import { useEffect, useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import './calendar.css';

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

import { ShowDetailingOrder } from "./showDetailingOrder";

import { getDetailingOrdersThunk } from "../redux/slices/getDetailingOrderThunk";

import { getOrdersThunk } from "../redux/slices/getOrderThunk";

import { Nivut } from "./nivut";

import { Maneger } from "./manegar";

// Import hebcal for Hebrew calendar support

// import { HebrewCalendar, HDate, Location, Event } from 'hebcal';

// בתחילת הקובץ, הוסף את הייבוא

import { HebrewCalendarHelper } from './HebrewCalendarHelper';

export const Calendar = () => {

    const [currentDate, setCurrentDate] = useState(new Date());

    const [view, setView] = useState('month'); // 'month' or 'week'

    const [selectedEvent, setSelectedEvent] = useState(null);

    const [showEventModal, setShowEventModal] = useState(false);

    const [showAddEventModal, setShowAddEventModal] = useState(false);

    const [jewishHolidays, setJewishHolidays] = useState([]);

    const orders = useSelector(state => state.orderSlice.orders);

    const detailingO = useSelector(state => state.orderSlice.detailingOrders);

    const schoolName = useSelector(state => state.schoolsSlice.school.name);

    const dispatch = useDispatch();

    const navigate = useNavigate();

    // Day names in Hebrew

    const dayNames = ["ראשון", "שני", "שלישי", "רביעי", "חמישי", "שישי", "שבת"];

    // Month names in English and Hebrew

    const monthNamesEn = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];

    const monthNamesHe = ["ינואר", "פברואר", "מרץ", "אפריל", "מאי", "יוני", "יולי", "אוגוסט", "ספטמבר", "אוקטובר", "נובמבר", "דצמבר"];

    // Hebrew months

    const hebrewMonths = ["ניסן", "אייר", "סיון", "תמוז", "אב", "אלול", "תשרי", "חשון", "כסלו", "טבת", "שבט", "אדר", "אדר ב'"];

    // Get orders from API

    const getOrders = async() => {

        await dispatch(getOrdersThunk());

    };

// ובמקום השימוש בספריית hebcal, השתמש בפונקציות מהקובץ שלנו:

// במקום getHebrewDate המקורי

const getHebrewDate = (date) => {

    return HebrewCalendarHelper.getHebrewDate(date);

  };

  // במקום getJewishHolidays המקורי

  const getHolidaysForDate = (date) => {

    return HebrewCalendarHelper.getHolidaysForDate(date);

  };

    // Show event details

    const showEventDetails = async(id) => {

        await dispatch(getDetailingOrdersThunk(id));

        setSelectedEvent(id);

        setShowEventModal(true);

    };

    // Navigate to previous month/week

    const goToPrevious = () => {

        const newDate = new Date(currentDate);

        if (view === 'month') {

            newDate.setMonth(newDate.getMonth() - 1);

        } else {

            newDate.setDate(newDate.getDate() - 7);

        }

        setCurrentDate(newDate);

    };

    // Navigate to next month/week

    const goToNext = () => {

        const newDate = new Date(currentDate);

        if (view === 'month') {

            newDate.setMonth(newDate.getMonth() + 1);

        } else {

            newDate.setDate(newDate.getDate() + 7);

        }

        setCurrentDate(newDate);

    };

    // Go to today

    const goToToday = () => {

        setCurrentDate(new Date());

    };

    // Convert Gregorian date to Hebrew date

    // const getHebrewDate = (date) => {

    //     const hDate = new HDate(date);

    //     return {

    //         day: hDate.getDate(),

    //         month: hDate.getMonthName(),

    //         year: hDate.getFullYear(),

    //         toString: hDate.toString('h')

    //     };

    // };

    // Get Jewish holidays for a specific month

    const getJewishHolidays = (year, month) => {

        // Israel location for holiday calculations

        // const location = Location.lookup('Jerusalem');

        // Get start and end dates for the month

        const startDate = new Date(year, month, 1);

        const endDate = new Date(year, month + 1, 0);

        // Get all events for the date range

        // const events = HebrewCalendar.calendar({

        //     start: startDate,

        //     end: endDate,

        //     location: location,

        //     sedrot: true,

        //     candlelighting: true,

        //     havdalah: true

        // });

