

## SCHEDULING API ERRORS AND PROPOSED SOLUTIONS

### Request Validation

Validation request returns a completely vague error list in the form request here is a sample

```
{
  "detail": [
    {
      "type": "missing",
      "loc": [
        "body"
      ],
      "msg": "Field required",
      "input": null
    }
  ]
}
```

### Proposed Solution

Your Form Validation Should Return A Highly human friendly validation and also ensure that all fields have been validated depending on their standard types e.g course\_id (string, UUID), hall\_id (string, UUID) e.t.c

### Sample of what is actually expected

```
{
  "errors": {
    "preferred teaching period start time": [
      "Preferred teaching period start time is required."
    ],
    "busy period end time": [
      "Busy period end time must be after the start time."
    ]
  }
}
```

### Timetable Break Constrains Enforcement

#### Enforced Constrains

The fixed break times are respected through-out all the days are enforced

#### Not Enforced Constrains

- **Days Exception (days\_exception)**

The days\_exception is not respected when tested breaks are still enforced on this days

## NB

The `days_exception` constraint specifies the days which will not have this break periods at all

- **Days Fixed Breaks (`days_fixed_breaks`)**

The `days_fixed_breaks` constraint is not respected when fixed breaks are clearly stated

## Proposal

- The break should not be scheduled on days listed in `days_exception`
- When the daily flag is true, the break time should be the same every day, unless there is a specific override in `days_fixed_breaks`
- The fixed break times specified in `days_fixed_breaks` should override the general `start_time` and `end_time`, but only on their respective days.
- On days in `days_exception`, the break should be skipped altogether
- If `daily` is true and no specific fixed break for that day exists, the break should follow the general `start_time` and `end_time`.

## Example

```
start_time: 12:00
end_time: 12:45
daily: true
days_exception: ["monday", "tuesday"]
days_fixed_breaks: [
  { "day": "monday", "start_time": "11:30", "end_time": "12:15" }
]
```

## Expected behavior:

On Monday: The break is not scheduled because it's in `days_exception`.

On Tuesday: The break is not scheduled for the same reason.

On Wednesday: The break is scheduled from 12:00 to 12:45 (following the general time, as no fixed break is specified).

On Monday (if not in exception): The break would follow the fixed break time 11:30 to 12:15.

## Edge Cases

### 1. Invalid Day in `days_exception`

Edge case:

`days_exception` contains a day that is not a valid weekday (e.g., "funday").

## Handling:

Detect invalid days and inform the user.

**Error message:****Title:**

Invalid day detected in exception list

**Message:**

The day "funday" in your days\_exception is not recognized. Please use only valid weekdays: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday.  
Please review and correct the list.

**2. start\_time after end\_time****Edge case:**

start\_time is later than end\_time (e.g., "14:00" to "12:00").

**Handling:**

Check time order and prompt correction.

**Error message:****Title:**

Invalid time range

**Message:**

Your start time ("14:00") is later than your end time ("12:00"). Please ensure the start time is earlier than the end time to define a valid break period.

**3. Missing start\_time or end\_time****Edge case:**

Either start\_time or end\_time is not provided.

**Handling:**

Require both times to be set.

**Error message:****Title:**

Incomplete break time configuration

**Message:**

Both start time and end time must be specified to define your break period. Please review your settings.

#### **4. Overlapping fixed breaks**

**Edge case:**

days\_fixed\_breaks contain overlapping break periods on the same day.

**Handling:**

Detect overlaps and notify the user.

**Error message:**

Title:

Conflicting fixed break periods

**Message:**

There are overlapping break periods scheduled on the same day. Please ensure each break period is distinct and does not overlap with others for clarity and proper scheduling.

#### **5. Empty days\_fixed\_breaks when daily is true**

**Edge case:**

daily is true, but days\_fixed\_breaks are empty, and no general start\_time/end\_time is set.

**Handling:**

Prompt users to specify break times.

**Error message:**

Title:

Missing break times for daily schedule

**Message:**

Your schedule is set to daily, but no break times are specified. Please set a general break time or define fixed breaks for specific days.

#### **6. Invalid time format**

**Edge case:**

start\_time or end\_time is not in valid "HH:MM" format.

**Handling:**

Validate time format and notify users.

**Error message:**

Title:

Invalid time format

**Message:**

One or more of your times are not in the correct format. Please use the "HH:MM" format (e.g., "12:30"). Correct the times and try again.

**7. Empty days\_fixed\_breaks when fixed break is specified****Edge case:**

days\_fixed\_breaks is empty, but a specific day is mentioned in the schedule.

**Handling:**

Notify users that fixed break data is missing.

**Error message:**

Title:

Missing fixed break details

**Message:**

You have specified fixed break days, but no break times are provided. Please add the start and end times for these days.

**NB: Some edge cases that might involve data validation can be handled in the form request**

**Scheduling Slots Duration Issue**

Currently, the scheduling system generates time slots with a constant duration of 30 minutes, regardless of the specific needs of each session or day. This results in rigid schedules that may not align with actual session lengths or preferences, leading to inefficient timetable management.

