data, { headers: { Authorization: `Bearer ${token}` } });

      }

      setShowTimetableForm(false);

      setEditTimetableId(null);

      setTimetableData({

        section: '',

        semester: '',

        daily\_schedules: [{ day\_of\_week: 'Monday', subject: '', start\_time: '08:30:00' }],

        semester\_start\_date: '',

        semester\_end\_date: '',

      });

      setSubjects([]);

      fetchSessions();

      fetchTimetables();

      setError('');

    } catch (err) {

      console.error('Failed to save timetable:', err);

      setError(err.response?.data?.detail || 'Failed to save timetable');

    }

  };

  const addSchedule = () => {

    setTimetableData({

      ...timetableData,

      daily\_schedules: [...timetableData.daily\_schedules, { day\_of\_week: 'Monday', subject: '', start\_time: '08:30:00' }],

    });

  };

  const updateSchedule = (index, field, value) => {

    const newSchedules = [...timetableData.daily\_schedules];

    newSchedules[index][field] = value;

    setTimetableData({ ...timetableData, daily\_schedules: newSchedules });

  };

  const handleEditTimetable = (timetable) => {

    setEditTimetableId(timetable.id);

    setTimetableData({

      section: timetable.section.id,

      semester: timetable.semester.toString(),

      daily\_schedules: [{

        day\_of\_week: timetable.day\_of\_week,

        subject: timetable.subject.id,

        start\_time: timetable.start\_time,

      }],

      semester\_start\_date: timetable.semester\_start\_date,

      semester\_end\_date: timetable.semester\_end\_date,

    });

    fetchSubjects(timetable.section.id, timetable.semester);

    setShowTimetableForm(true);

  };

  const handleDeleteTimetable = async (timetableId) => {

    const token = localStorage.getItem('access\_token');

    if (window.confirm('Are you sure you want to delete this timetable?')) {

      try {

        await axios.delete(`http://localhost:8000/api/timetables/${timetableId}/`, { headers: { Authorization: `Bearer ${token}` } });

        fetchTimetables();

        fetchSessions();

        setError('');

      } catch (err) {

        console.error('Failed to delete timetable:', err);

        setError('Failed to delete timetable');

      }

    }

  };

  return (

    <div className="p-6 bg-gray-100 min-h-screen">

      <h2 className="text-2xl font-bold mb-4 text-gray-800">Teacher Calendar</h2>

      {error && <p className="text-red-600 mb-4">{error}</p>}

      <div className="mb-4 flex space-x-4">

        <Link to="/attendance-stats" className="text-blue-600 hover:underline">View Attendance Statistics</Link>

        <button

          onClick={() => {

            setEditTimetableId(null);

            setTimetableData({

              section: '',

              semester: '',

              daily\_schedules: [{ day\_of\_week: 'Monday', subject: '', start\_time: '08:30:00' }],

              semester\_start\_date: '',

              semester\_end\_date: '',

            });

            setShowTimetableForm(true);

          }}

          className="bg-blue-600 text-white py-2 px-4 rounded-md hover:bg-blue-700"

        >

          Create Timetable

        </button>

      </div>

      <h3 className="text-xl font-bold mb-2 text-gray-800">Your Timetables</h3>

      {timetables.length > 0 ? (

        <ul className="space-y-4 mb-6">

          {timetables.map((timetable) => (

            <li key={timetable.id} className="p-4 bg-white rounded-lg shadow-md flex justify-between items-center">

              <div>

                <p><strong>Section:</strong> {timetable.section.name} (Year: {timetable.section.year}, Semester: {timetable.semester}, Program: {timetable.section.program})</p>

                <p><strong>Period:</strong> {timetable.semester\_start\_date} to {timetable.semester\_end\_date}</p>

                <p><strong>Schedule:</strong> {timetable.day\_of\_week}: {timetable.subject.name} at {timetable.start\_time}</p>

              </div>

              <div className="space-x-2">

                <button onClick={() => handleEditTimetable(timetable)} className="bg-yellow-600 text-white py-1 px-2 rounded-md hover:bg-yellow-700">Edit</button>