        // Convert to a more usable format

        return events.map(event => ({

            date: event.getDate().greg(),

            title: event.render('he'),

            desc: event.getDesc(),

            category: event.getCategories()[0]

        }));

    };

    // Generate days for monthly view

    const generateMonthDays = () => {

        const year = currentDate.getFullYear();

        const month = currentDate.getMonth();

        // First day of the month

        const firstDay = new Date(year, month, 1);

        // Last day of the month

        const lastDay = new Date(year, month + 1, 0);

        // Get the day of the week for the first day (0 = Sunday, 6 = Saturday)

        const firstDayOfWeek = firstDay.getDay();

        // Calculate days from previous month to show

        const daysFromPrevMonth = firstDayOfWeek;

        // Calculate total days to show (previous month + current month + next month)

        const totalDays = daysFromPrevMonth + lastDay.getDate();

        // Ensure we complete the last week if needed

        const daysFromNextMonth = 7 - (totalDays % 7 || 7);

        const allDays = [];

        // Add days from previous month

        const prevMonth = new Date(year, month, 0);

        const prevMonthDays = prevMonth.getDate();

        for (let i = prevMonthDays - daysFromPrevMonth + 1; i <= prevMonthDays; i++) {

            const date = new Date(year, month - 1, i);

            allDays.push({

                date,

                isCurrentMonth: false,

                isToday: false,

                hebrewDate: getHebrewDate(date),

                holidays: getHolidaysForDate(date)

            });

        }

        // Add days from current month

        const today = new Date();

        for (let i = 1; i <= lastDay.getDate(); i++) {

            const date = new Date(year, month, i);

            allDays.push({

                date,

                isCurrentMonth: true,

                isToday: date.toDateString() === today.toDateString(),

                hebrewDate: getHebrewDate(date),

                holidays: getHolidaysForDate(date)

            });

        }

        // Add days from next month

        for (let i = 1; i <= daysFromNextMonth; i++) {

            const date = new Date(year, month + 1, i);

            allDays.push({

                date,

                isCurrentMonth: false,

                isToday: false,

                hebrewDate: getHebrewDate(date),

                holidays: getHolidaysForDate(date)

            });

        }

        return allDays;

    };

    // Get holidays for a specific date

    // const getHolidaysForDate = (date) => {

    //     if (!jewishHolidays || jewishHolidays.length === 0) return [];

    //     return jewishHolidays.filter(holiday =>

    //         holiday.date.toDateString() === date.toDateString()

    //     );

    // };

    // Generate days for weekly view

    const generateWeekDays = () => {

        const weekDays = [];

        const startOfWeek = new Date(currentDate);

        // Adjust to start of week (Sunday)

        const dayOfWeek = startOfWeek.getDay();

        startOfWeek.setDate(startOfWeek.getDate() - dayOfWeek);

        // Generate 7 days starting from Sunday

        for (let i = 0; i < 7; i++) {

            const date = new Date(startOfWeek);

            date.setDate(startOfWeek.getDate() + i);

            weekDays.push({

                date,

                isToday: date.toDateString() === new Date().toDateString(),

                hebrewDate: getHebrewDate(date),

                holidays: getHolidaysForDate(date)

            });

        }

        return weekDays;

    };

    // Generate time slots for weekly view

    const generateTimeSlots = () => {

        const timeSlots = [];

        for (let hour = 8; hour <= 20; hour++) {

            timeSlots.push(`${hour}:00`);

        }

        return timeSlots;

    };

    // Find events for a specific date

    const getEventsForDate = (date) => {

        if (!orders || orders.length === 0) return [];

        return orders.filter(order => {

            const orderDate = new Date(order.dateOfEvent);

            return orderDate.toDateString() === date.toDateString();

        });

    };

    // Format date for display

    const formatDate = (date) => {

        return date.toLocaleDateString('he-IL');

    };

    // Handle saving a new event

    const handleSaveEvent = (eventData) => {

        // Here you would typically dispatch an action to save the event

        console.log('Saving event:', eventData);

        // For now, we'll just close the modal

        setShowAddEventModal(false);

    };

    useEffect(() => {

        getOrders();

        // Get Jewish holidays for the current month

        const holidays = getJewishHolidays(

            currentDate.getFullYear(),

            currentDate.getMonth()

        );

        setJewishHolidays(holidays);

