# Trips Allocation

**npm package:** `flatpickr`

---

# Backend: `tripsAllocatedRange` Function (PHP)

php
Copy Edit

```
publicfunctiontripsAllocatedRange($request)
{
    $start_date = $request->start_date;
    $end_date = $request->end_date;

    $site_settings = $this->fetchSiteSettings()->keyBy('name');
    $weekends = array_map('trim', explode(',',
$site_settings['Weekend']->value));
    $holidays = array_map('trim', explode(',',
$site_settings['Holiday Dates']->value));

    $current_date = $start_date;
    $trips_allocated = [];

    while ($current_date <= $end_date) {
        $carbonDate = Carbon::parse($current_date);
        $dayOfWeek = $carbonDate->format('l');

        $isHoliday = in_array($carbonDate->format('Y-m-d'),
$holidays);
        $isWeekend = in_array($dayOfWeek, $weekends);

        if (!$isHoliday && !$isWeekend) {
            $trips_allocated[$current_date] = [
                'trips'      => $this-
>tripsAllocated($current_date),
                'is_holiday' => $isHoliday,
                'is_weekend' => $isWeekend
            ];
        } else {
            // Still list holidays/weekends with 0 trips
            $trips_allocated[$current_date] = [
```

```
                'trips'      => 0,
                'is_holiday' => $isHoliday,
                'is_weekend' => $isWeekend
            ];
        }

        $current_date = $carbonDate->addDay()->format('Y-m-d');
    }

    returnresponse()->json($trips_allocated);
}
```

---

# Frontend Initialization (JavaScript)

javascript

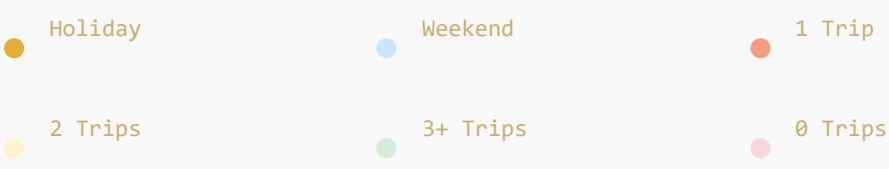
Copy Edit

```
let tripData = {}; // Global store
const today = newDate();
today.setHours(0, 0, 0, 0);

const proposedEmptyingDate = "{{ $application ? $application->proposed_emptying_date : '' }}";

flatpickr('.flatpickr-reschedule', {
    dateFormat: 'Y-m-d',
    allowInput: true,
    minDate: "today",

    onReady: function (selectedDates, dateStr, instance) {
        if (instance.input.id === 'confirmed_emptying_date') {
            // Inject legend
            const legendHTML = `
```