                <button onClick={() => handleDeleteTimetable(timetable.id)} className="bg-red-600 text-white py-1 px-2 rounded-md hover:bg-red-700">Delete</button>

              </div>

            </li>

          ))}

        </ul>

      ) : (

        <p className="text-gray-600 mb-6">No timetables found.</p>

      )}

      <div className="mb-4">

        <label className="block text-gray-800">Filter Sessions by Section:</label>

        <select value={selectedSection} onChange={(e) => setSelectedSection(e.target.value)} className="w-full p-2 border rounded-md bg-white text-gray-800">

          <option value="">All Sections</option>

          {sections.map((section) => (

            <option key={section.id} value={section.id}>{section.name} (Year: {section.year}, Program: {section.program})</option>

          ))}

        </select>

      </div>

      {showTimetableForm && (

        <div className="fixed top-0 left-0 w-full h-full bg-black bg-opacity-50 flex items-center justify-center z-50">

          <div className="bg-white p-6 rounded-lg shadow-lg max-w-2xl w-full max-h-[80vh] overflow-y-auto">

            <h3 className="text-xl font-bold mb-4 text-gray-800">{editTimetableId ? 'Edit Timetable' : 'Create Timetable'}</h3>

            <form onSubmit={handleTimetableSubmit}>

              {editTimetableId && (

                <div className="mb-4">

                  <label className="block text-gray-800">Semester Period:</label>

                  <div className="flex space-x-2">

                    <input

                      type="date"

                      value={timetableData.semester\_start\_date}

                      onChange={handleDateChange('semester\_start\_date')}

                      className="w-full p-2 border rounded-md text-gray-800"

                      required

                    />

                    <span className="text-gray-800">to</span>

                    <input

                      type="date"

                      value={timetableData.semester\_end\_date}

                      onChange={handleDateChange('semester\_end\_date')}

                      className="w-full p-2 border rounded-md text-gray-800"

                      required

                    />

                  </div>

                </div>

              )}

              <div className="mb-4">

                <label className="block text-gray-800">Section and Semester:</label>

                <select

                  value={timetableData.section && timetableData.semester ? `${timetableData.section}-${timetableData.semester}` : ''}

                  onChange={handleSectionSemesterChange}

                  className="w-full p-2 border rounded-md bg-white text-gray-800"

                  required

                >

                  <option value="">Select Section and Semester</option>

                  {sections.flatMap((section) =>

                    section.available\_semesters.map((semester) => (

                      <option key={`${section.id}-${semester}`} value={`${section.id}-${semester}`}>

                        {section.name} (Year: {section.year}, Semester: {semester}, Program: {section.program})

                      </option>

                    ))

                  )}

                </select>

              </div>

              {timetableData.daily\_schedules.map((schedule, index) => (

                <div key={index} className="mb-4 flex space-x-2">

                  <select

                    value={schedule.day\_of\_week}

                    onChange={(e) => updateSchedule(index, 'day\_of\_week', e.target.value)}

                    className="p-2 border rounded-md bg-white text-gray-800"

                  >

                    {['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'].map((day) => (

                      <option key={day} value={day}>{day}</option>

                    ))}

                  </select>

                  <select

                    value={schedule.subject}

                    onChange={(e) => updateSchedule(index, 'subject', e.target.value)}

                    className="p-2 border rounded-md bg-white text-gray-800"

                    required

                  >

                    <option value="">Select Subject</option>

                    {subjects.map((subject) => (

                      <option key={subject.id} value={subject.id}>{subject.name} (Semester: {subject.semester})</option>

                    ))}

                  </select>

                  <select

                    value={schedule.start\_time}

                    onChange={(e) => updateSchedule(index, 'start\_time', e.target.value)}

                    className="p-2 border rounded-md bg-white text-gray-800"

                    required

                  >

                    <option value="">Select Time</option>

                    {timeSlots.map((slot) => (

                      <option key={slot} value={slot}>{slot}</option>

                    ))}

                  </select>

                </div>

              ))}

              <button

                type="button"

                onClick={addSchedule}

                className="bg-green-600 text-white py-1 px-2 rounded-md hover:bg-green-700 mb-4"