    }, [currentDate]);

    return (

        <div className="main">

            {schoolName === "maneger" ? <Maneger /> : <Nivut />}

            <div className="calendar-container">

                <div className="calendar-header">

                    <div className="calendar-title">

                        {view === 'month'

                            ? `${monthNamesHe[currentDate.getMonth()]} ${currentDate.getFullYear()}`

                            : `שבוע ${Math.ceil(currentDate.getDate() / 7)} - ${monthNamesHe[currentDate.getMonth()]} ${currentDate.getFullYear()}`

                        }

                    </div>

                    <div className="calendar-nav">

                        <button className="calendar-nav-btn" onClick={goToPrevious}>▶</button>

                        <button className="calendar-today-btn" onClick={goToToday}>היום</button>

                        <button className="calendar-nav-btn" onClick={goToNext}>◀</button>

                    </div>

                </div>

                <div className="calendar-view-toggle">

                    <button

                        className={`view-btn ${view === 'month' ? 'active' : ''}`}

                        onClick={() => setView('month')}

                    >

                        חודשי

                    </button>

                    <button

                        className={`view-btn ${view === 'week' ? 'active' : ''}`}

                        onClick={() => setView('week')}

                    >

                        שבועי

                    </button>

                </div>

                {view === 'month' ? (

                    <>

                        <div className="calendar-days-header">

                            {dayNames.map((day, index) => (

                                <div key={index} className="day-name">{day}</div>

                            ))}

                        </div>

                        <div className="calendar-month-grid">

                            {generateMonthDays().map((day, index) => (

                                <div

                                    key={index}

                                    className={`calendar-day ${!day.isCurrentMonth ? 'other-month' : ''} ${day.isToday ? 'today' : ''}`}

                                >

                                    <div className="day-number">

                                        {day.date.getDate()}

                                        <span className="hebrew-date">{day.hebrewDate.day}</span>

                                    </div>

                                    {/\* Display Jewish holidays \*/}

                                    {day.holidays.map((holiday, holidayIndex) => (

                                        <div

                                            key={`holiday-${holidayIndex}`}

                                            className="holiday-event"

                                            title={holiday.desc}

                                        >

                                            {holiday.title}

                                        </div>

                                    ))}

                                    {/\* Display regular events \*/}

                                    {getEventsForDate(day.date).map((event, eventIndex) => (

                                        <div

                                            key={eventIndex}

                                            className={`event event-${eventIndex % 3 === 0 ? 'blue' : eventIndex % 3 === 1 ? 'green' : 'orange'}`}

                                            onClick={() => showEventDetails(event.idOrder)}

                                        >

                                            {event.title || 'אירוע'}

                                        </div>

                                    ))}

                                </div>

                            ))}

                        </div>

                    </>

                ) : (

                    <div className="calendar-week-view">

                        <div className="time-column">

                            <div className="week-day-header"></div>

                            {generateTimeSlots().map((time, index) => (

                                <div key={index} className="time-slot">{time}</div>

                            ))}

                        </div>

                        {generateWeekDays().map((day, dayIndex) => (

                            <div key={dayIndex} className="week-day-column">

                                <div className={`week-day-header ${day.isToday ? 'today' : ''}`}>

                                    <div>{dayNames[day.date.getDay()]}</div>

                                    <div className="week-date">

                                        {formatDate(day.date)}

                                        <span className="hebrew-date-small">{day.hebrewDate.toString}</span>

                                    </div>

                                </div>

                                {/\* Display Jewish holidays at the top of the day column \*/}

                                {day.holidays.length > 0 && (

                                    <div className="week-holiday-container">

                                        {day.holidays.map((holiday, holidayIndex) => (

                                            <div

                                                key={`holiday-${holidayIndex}`}

                                                className="holiday-event-week"

                                                title={holiday.desc}

                                            >

                                                {holiday.title}

                                            </div>

                                        ))}

                                    </div>

                                )}

                                {generateTimeSlots().map((time, timeIndex) => {

                                    const events = getEventsForDate(day.date);

                                    const hour = parseInt(time.split(':')[0]);

                                    // Filter events for this specific hour

                                    const hourEvents = events.filter(event => {

                                        const eventDate = new Date(event.dateOfEvent);

                                        return eventDate.getHours() === hour;

                                    });

