the **attendance marking feature** for your React frontend, integrating it with the Django backend’s /api/mark-attendance/<session\_id>/ endpoint. This feature will allow teachers to mark attendance for a specific session, with a default of "Absent" and the ability to set "Present" for students in the session’s section. We’ll enhance the existing Calendar component to trigger an attendance form and create a new AttendanceForm component to handle the marking process.

**Requirements Recap**

* **Endpoint**: POST /api/mark-attendance/<session\_id>/
  + Payload: [{"student\_id": 1, "status": "Present"}, ...].
* **Behavior**:
  + Default status is "Absent" (already set in the model).
  + Teachers mark "Present" for attending students.
  + Restricted to the session’s teacher.
* **Display**: List students in the session’s section with toggleable status.

**Step 1: Update Calendar.js**

Modify the Calendar component to open an attendance form when a session is clicked.

jsx

CollapseWrapCopy

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

import axios from 'axios';

import FullCalendar from '@fullcalendar/react';

import dayGridPlugin from '@fullcalendar/daygrid';

import timeGridPlugin from '@fullcalendar/timegrid';

import AttendanceForm from './AttendanceForm';

function Calendar() {

const [events, setEvents] = useState([]);

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

const [selectedSession, setSelectedSession] = useState(null);

useEffect(() => {

const fetchSessions = async () => {

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

if (!token) {

setError('No authentication token found. Please log in.');

return;

}

try {

const response = await axios.get('http://localhost:8000/api/calendar/', {

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

});

const sessionEvents = response.data.map((session) => ({

title: `${session.timetable.subject.name} (${session.status})`,

start: `${session.date}T${session.timetable.start\_time}`,

end: `${session.date}T${new Date(`1970-01-01T${session.timetable.start\_time}`).getTime() + 60\*60\*1000}`.slice(0, -1),

extendedProps: {

sessionId: session.id,

status: session.status,

},

}));

setEvents(sessionEvents);

} catch (err) {

setError('Failed to fetch sessions');

console.error(err);

}

};

fetchSessions();

}, []);

const handleEventClick = (info) => {

if (info.event.extendedProps.status !== 'Scheduled') {

alert('Attendance can only be marked for Scheduled sessions.');

return;

}

setSelectedSession({

id: info.event.extendedProps.sessionId,

title: info.event.title,

});

};

const closeAttendanceForm = () => {

setSelectedSession(null);

*// Refresh calendar after marking attendance*

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

axios.get('http://localhost:8000/api/calendar/', {

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

}).then((response) => {

const updatedEvents = response.data.map((session) => ({

title: `${session.timetable.subject.name} (${session.status})`,

start: `${session.date}T${session.timetable.start\_time}`,

end: `${session.date}T${new Date(`1970-01-01T${session.timetable.start\_time}`).getTime() + 60\*60\*1000}`.slice(0, -1),

extendedProps: {

sessionId: session.id,

status: session.status,

},

}));

setEvents(updatedEvents);

});

};

return (

<div style={{ maxWidth: '1000px', margin: '50px auto' }}>

<h2>Your Teaching Schedule</h2>

{error && <p style={{ color: 'red' }}>{error}</p>}

<FullCalendar

plugins={[dayGridPlugin, timeGridPlugin]}

initialView="timeGridWeek"

events={events}

slotMinTime="08:00:00"

slotMaxTime="14:00:00"

headerToolbar={{

left: 'prev,next today',

center: 'title',

right: 'dayGridMonth,timeGridWeek,timeGridDay',

}}

eventClick={handleEventClick}

/>

{selectedSession && (

<AttendanceForm session={selectedSession} onClose={closeAttendanceForm} />

)}

</div>

);

}

export default Calendar;

**Step 2: Create AttendanceForm.js**

This new component fetches students for the session’s section, allows marking attendance, and submits to the backend.

jsx

CollapseWrapCopy

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

import axios from 'axios'*;*

function AttendanceForm({ session, onClose }) {

const [students, setStudents] = useState([])*;*

const [attendanceData, setAttendanceData] = useState({})*;*

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

const [success, setSuccess] = useState('')*;*

useEffect(() => {

const fetchStudents = async () => {

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

try {

// Fetch session details to get section

const sessionResponse = await axios.get(`http://localhost:8000/api/sessions/${session.id}/`, {

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

})*;*

const sectionId = sessionResponse.data.timetable.section.split(' - ')[1]*; // Extract section ID if needed*

// Fetch students in the section (assuming section is a string*; adjust if ID-based)*

const studentsResponse = await axios.get('http://localhost:8000/api/students/', {