              >

                Add Another Schedule

              </button>

              <div className="flex justify-end space-x-2">

                <button

                  type="button"

                  onClick={() => {

                    setShowTimetableForm(false);

                    setEditTimetableId(null);

                    setTimetableData({

                      section: '',

                      semester: '',

                      daily\_schedules: [{ day\_of\_week: 'Monday', subject: '', start\_time: '08:30:00' }],

                      semester\_start\_date: '',

                      semester\_end\_date: '',

                    });

                    setError('');

                  }}

                  className="bg-gray-600 text-white py-2 px-4 rounded-md hover:bg-gray-700"

                >

                  Cancel

                </button>

                <button type="submit" className="bg-blue-600 text-white py-2 px-4 rounded-md hover:bg-blue-700">

                  {editTimetableId ? 'Update Timetable' : 'Save Timetable'}

                </button>

              </div>

            </form>

          </div>

        </div>

      )}

      {sessions.length > 0 ? (

        <ul className="space-y-4">

          {sessions.map((session) => (

            <li key={session.id} className="p-4 bg-white rounded-lg shadow-md">

              <p><strong>Date:</strong> {session.date}</p>

              <p><strong>Day:</strong> {session.timetable.day\_of\_week}</p>

              <p><strong>Time:</strong> {session.timetable.start\_time}</p>

              <p><strong>Subject:</strong> {session.timetable.subject.name}</p>

              <p><strong>Section:</strong> {session.timetable.section.name} (Year: {session.timetable.section.year}, Semester: {session.timetable.semester}, Program: {session.timetable.section.program})</p>

              <p><strong>Period:</strong> {session.timetable.semester\_start\_date} to {session.timetable.semester\_end\_date}</p>

              <p><strong>Status:</strong> {session.status}</p>

              {session.status === 'Scheduled' ? (

                <Link to={`/attendance/${session.id}`} className="text-blue-600 hover:underline">Mark Attendance</Link>

              ) : (

                <Link to={`/attendance/${session.id}`} className="text-green-600 hover:underline">Update Attendance</Link>

              )}

            </li>

          ))}

        </ul>

      ) : (

        <p className="text-gray-600">No sessions found for this section.</p>

      )}

    </div>

  );

}

export default Calendar;

>>> Login.jsx ->

/\*

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

function Login() {

  const [username, setUsername] = useState('');

  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const navigate = useNavigate();

  useEffect(() => {

    if (localStorage.getItem('access\_token')) {

      navigate('/calendar');

    }

  }, [navigate]);

  const handleLogin = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://localhost:8000/api/token/', {

        username,

        password,

      });

      localStorage.setItem('access\_token', response.data.access);

      localStorage.setItem('refresh\_token', response.data.refresh);

      setError('');

      navigate('/calendar');

    } catch (err) {

      setError('Invalid credentials');

      console.error(err);

    }

  };

  return (

    <div style={{ maxWidth: '400px', margin: '50px auto', textAlign: 'center' }}>

      <h2>Teacher Login</h2>

      <form onSubmit={handleLogin}>

        <div>

          <input

            type="text"

            placeholder="Username"

            value={username}

            onChange={(e) => setUsername(e.target.value)}

            style={{ width: '100%', padding: '8px', margin: '10px 0' }}

          />

        </div>

        <div>

          <input

            type="password"

            placeholder="Password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

            style={{ width: '100%', padding: '8px', margin: '10px 0' }}

          />

        </div>

        {error && <p style={{ color: 'red' }}>{error}</p>}

        <button type="submit" style={{ padding: '10px 20px' }}>Login</button>

      </form>

    </div>

  );

}

export default Login;

\*/

import React, { useState } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

function Login() {

  const [username, setUsername] = useState('');

  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const navigate = useNavigate();

  const handleLogin = async (e) => {

    e.preventDefault();

    console.log('You have Presesed submit button')

    console.log('USERNAME IS ::',username)

    console.log('PASSWORD IS ::',password)

    // console.log(first)

    try {

      const response = await axios.post('http://127.0.0.1:8000/api/token/', {

        username,

        password,

      });

      console.log(response)

      localStorage.setItem('access\_token', response.data.access);

      localStorage.setItem('refresh\_token', response.data.refresh);

      setError('');

      navigate('/calendar');