                                    return (

                                        <div key={timeIndex} className="week-time-slot">

                                            {hourEvents.map((event, eventIndex) => (

                                                <div

                                                    key={eventIndex}

                                                    className={`event event-${eventIndex % 3 === 0 ? 'blue' : eventIndex % 3 === 1 ? 'green' : 'orange'}`}

                                                    onClick={() => showEventDetails(event.idOrder)}

                                                >

                                                    {event.title || 'אירוע'}

                                                </div>

                                            ))}

                                        </div>

                                    );

                                })}

                            </div>

                        ))}

                    </div>

                )}

            </div>

            {/\* Add Event Button \*/}

            <div className="add-event-btn" title="הוסף אירוע" onClick={() => setShowAddEventModal(true)}>+</div>

            {/\* Event Modal \*/}

            {showEventModal && (

                <div className="event-modal">

                    <div className="event-modal-content">

                        <button className="event-modal-close" onClick={() => setShowEventModal(false)}>×</button>

                        <h2 className="event-modal-title">פרטי אירוע</h2>

                        <div className="event-modal-details">

                            <ShowDetailingOrder detailingOrders={detailingO} setd={setShowEventModal} />

                        </div>

                        <div className="event-modal-actions">

                            <button className="event-modal-btn event-modal-btn-secondary" onClick={() => setShowEventModal(false)}>סגור</button>

                        </div>

                    </div>

                </div>

            )}

            {/\* Add Event Modal \*/}

            {showAddEventModal && (

                <AddEventModal

                    isOpen={showAddEventModal}

                    onClose={() => setShowAddEventModal(false)}

                    onSave={handleSaveEvent}

                />

            )}

        </div>

    );

};

// Add Event Modal Component

const AddEventModal = ({ isOpen, onClose, onSave }) => {

    const [eventData, setEventData] = useState({

        title: '',

        date: new Date().toISOString().split('T')[0],

        startTime: '09:00',

        endTime: '10:00',

        description: '',

        color: '#ff2e63'

    });

    const handleChange = (e) => {

        const { name, value } = e.target;

        setEventData(prev => ({

            ...prev,

            [name]: value

        }));

    };

    const handleSubmit = (e) => {

        e.preventDefault();

        onSave(eventData);

    };

    if (!isOpen) return null;

    return (

        <div className="event-modal">

            <div className="event-modal-content">

                <button className="event-modal-close" onClick={onClose}>×</button>

                <h2 className="event-modal-title">הוסף אירוע חדש</h2>

                <form onSubmit={handleSubmit}>

                    <div className="form-group">

                        <label htmlFor="title">כותרת</label>

                        <input

                            type="text"

                            id="title"

                            name="title"

                            value={eventData.title}

                            onChange={handleChange}

                            required

                            className="form-control"

                        />

                    </div>

                    <div className="form-group">

                        <label htmlFor="date">תאריך</label>

                        <input

                            type="date"

                            id="date"

                            name="date"

                            value={eventData.date}

                            onChange={handleChange}

                            required

                            className="form-control"

                        />

                    </div>

                    <div className="form-row">

                        <div className="form-group half">

                            <label htmlFor="startTime">שעת התחלה</label>

                            <input

                                type="time"

                                id="startTime"

                                name="startTime"

                                value={eventData.startTime}

                                onChange={handleChange}

                                className="form-control"

                            />

                        </div>

                        <div className="form-group half">

                            <label htmlFor="endTime">שעת סיום</label>

                            <input

                                type="time"

                                id="endTime"

                                name="endTime"

                                value={eventData.endTime}

                                onChange={handleChange}

                                className="form-control"

                            />

                        </div>

                    </div>

                    <div className="form-group">

                        <label htmlFor="description">תיאור</label>

                        <textarea

                            id="description"

                            name="description"

                            value={eventData.description}

                            onChange={handleChange}

                            className="form-control"

                            rows="3"

                        ></textarea>

                    </div>

                    <div className="form-group">

                        <label htmlFor="color">צבע</label>

                        <div className="color-options">

                            {['#ff2e63', '#4361ee', '#4CAF50', '#ff9e00', '#9c27b0'].map(color => (

                                <div

                                    key={color}

                                    className={`color-option ${eventData.color === color ? 'selected' : ''}`}

                                    style={{ backgroundColor: color }}

                                    onClick={() => setEventData(prev => ({ ...prev, color }))}

                                ></div>

                            ))}

                        </div>

                    </div>

                    <div className="event-modal-actions">

                        <button type="button" className="event-modal-btn event-modal-btn-secondary" onClick={onClose}>