**Mock Example of Error:**

```
{
  "day": "Friday",
  "start_time": "14:00",
  "end_time": "14:30",
  "break": false,
  "duration": "30min",
  "teacher_id": "99z8y7x6-w5v4-u3t2-s1r0-q9p8o7n6m5l4",
  "teacher_name": "Dr Chen",
  "course_id": "f0g1h2i3-j4k5-6l7m-8n9p-0q1r2s3t4u5v",
  "course_name": "Formal Methods in Software Engineering",
  "hall_id": "c1d2e3f4-a5b6-7c8d-9e0f-1a2b3c4d5e6f",
  "hall_name": "Main Lecture Theatre A"
},
```

```

{
  "day": "Friday",
  "start_time": "14:30",
  "end_time": "15:00",
  "break": false,
  "duration": "30min",
  "teacher_id": "a9b1c2d3-e4f5-67g8-90h1-2i3j4k5l6m7n",
  "teacher_name": "Ms Davis",
  "course_id": "b3d5e7f9-a1c3-5e7f-9g1h-3i5j7k9l1m3n",
  "course_name": "Object-Oriented Design and Analysis (OODA)",
  "hall_id": "f5e4d3c2-b1a0-9876-5432-1fedcba98765",
  "hall_name": "Medium Lecture Hall B"
}

```

## Proposed Solution

Introduce a flexible configuration called periods that allows defining custom slot durations. This configuration supports global, exception, and fixed-day period settings, providing granular control over schedule slot lengths.

```

"periods": {
  "daily": boolean,
  "period": number, // in minutes
  "constrains": [
    {
      "days_exception": [ "monday", "tuesday", ... ] // optional
    },
    {
      "days_fixed_periods": [
        {
          "day": "monday",
          "period": 120
        },
        {
          "day": "wednesday",
          "period": 45
        }
      ]
    }
  ]
}

```

### Key points:

- daily: If true, apply a uniform period to all days unless overridden.
- period: Default duration in minutes.
- constraints: List of exceptions or fixed day-specific periods.

### Examples

#### Scenario 1: Global 45-minute slots with specific days overridden

```
"periods": {  
  "daily": true,  
  "period": 45,  
  "constraints": [  
    {  
      "days_fixed_periods": [  
        { "day": "monday", "period": 30 },  
        { "day": "wednesday", "period": 60 }  
      ]  
    }  
  ]  
}
```

#### Result:

- All days have 45-minute slots by default.
- Monday slots are 30 minutes.
- Wednesday slots are 60 minutes.

#### Scenario 2: No global setting, only specific days

```
"periods": {  
  "daily": false,  
  "period": 30,  
  "constraints": [  
    {  
      "days_fixed_periods": [  
        { "day": "friday", "period": 90 }  
      ]  
    }  
  ]  
}
```

**Result:**

Default slots are 30 minutes.

Friday slots are 90 minutes.

**Handling Days With No Schedule or Only Break Periods**

In the schedule data structure, days may sometimes contain only break slots without any scheduled classes. When such days are processed for display or further use, they may produce empty or irrelevant class arrays, leading to confusion or clutter in the schedule output. The system should avoid generating or displaying days that have no classes scheduled, even if break slots are present, to maintain clarity and relevance in the schedule data.

**Example**

```
{
  "day": "Saturday",
  "slots": [
    {
      "day": "Saturday",
      "start_time": "12:00",
      "end_time": "12:45",
      "break": true,
      "duration": null,
      "teacher_id": null,
      "teacher_name": null,
      "course_id": null,
      "course_name": null,
      "hall_id": null,
      "hall_name": null
    }
  ]
}
```

**Current Output:**

The system might produce an entry for Saturday with only break slots, which could be unnecessary if no classes are scheduled.

**Problem:**

Including days with only break slots in the schedule can lead to a cluttered schedule view, as these days do not contain any actual classes. This can confuse users and reduce the clarity of the schedule display, especially when generating weekly schedules or reports.

**Solution Proposal:**

Implement logic to exclude days that contain only break slots and no classes from the final schedule output. Specifically:

- When processing each day's slots, check if there are any non-break class slots.
- If none exist (i.e., the day contains only break slots), omit that day from the final schedule.

**Enforcing Teacher Preferred Teaching Times to Prevent Scheduling Outside Preferences**

Currently, teachers may be scheduled to teach outside their specified preferred times, leading to conflicts and dissatisfaction. This issue undermines the scheduling system's compliance with teachers' preferences, potentially causing operational and morale problems. It is essential that the system strictly enforces teachers' preferred teaching time windows, ensuring no schedule places teachers outside these specified periods.

**Example:**

Teacher A has a preferred teaching window from 09:00 to 12:00.

Current scheduling might assign Teacher A to a class at 14:00, which violates their preference.

**Solution Proposal:**

Implement a strict validation and enforcement mechanism that ensures that for the endpoint where there are scheduled with preference is strictly enforced