    } catch (err) {

      setError(err.message);

      console.error(err);

    }

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://localhost:8000/api/token/', { username, password });

      localStorage.setItem('access\_token', response.data.access);

      window.location.href = '/calendar';

    } catch (err) {

      setError('Invalid credentials');

    }

  };

  return (

    <div style={{ maxWidth: '400px', margin: '50px auto', textAlign: 'center' }}>

      <h2>Teacher Login</h2>

      <form onSubmit={handleLogin}>

        <div>

          <input

            type="text"

            placeholder="Username"

            value={username}

            onChange={(e) => setUsername(e.target.value)}

            style={{ width: '100%', padding: '8px', margin: '10px 0' }}

          />

        </div>

        <div>

          <input

            type="password"

            placeholder="Password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

            style={{ width: '100%', padding: '8px', margin: '10px 0' }}

          />

        </div>

        {error && <p style={{ color: 'red' }}>{error}</p>}

        <button type="submit" style={{ padding: '10px 20px' }}>Login</button>

      </form>

    </div>

  );

}

export default Login;

>>> ProfileIcon.jsx ->

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

function ProfileIcon() {

  const [profile, setProfile] = useState(null);

  const [error, setError] = useState('');

  const [showDropdown, setShowDropdown] = useState(false);

  const navigate = useNavigate();

  useEffect(() => {

    const fetchProfile = async () => {

      const token = localStorage.getItem('access\_token');

      if (!token) {

        setError('Not logged in');

        navigate('/');

        return;

      }

      try {

        const response = await axios.get('http://localhost:8000/api/teacher-info/', {

          headers: { Authorization: `Bearer ${token}` },

        });

        setProfile(response.data);

        setError('');

      } catch (err) {

        console.error('Failed to load profile:', err);

        setError('Failed to load profile');

        navigate('/');

      }

    };

    fetchProfile();

    const checkTokenExpiry = setInterval(() => {

      const token = localStorage.getItem('access\_token');

      if (token && profile?.token\_expiry) {

        const expiry = new Date(profile.token\_expiry);

        const now = new Date();

        if (now >= expiry) {

          localStorage.removeItem('access\_token');

          setProfile(null);

          setError('Session expired');

          navigate('/');

        }

      }

    }, 1000); // Check every second

    return () => clearInterval(checkTokenExpiry); // Cleanup on unmount

  }, [navigate, profile?.token\_expiry]);

  const handleLogout = () => {

    localStorage.removeItem('access\_token');

    setProfile(null);

    setShowDropdown(false);

    navigate('/');

  };

  return (

    <div className="relative">

      <button

        onClick={() => setShowDropdown(!showDropdown)}

        className="bg-blue-500 text-white p-2 rounded-full hover:bg-blue-600 focus:outline-none"

      >

        Profile

      </button>

      {showDropdown && (

        <div className="absolute right-0 mt-2 w-64 bg-white rounded-lg shadow-lg p-4 text-gray-800">

          {error ? (

            <p className="text-red-600">{error}</p>

          ) : profile ? (

            <>

              <p><strong>Name:</strong> {profile.name}</p>

              <p><strong>Role:</strong> {profile.is\_admin ? 'Admin' : 'Teacher'}</p>

              <p><strong>Last Login:</strong> {profile.last\_login || 'N/A'}</p>

              <p><strong>Token Expiry:</strong> {profile.token\_expiry || 'N/A'}</p>

              <button

                onClick={handleLogout}

                className="mt-2 bg-red-600 text-white py-1 px-2 rounded-md hover:bg-red-700"

              >

                Logout

              </button>

            </>

          ) : (

            <p>Loading...</p>

          )}

        </div>

      )}

    </div>

  );

}

export default ProfileIcon;

>>>> TimetableForm.jsx ->

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function TimetableForm({ onClose }) {

  const [section, setSection] = useState('');

  const [teacher, setTeacher] = useState('');

  const [semesterStart, setSemesterStart] = useState('2025-03-01');

  const [semesterEnd, setSemesterEnd] = useState('2025-08-31');

  const [dailySchedules, setDailySchedules] = useState({

    Monday: [], Tuesday: [], Wednesday: [], Thursday: [], Friday: [], Saturday: [],

  });

  const [sections, setSections] = useState([]);

  const [teachers, setTeachers] = useState([]);

  const [subjects, setSubjects] = useState([]);

  const [error, setError] = useState('');

  const [success, setSuccess] = useState('');

  useEffect(() => {

    const token = localStorage.getItem('access\_token');

    if (!token) {

      setError('Please log in first');

      return;