                            ביטול

                        </button>

                        <button type="submit" className="event-modal-btn event-modal-btn-primary">

                            שמור אירוע

                        </button>

                    </div>

                </form>

            </div>

        </div>

    );

};

/\* Calendar Component Styles \*/

.calendar-container {

    max-width: 1200px;

    margin: 30px auto;

    background-color: white;

    border-radius: 15px;

    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);

    overflow: hidden;

    direction: rtl;

  }

  /\* Calendar Header \*/

  .calendar-header {

    background: linear-gradient(135deg, #1a1a2e, #ff2e63);

    padding: 20px;

    color: white;

    display: flex;

    justify-content: space-between;

    align-items: center;

  }

  .calendar-title {

    font-size: 1.8rem;

    font-weight: 700;

    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);

  }

  .calendar-nav {

    display: flex;

    gap: 10px;

  }

  .calendar-nav-btn {

    background-color: rgba(255, 255, 255, 0.2);

    border: none;

    color: white;

    width: 40px;

    height: 40px;

    border-radius: 50%;

    display: flex;

    align-items: center;

    justify-content: center;

    cursor: pointer;

    transition: all 0.3s ease;

    font-size: 1.2rem;

  }

  .calendar-nav-btn:hover {

    background-color: rgba(255, 255, 255, 0.4);

    transform: translateY(-2px);

  }

  .calendar-today-btn {

    background-color: rgba(255, 255, 255, 0.2);

    border: none;

    color: white;

    padding: 8px 15px;

    border-radius: 20px;

    cursor: pointer;

    transition: all 0.3s ease;

    font-weight: 600;

    font-size: 0.9rem;

  }

  .calendar-today-btn:hover {

    background-color: rgba(255, 255, 255, 0.4);

  }

  /\* View Toggle \*/

  .calendar-view-toggle {

    display: flex;

    padding: 15px;

    background-color: #f8f9fa;

    border-bottom: 1px solid #e9ecef;

  }

  .view-btn {

    padding: 10px 20px;

    background: none;

    border: none;

    border-radius: 20px;

    cursor: pointer;

    font-weight: 600;

    color: #666;

    transition: all 0.3s ease;

  }

  .view-btn.active {

    background-color: #ff2e63;

    color: white;

    box-shadow: 0 4px 10px rgba(255, 46, 99, 0.3);

  }

  .view-btn:hover:not(.active) {

    background-color: #e9ecef;

  }

  /\* Days Header \*/

  .calendar-days-header {

    display: grid;

    grid-template-columns: repeat(7, 1fr);

    background-color: #f8f9fa;

    border-bottom: 1px solid #e9ecef;

  }

  .day-name {

    padding: 15px 10px;

    text-align: center;

    font-weight: 600;

    color: #1a1a2e;

  }

  /\* Monthly View \*/

  .calendar-month-grid {

    display: grid;

    grid-template-columns: repeat(7, 1fr);

    gap: 1px;

    background-color: #e9ecef;

  }

  .calendar-day {

    min-height: 120px;

    background-color: white;

    padding: 10px;

    transition: all 0.3s ease;

    position: relative;

  }

  .calendar-day:hover {

    background-color: #f8f9fa;

  }

  .day-number {

    font-size: 1.1rem;

    font-weight: 600;

    margin-bottom: 10px;

    color: #333;

  }

  .other-month .day-number {

    color: #adb5bd;

  }

  .today {

    background-color: rgba(255, 46, 99, 0.05);

  }

  .today .day-number {

    background-color: #ff2e63;

    color: white;

    width: 30px;

    height: 30px;

    border-radius: 50%;

    display: flex;

    align-items: center;

    justify-content: center;

    box-shadow: 0 4px 8px rgba(255, 46, 99, 0.3);

  }

  /\* Weekly View \*/

  .calendar-week-view {

    display: grid;

    grid-template-columns: 80px repeat(7, 1fr);

    gap: 1px;

    background-color: #e9ecef;

  }

  .time-column {

    background-color: #f8f9fa;

