## **Frontend Data Requirements for CalendarSchedule Component**

This document details the data structures and formats required by the CalendarSchedule React component to render the weekly view and handle event interactions.

**1. weekDays (Object/Map)**

* **Purpose:** Maps numerical day representation (as returned by date.getDay(), where Sunday is 0) to localized day names. Used for displaying day headers in the calendar grid and event details.
* **Structure:**

| {  [dayNumber: number]: **string**; // Key is the day number (0-6), value is the day name } |
| --- |

* **Example (@/lib/mockData.ts):**

| **export** **const** weekDays: { [key: number]: string } = {  0: "Chủ Nhật", // Sunday  1: "Thứ 2", // Monday  2: "Thứ 3", // Tuesday  3: "Thứ 4", // Wednesday  4: "Thứ 5", // Thursday  5: "Thứ 6", // Friday  6: "Thứ 7", // Saturday }; |
| --- |

**2. timeSlots (Array of Strings)**

* **Purpose:** Defines the time intervals displayed as rows in the weekly calendar grid. Events are placed within the cell corresponding to their day and matching time slot.
* **Structure:**  
  string[] // Each string represents a time slot, typically "HH:MM - HH:MM"
* **Example (@/lib/mockData.ts):**

| **export** **const** timeSlots: string[] = [  "07:00 - 08:30",  "08:45 - 10:15",  "10:30 - 12:00",  // ... more slots ]; |
| --- |

* + **Note:** The format of the strings in timeSlots *must exactly match* the format created by concatenating event.startTime and event.endTime for the filtering logic (${event.startTime} - ${event.endTime} === slot) to work correctly.

**3. teachers (Array of Objects)**

* **Purpose:** Provides information about each teacher, used for displaying names, avatars, subjects, and color-coding events in the calendar and dialog.
* **Structure:**

| Array<{  id: number; // Unique identifier for the teacher  name: string; // Teacher's full name  subject: string; // Subject the teacher teaches  avatar?: string; // Optional URL to the teacher's avatar image  color?: string; // Optional Tailwind CSS class string for background/border color (e.g., "bg-blue-100 border-blue-300") }> |
| --- |

* **Example (@/lib/mockData.ts):**  
  export const teachers: Teacher[] = [ // Assuming a Teacher type/interface is defined

| {  id: 1,  name: "Nguyễn Văn A",  subject: "Toán",  avatar: "/avatars/teacher1.png", // Example path  color: "bg-blue-100 border-blue-300 text-blue-800",  },  // ... more teachers ]; |
| --- |

**4. scheduleEvents (Array of Objects)**

* **Purpose:** Contains the actual schedule data. Each object represents a single class or event to be displayed on the calendar.
* **Structure:**

| Array<{  id: number; // Unique identifier **for** the event  title: string; // Name **or** title of the event (e.g., "Lớp Toán 10A1")  teacherId: number; // ID linking to the teacher **in** the `teachers` array  room: string; // Room number **or** name (e.g., "Phòng A101")  day: number; // Day of the week (0 **for** Sunday, 1 **for** Monday, ..., 6 **for** Saturday)  startTime: string; // Start time **in** "HH:MM" format (24-hour)  endTime: string; // End time **in** "HH:MM" format (24-hour) }> |
| --- |

* **Example (@/lib/mockData.ts):**  
  export const scheduleEvents: ScheduleEvent[] = [ // Assuming a ScheduleEvent type/interface

| {  id: 101,  title: "Toán Cao Cấp",  teacherId: 1, // Links to teacher Nguyễn Văn A  room: "A101",  day: 1, // Monday  startTime: "07:00",  endTime: "08:30",  },  // ... more events ]; |
| --- |

* + **Constraint:** The combination of startTime and endTime (formatted as "${startTime} - ${endTime}") must match one of the strings present in the timeSlots array for the event to be correctly placed in the grid.

**5. Utility Functions (@/lib/utils)**

* getWeekNumber(date: Date): number: Calculates the week number of the year for a given date.
* getDateForDay(currentDate: Date, targetDay: number): Date: Calculates the actual Date object for a specific day of the week (targetDay) within the week of the currentDate.
* getCurrentDay(): number: Returns the current day of the week (0-6).

**6. API Functions (@/lib/api)**

* updateScheduleEvent(eventId: number, eventData: any): Promise<void>: An asynchronous function expected to handle the logic for updating an event on the backend (or mock backend). Takes the event ID and the updated event object.
* deleteScheduleEvent(eventId: number): Promise<void>: An asynchronous function expected to handle the logic for deleting an event on the backend (or mock backend). Takes the event ID.

**Summary:**

The component relies heavily on structured arrays and objects, primarily scheduleEvents and teachers, along with helper data like timeSlots and weekDays. Consistent formatting, especially for time strings and IDs, is crucial for the filtering and rendering logic to function correctly. The component also depends on external utility and API functions for date calculations and data manipulation.