    }

    const fetchSections = async () => {

      try {

        const response = await axios.get('http://localhost:8000/api/sections/', {

          headers: { Authorization: `Bearer ${token}` },

        });

        setSections(response.data);

      } catch (err) {

        setError('Failed to load sections');

      }

    };

    const fetchTeachers = async () => {

      try {

        const response = await axios.get('http://localhost:8000/api/teachers/', {

          headers: { Authorization: `Bearer ${token}` },

        });

        setTeachers(response.data);

        if (response.data.length > 0) setTeacher(response.data[0].id);

      } catch (err) {

        setError('Failed to load teachers');

      }

    };

    fetchSections();

    fetchTeachers();

  }, []);

  useEffect(() => {

    if (section) {

      const fetchSubjects = async () => {

        const token = localStorage.getItem('access\_token');

        const selectedSection = sections.find((sec) => sec.id === parseInt(section));

        if (selectedSection) {

          try {

            const response = await axios.get('http://localhost:8000/api/subjects/', {

              headers: { Authorization: `Bearer ${token}` },

            });

            const semesterSubjects = response.data.filter((sub) => sub.semester === selectedSection.year);

            setSubjects(semesterSubjects);

          } catch (err) {

            setError('Failed to load subjects');

          }

        }

      };

      fetchSubjects();

    }

  }, [section, sections]);

  const addSchedule = (day) => {

    if (dailySchedules[day].length >= 5) {

      setError(`Cannot add more than 5 lectures on ${day}`);

      return;

    }

    setDailySchedules((prev) => ({

      ...prev,

      [day]: [...prev[day], { subject: subjects[0]?.id || '', start\_time: '08:30:00' }],

    }));

    setError('');

  };

  const updateSchedule = (day, index, field, value) => {

    setDailySchedules((prev) => {

      const updated = [...prev[day]];

      updated[index] = { ...updated[index], [field]: value };

      return { ...prev, [day]: updated };

    });

  };

  const removeSchedule = (day, index) => {

    setDailySchedules((prev) => ({

      ...prev,

      [day]: prev[day].filter((\_, i) => i !== index),

    }));

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    const token = localStorage.getItem('access\_token');

    if (!token) {

      setError('Please log in first');

      return;

    }

    const payload = {

      section,

      teacher,

      daily\_schedules: Object.entries(dailySchedules).flatMap(([day, schedules]) =>

        schedules.map((sched) => ({

          day\_of\_week: day,

          subject: sched.subject,

          start\_time: sched.start\_time,

        }))

      ),

      semester\_start\_date: semesterStart,

      semester\_end\_date: semesterEnd,

    };

    if (payload.daily\_schedules.length === 0) {

      setError('Please add at least one lecture');

      return;

    }

    console.log('Submitting payload:', JSON.stringify(payload, null, 2));

    try {

      const response = await axios.post('http://localhost:8000/api/timetables/', payload, {

        headers: { Authorization: `Bearer ${token}` },

      });

      console.log('Response:', response.data);

      setSuccess('Semester timetable created successfully');

      setError('');

      setTimeout(onClose, 2000);

    } catch (err) {

      console.error('Error response:', err.response ? err.response.data : err.message);

      setError(err.response?.data?.detail || 'Failed to create timetable');

      setSuccess('');

    }

  };

  const days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];

  return (

    <div className="fixed top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 bg-white p-6 rounded-lg shadow-lg z-50 max-w-4xl w-full max-h-[80vh] overflow-y-auto">