  }

  .time-slot {

    height: 60px;

    padding: 5px;

    border-bottom: 1px solid #e9ecef;

    display: flex;

    align-items: center;

    justify-content: center;

    font-size: 0.8rem;

    color: #666;

  }

  .week-day-column {

    background-color: white;

  }

  .week-day-header {

    padding: 10px;

    text-align: center;

    font-weight: 600;

    border-bottom: 1px solid #e9ecef;

    background-color: #f8f9fa;

  }

  .week-day-header.today {

    background-color: rgba(255, 46, 99, 0.1);

    color: #ff2e63;

  }

  .week-time-slot {

    height: 60px;

    border-bottom: 1px solid #e9ecef;

    padding: 5px;

    position: relative;

  }

  /\* Events \*/

  .event {

    background-color: #ff2e63;

    color: white;

    padding: 5px 8px;

    border-radius: 5px;

    font-size: 0.85rem;

    margin-bottom: 5px;

    cursor: pointer;

    transition: all 0.3s ease;

    overflow: hidden;

    text-overflow: ellipsis;

    white-space: nowrap;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  }

  .event:hover {

    transform: translateY(-2px);

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);

  }

  .event-blue {

    background-color: #4361ee;

  }

  .event-green {

    background-color: #4CAF50;

  }

  .event-orange {

    background-color: #ff9e00;

  }

  /\* Event Modal \*/

  .event-modal {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background-color: rgba(0, 0, 0, 0.5);

    display: flex;

    align-items: center;

    justify-content: center;

    z-index: 1000;

  }

  .event-modal-content {

    background-color: white;

    border-radius: 15px;

    width: 90%;

    max-width: 500px;

    padding: 25px;

    box-shadow: 0 15px 30px rgba(0, 0, 0, 0.2);

    position: relative;

  }

  .event-modal-close {

    position: absolute;

    top: 15px;

    left: 15px;

    background: none;

    border: none;

    font-size: 1.5rem;

    cursor: pointer;

    color: #666;

    transition: color 0.3s ease;

  }

  .event-modal-close:hover {

    color: #ff2e63;

  }

  .event-modal-title {

    font-size: 1.8rem;

    font-weight: 700;

    margin-bottom: 15px;

    color: #1a1a2e;

  }

  .event-modal-details {

    margin-bottom: 20px;

  }

  .event-detail {

    display: flex;

    margin-bottom: 10px;

    align-items: center;

  }

  .event-detail-icon {

    margin-left: 10px;

    color: #ff2e63;

    font-size: 1.2rem;

  }

  .event-detail-text {

    color: #666;

  }

  .event-modal-actions {

    display: flex;

    justify-content: flex-end;

    gap: 10px;

    margin-top: 20px;

  }

  .event-modal-btn {

    padding: 10px 20px;

    border-radius: 5px;

    font-weight: 600;

    cursor: pointer;

    transition: all 0.3s ease;

  }

  .event-modal-btn-primary {

    background-color: #ff2e63;

    color: white;

    border: none;

  }

  .event-modal-btn-primary:hover {

    background-color: #ff0a54;

  }

  .event-modal-btn-secondary {

    background-color: #f8f9fa;

    color: #666;

    border: 1px solid #e9ecef;

  }

  .event-modal-btn-secondary:hover {

    background-color: #e9ecef;

  }

  /\* Add Event Button \*/

  .add-event-btn {

    position: fixed;

    bottom: 30px;

    left: 30px;

    width: 60px;

    height: 60px;

    border-radius: 50%;

    background-color: #ff2e63;

    color: white;

    display: flex;

    align-items: center;

    justify-content: center;

    font-size: 1.5rem;

    box-shadow: 0 5px 15px rgba(255, 46, 99, 0.4);

    cursor: pointer;

    transition: all 0.3s ease;

    z-index: 100;

  }

  .add-event-btn:hover {

    transform: translateY(-5px);

    box-shadow: 0 8px 20px rgba(255, 46, 99, 0.5);

  }

  /\* Responsive Design \*/

  @media (max-width: 992px) {

    .calendar-month-grid {

      grid-template-columns: repeat(7, 1fr);

    }

    .calendar-day {

      min-height: 100px;

    }

  }

  @media (max-width: 768px) {

    .calendar-day {

      min-height: 80px;

      padding: 5px;