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

})*;*

const sectionStudents = studentsResponse.data.filter(

(student) => student.section === sessionResponse.data.timetable.section

)*;*

setStudents(sectionStudents)*;*

// Initialize attendance data with default "Absent"

const initialAttendance = {}*;*

sectionStudents.forEach((student) => {

initialAttendance[student.id] = 'Absent'*;*

})*;*

setAttendanceData(initialAttendance)*;*

} catch (err) {

setError('Failed to load students')*;*

console.error(err)*;*

}

}*;*

fetchStudents()*;*

}, [session])*;*

const handleStatusChange = (studentId, status) => {

setAttendanceData((prev) => ({

...prev,

[studentId]: status,

}))*;*

}*;*

const handleSubmit = async (e) => {

e.preventDefault()*;*

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

const payload = Object.keys(attendanceData).map((studentId) => ({

student\_id: parseInt(studentId),

status: attendanceData[studentId],

}))*;*

try {

const response = await axios.post(

`http://localhost:8000/api/mark-attendance/${session.id}/`,

{ attendance: payload },

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

)*;*

setSuccess(response.data.message)*;*

setError('')*;*

setTimeout(onClose, 2000)*; // Close form after 2 seconds*

} catch (err) {

setError(err.response?.data?.error || 'Failed to mark attendance')*;*

setSuccess('')*;*

}

}*;*

return (

<div style={{

position: 'fixed', top: '50%', left: '50%', transform: 'translate(-50%, -50%)',

background: 'white', padding: '20px', borderRadius: '8px', boxShadow: '0 0 10px rgba(0,0,0,0.3)',

zIndex: 1000, maxWidth: '600px', width: '100%'

}}>

<h3>Mark Attendance for {session.title}</h3>

{error && <p style={{ color: 'red' }}>{error}</p>}

{success && <p style={{ color: 'green' }}>{success}</p>}

<form onSubmit={handleSubmit}>

<table style={{ width: '100%', borderCollapse: 'collapse' }}>

<thead>

<tr>

<th style={{ border: '1px solid #ddd', padding: '8px' }}>Roll Number</th>

<th style={{ border: '1px solid #ddd', padding: '8px' }}>Name</th>

<th style={{ border: '1px solid #ddd', padding: '8px' }}>Status</th>

</tr>

</thead>

<tbody>

{students.map((student) => (

<tr key={student.id}>

<td style={{ border: '1px solid #ddd', padding: '8px' }}>{student.roll\_number}</td>

<td style={{ border: '1px solid #ddd', padding: '8px' }}>{student.first\_name} {student.last\_name}</td>

<td style={{ border: '1px solid #ddd', padding: '8px' }}>

<select

value={attendanceData[student.id] || 'Absent'}

onChange={(e) => handleStatusChange(student.id, e.target.value)}

style={{ padding: '4px' }}

>

<option value="Absent">Absent</option>

<option value="Present">Present</option>

</select>

</td>

</tr>

))}

</tbody>

</table>

<div style={{ marginTop: '20px', textAlign: 'right' }}>

<button type="button" onClick={onClose} style={{ padding: '8px 16px', marginRight: '10px' }}>

Cancel

</button>

<button type="submit" style={{ padding: '8px 16px' }}>Save Attendance</button>

</div>

</form>

</div>

)*;*

}

export default AttendanceForm*;*

**Step 3: Update Backend for Students Endpoint**

The AttendanceForm needs a way to fetch students, but we don’t have a /api/students/ endpoint yet. Let’s add it.

**3.1 Update core/serializers.py**

Add:

python

CollapseWrapCopy

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', 'section', 'subjects']

**3.2 Update core/views.py**

Add:

python

CollapseWrapCopy

class StudentViewSet(viewsets.ReadOnlyModelViewSet):

queryset = Student.objects.all()

serializer\_class = StudentSerializer

permission\_classes = [IsAuthenticated]

def get\_queryset(self):

*# Optionally restrict to students in teacher’s sections*

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

**3.3 Update core/urls.py**

python

CollapseWrapCopy

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

**3.4 Apply Changes**

bash

CollapseWrapCopy

python manage.py runserver

**How It Works**

1. **Calendar Click**:
   * Clicking a "Scheduled" session opens the AttendanceForm.
   * Non-"Scheduled" sessions show an alert.
2. **Attendance Form**:
   * Fetches students in the session’s section via /api/students/.
   * Displays a table with roll number, name, and a dropdown (default "Absent").
   * Teachers toggle "Present" or leave as "Absent".
   * Submits to /api/mark-attendance/<session\_id>/.
3. **Backend**:
   * Updates Attendance records, sets session to "Completed".
   * Returns success message.

**Testing**

1. **Add Sample Data** (if not done):

python

CollapseWrapCopy

from core.models import \*

from datetime import date

program = Program.objects.create(name="BALLB 5 Yr", duration\_years=5)

section = Section.objects.create(program=program, name="Section A", year=1)

student = Student.objects.create(roll\_number="G240001", first\_name="Alice", last\_name="Smith", email="alice@example.com", section=section)

subject = Subject.objects.create(name="Law of Contract I", is\_law\_subject=True, semester=1)

teacher = Teacher.objects.get(email="john.doe@example.com")

timetable = Timetable.objects.create(section=section, subject=subject, teacher=teacher, day\_of\_week="Monday", start\_time="08:30:00", semester\_start\_date=date(2025, 3, 1), semester\_end\_date=date(2025, 6, 30))

session = Session.objects.create(timetable=timetable, date=date(2025, 3, 31))

1. **Run Both**:
   * Backend: python manage.py runserver.
   * Frontend: npm start.
2. **Test Flow**:
   * Log in at http://localhost:3000/.
   * Go to calendar, click a session (e.g., March 31, 2025).
   * Mark attendance, save, and see the calendar update.

**Next Steps**

* **Enhance Form**: Add loading states, better error handling.
* **Stats**: Build the attendance stats display (/api/attendance-stats/<roll\_number>/).
* **Polish**: Style the form better (e.g., with Material-UI).