      <h3 className="text-xl font-bold mb-4 text-gray-800">Add Semester Timetable</h3>

      {error && <p className="text-red-500 mb-4">{error}</p>}

      {success && <p className="text-green-500 mb-4">{success}</p>}

      <form onSubmit={handleSubmit}>

        <div className="mb-4">

          <label className="block text-gray-700 mb-2 font-medium">Section:</label>

          <select

            value={section}

            onChange={(e) => setSection(e.target.value)}

            className="w-full p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

          >

            <option value="">Select Section</option>

            {sections.map((sec) => (

              <option key={sec.id} value={sec.id}>

                {sec.name} - Semester {sec.year}

              </option>

            ))}

          </select>

        </div>

        <div className="mb-4">

          <label className="block text-gray-700 mb-2 font-medium">Teacher:</label>

          <select

            value={teacher}

            onChange={(e) => setTeacher(e.target.value)}

            className="w-full p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

          >

            <option value="">Select Teacher</option>

            {teachers.map((t) => (

              <option key={t.id} value={t.id}>

                {t.first\_name} {t.last\_name} ({t.email})

              </option>

            ))}

          </select>

        </div>

        <div className="mb-4 grid grid-cols-2 gap-4">

          <div>

            <label className="block text-gray-700 mb-2 font-medium">Semester Start:</label>

            <input

              type="date"

              value={semesterStart}

              onChange={(e) => setSemesterStart(e.target.value)}

              className="w-full p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

            />

          </div>

          <div>

            <label className="block text-gray-700 mb-2 font-medium">Semester End:</label>

            <input

              type="date"

              value={semesterEnd}

              onChange={(e) => setSemesterEnd(e.target.value)}

              className="w-full p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

            />

          </div>

        </div>

        {section && (

          <div>

            {days.map((day) => (

              <div key={day} className="mb-4">

                <h4 className="text-lg font-semibold mb-2">{day}</h4>

                {dailySchedules[day].map((sched, index) => (

                  <div key={index} className="flex space-x-2 mb-2 items-center">

                    <select

                      value={sched.subject}

                      onChange={(e) => updateSchedule(day, index, 'subject', e.target.value)}

                      className="w-1/2 p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

                    >

                      <option value="">Select Subject</option>

                      {subjects.map((sub) => (

                        <option key={sub.id} value={sub.id}>

                          {sub.name}

                        </option>

                      ))}

                    </select>

                    <select

                      value={sched.start\_time}

                      onChange={(e) => updateSchedule(day, index, 'start\_time', e.target.value)}

                      className="w-1/4 p-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"

                    >

                      <option value="08:30:00">08:30 AM</option>

                      <option value="09:30:00">09:30 AM</option>

                      <option value="10:30:00">10:30 AM</option>

                      <option value="12:00:00">12:00 PM</option>

                      <option value="13:00:00">01:00 PM</option>

                    </select>

                    <button

                      type="button"

                      onClick={() => removeSchedule(day, index)}

                      className="bg-red-500 text-white py-1 px-2 rounded-md hover:bg-red-600"

                    >

                      Remove

                    </button>

                  </div>

                ))}

                <button

                  type="button"

                  onClick={() => addSchedule(day)}

                  className="bg-gray-500 text-white py-1 px-2 rounded-md hover:bg-gray-600"

                >

                  Add Lecture

                </button>

              </div>

            ))}

          </div>

        )}

        <div className="mt-6 flex justify-end space-x-2">

          <button

            type="button"

            onClick={onClose}

            className="bg-gray-500 text-white py-2 px-4 rounded-md hover:bg-gray-600 transition"

          >

            Cancel

          </button>

          <button

            type="submit"

            className="bg-blue-500 text-white py-2 px-4 rounded-md hover:bg-blue-600 transition"

          >

            Save Semester Timetable

          </button>

        </div>

      </form>

    </div>

  );

}

export default TimetableForm;

Now by analyzing these all ( SRS , programs , the code ) etc you might know what are the requirements for our automated attendance system for the law\_college where our database name is law\_college . but still if any details you want you can ask so you can better understand it and build the project quickly. Let me you you again we want fully automated attendance system where admin should be able to perform CRUD operations on Student, Teacher, Timetable, Session , Stats , everything on that can be CRUD operation performed , the teacher should be able to add timetable, take student attendance mark them present/ absent ( if need they should be able to update student attendance ) and these all things should be fully automated. If you are still not clear please ask for additional details so we can further proceed