    }

    .day-number {

      font-size: 1rem;

      margin-bottom: 5px;

    }

    .event {

      padding: 3px 5px;

      font-size: 0.75rem;

    }

    .calendar-week-view {

      grid-template-columns: 60px repeat(7, 1fr);

    }

  }

  @media (max-width: 576px) {

    .calendar-month-grid {

      grid-template-columns: repeat(7, 1fr);

    }

    .calendar-day {

      min-height: 60px;

      padding: 3px;

    }

    .day-number {

      font-size: 0.9rem;

      margin-bottom: 3px;

    }

    .event {

      padding: 2px 4px;

      font-size: 0.7rem;

      margin-bottom: 2px;

    }

    .calendar-title {

      font-size: 1.4rem;

    }

    .calendar-week-view {

      display: flex;

      flex-direction: column;

    }

    .week-day-column {

      display: flex;

      flex-direction: column;

    }

    .time-column {

      display: none;

    }

    .week-time-slot {

      height: auto;

      min-height: 40px;

    }

  }

  /\* Hebrew Date Styles \*/

.hebrew-date {

    display: block;

    font-size: 0.75rem;

    color: #666;

    margin-top: 2px;

    font-weight: normal;

  }

  .hebrew-date-small {

    display: block;

    font-size: 0.7rem;

    color: #666;

    margin-top: 2px;

  }

  /\* Holiday Styles \*/

  .holiday-event {

    background-color: #f8d7da;

    color: #721c24;

    padding: 2px 5px;

    border-radius: 3px;

    font-size: 0.75rem;

    margin-bottom: 3px;

    cursor: default;

    white-space: nowrap;

    overflow: hidden;

    text-overflow: ellipsis;

    border-right: 3px solid #721c24;

  }

  .holiday-event-week {

    background-color: #f8d7da;

    color: #721c24;

    padding: 2px 5px;

    border-radius: 3px;

    font-size: 0.7rem;

    margin-bottom: 2px;

    cursor: default;

    white-space: nowrap;

    overflow: hidden;

    text-overflow: ellipsis;

    border-right: 3px solid #721c24;

  }

  .week-holiday-container {

    padding: 5px;

    background-color: #fff9f9;

    border-bottom: 1px solid #f5c6cb;

  }

  /\* Form Styles \*/

  .form-group {

    margin-bottom: 20px;

  }

  .form-row {

    display: flex;

    gap: 15px;

  }

  .form-group.half {

    width: 50%;

  }

  .form-control {

    width: 100%;

    padding: 10px 15px;

    border: 1px solid #e9ecef;

    border-radius: 5px;

    font-size: 1rem;

    transition: border-color 0.3s ease;

  }

  .form-control:focus {

    outline: none;

    border-color: #ff2e63;

    box-shadow: 0 0 0 3px rgba(255, 46, 99, 0.1);

  }

  label {

    display: block;

    margin-bottom: 8px;

    font-weight: 600;

    color: #333;

  }

  .color-options {

    display: flex;

    gap: 10px;

  }

  .color-option {

    width: 30px;

    height: 30px;

    border-radius: 50%;

    cursor: pointer;

    transition: all 0.3s ease;

    border: 2px solid transparent;

  }

  .color-option:hover {

    transform: scale(1.1);

  }

  .color-option.selected {

    border-color: #333;

    transform: scale(1.1);

    box-shadow: 0 0 0 2px white, 0 0 0 4px rgba(0, 0, 0, 0.1);

  }

  /\* Shabbat and Holiday Styling \*/

  .calendar-day:nth-child(7n),

  .calendar-day:nth-child(7n-1) {

    background-color: #f8f9fa;

  }

  .calendar-day.has-holiday {

    background-color: #fff9f9;

  }

  /\* Responsive adjustments for Hebrew dates \*/

  @media (max-width: 768px) {

    .hebrew-date {

      font-size: 0.65rem;

    }

    .holiday-event {

      font-size: 0.65rem;

      padding: 1px 3px;

    }

  }

  @media (max-width: 576px) {

    .hebrew-date {

      font-size: 0.6rem;

    }

    .hebrew-date-small {

      display: none;

    }

    .holiday-event {

      font-size: 0.6rem;

      padding: 1px 2px;

      margin-bottom: 1px;

    }

  }