---

```
            `;

instance.calendarContainer.insertAdjacentHTML("afterbegin", legendHTML);
            fetchAndDisplayTrips(instance);
        }
    },
```

```
    onMonthChange: function (selectedDates, dateStr, instance) {
        if (instance.input.id === 'confirmed_emptying_date') {
            fetchAndDisplayTrips(instance);
        }
    },

    onDayCreate: function (dObj, dStr, fp, dayElem) {
        const dateObj = dayElem.dateObj;
        if (!dateObj) return;

        const year = dateObj.getFullYear();
        const month = String(dateObj.getMonth() + 1).padStart(2,
'0');
        const day = String(dateObj.getDate()).padStart(2, '0');
        const dateStrKey = `${year}-${month}-${day}`;

        // Disable past dates
        const isPast = dateObj < today;
        if (isPast) {
            dayElem.classList.add('flatpickr-disabled');
            dayElem.style.backgroundColor = "#eee";
            dayElem.style.color = "#888";
            dayElem.style.cursor = "not-allowed";
            dayElem.addEventListener('click', (e) => {
                e.preventDefault();
                e.stopPropagation();
                Swal.fire({
                    toast: true,
                    position: 'top-end',
                    icon: 'warning',
                    title: 'Cannot select a past date.',
                    showConfirmButton: false,
                    timer: 2000,
                    timerProgressBar: true
                });
            });
            return;
        }

        // Trip data coloring
        if (tripData.hasOwnProperty(dateStrKey)) {
            const { trips, is_holiday, is_weekend } =
tripData[dateStrKey];
            dayElem.removeAttribute("style");
            dayElem.style.cursor = "pointer";

            let tooltip = `Trips Available: ${trips}`;
            if (is_holiday) tooltip += " (Holiday)";
            if (is_weekend) tooltip += " (Weekend)";
            dayElem.setAttribute("title", tooltip);

            // Coloring
            if (is_holiday) {
                dayElem.style.backgroundColor = "rgb(228, 173,
56)";
                dayElem.style.color = "#000";
```

```
            } elseif (is_weekend) {
                dayElem.style.backgroundColor = "#cce5ff";
                dayElem.style.color = "#004085";
            } elseif (trips === 0) {
                dayElem.style.backgroundColor = "#f8d7da";
                dayElem.style.color = "#721c24";
            } elseif (trips === 1) {
                dayElem.style.backgroundColor = "rgb(245, 157,
130)";
                dayElem.style.color = "#856404";
            } elseif (trips === 2) {
                dayElem.style.backgroundColor = "#fff3cd";
                dayElem.style.color = "#856404";
            } else {
                dayElem.style.backgroundColor = "#d4edda";
                dayElem.style.color = "#155724";
            }

            dayElem.style.borderRadius = "50%";

            // Block selection for invalid dates
            if (is_holiday || is_weekend || trips === 0) {
                dayElem.addEventListener('click', (e) => {
                    e.preventDefault();
                    e.stopPropagation();
                    let message = is_holiday
                        ? 'Cannot select a holiday date.'
                        : is_weekend
                        ? 'Cannot select a weekend date.'
                        : 'No trips available for this date.';
                    Swal.fire({
                        toast: true,
                        position: 'top-end',
                        icon: 'warning',
                        title: message,
                        showConfirmButton: false,
                        timer: 2000,
                        timerProgressBar: true
                    });
                });
            }
        }
    },

    disable: [
        function (date) {
            if (date < today) returntrue;
            const year = date.getFullYear();
            const month = String(date.getMonth() + 1).padStart(2,
'0');
            const day = String(date.getDate()).padStart(2, '0');
            const dateKey = `${year}-${month}-${day}`;

            if (tripData[dateKey]) {
                const { trips, is_holiday, is_weekend } =
tripData[dateKey];
                return is_holiday || is_weekend || trips === 0;
```

```
            }
            returnfalse;
        }
    ]
});

functionfetchAndDisplayTrips(instance) {
    const calendarContainer = instance.calendarContainer;
    const dayElements =
calendarContainer.querySelectorAll(".flatpickr-day");
    if (dayElements.length === 0) return;

    const firstVisibleDay = newDate(dayElements[0].dateObj);
    firstVisibleDay.setDate(firstVisibleDay.getDate() -
firstVisibleDay.getDay());
    const lastVisibleDay = newDate(dayElements[dayElements.length
- 1].dateObj);
    lastVisibleDay.setDate(lastVisibleDay.getDate() + (6 -
lastVisibleDay.getDay()));

    const startDateFormatted =
firstVisibleDay.toISOString().slice(0, 10);
    const endDateFormatted =
lastVisibleDay.toISOString().slice(0, 10);

    constformatDMY = (d) => `${String(d.getDate()).padStart(2,
'0')}.${String(d.getMonth() + 1).padStart(2,
'0')}.${d.getFullYear()}`;
    const displayTarget = document.getElementById("visible-range-
display");
    if (displayTarget) {
        displayTarget.innerText = `Calendar Grid:
${formatDMY(firstVisibleDay)} - ${formatDMY(lastVisibleDay)}`;
    }

    $.ajax({
        url: "{{ route('schedule.tripsallocated.range') }}",
        type: 'POST',
        data: {
            start_date: startDateFormatted,
            end_date: endDateFormatted
        },
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')
        },
        dataType: 'json',
        success: function (response) {
            tripData = response;
            console.log("✅ Trip data loaded:", tripData);
            instance.redraw();
        },
        error: function (xhr, status, error) {
            console.error("❌ Failed to fetch trip data:",
error);
        }
```

```
    });
}
```