Flask Simple Task Timer API - Challenge

Overview

Build a Task Timer API where each timer session is independent. Start a timer to get a session ID, then stop it later using that ID. Track all sessions and filter by status.

Time Limit

40 minutes

Requirements

API Endpoints

POST `/sessions/start`

Start a new timer session

Request Body:

```
{
  "task_name": "coding"
}
```

Success Response (201):

```
{
  "session_id": 1,
  "task_name": "coding",
  "started_at": "2024-01-15T14:30:00",
  "status": "running"
}
```

Error Response (400):

```
{
  "error": "Invalid task name",
  "message": "Task name must be between 3 and 20 characters"
}
```

Validation Rules:
- Task name: 3-20 characters

POST `/sessions/{session_id}/stop`

Stop a running timer session

Success Response (200):

```
{
  "session_id": 1,
  "task_name": "coding",
  "started_at": "2024-01-15T14:30:00",
  "stopped_at": "2024-01-15T15:45:00",
  "duration_seconds": 4500,
  "status": "completed"
}
```

Error Response (404):

```
{
  "error": "Session not found",
  "message": "No session with ID 1"
}
```

Error Response (400):

```
{
  "error": "Session already stopped",
  "message": "Session 1 was already stopped"
}
```

GET `/sessions`

List all sessions with optional filtering

Query Parameters (optional):
- `status` : Filter by status (`running` or `completed`)

Success Response (200):

```
{
  "sessions": [
    {
      "session_id": 1,
      "task_name": "coding",
      "started_at": "2024-01-15T14:30:00",
      "stopped_at": "2024-01-15T15:45:00",
      "duration_seconds": 4500,
      "status": "completed"
    },
    {
      "session_id": 2,
      "task_name": "meeting",
      "started_at": "2024-01-15T16:00:00",
      "stopped_at": null,
      "duration_seconds": 1200,
      "status": "running"
    }
  ],
  "total": 2
}
```

Note: For running sessions, calculate `duration_seconds` from start time to current time.

Examples:

```
GET /sessions                   # All sessions
GET /sessions?status=running    # Only running sessions
GET /sessions?status=completed  # Only completed sessions
```

GET `/sessions/{session_id}`

Get details of a specific session

Success Response (200):

```
{
  "session_id": 1,
  "task_name": "coding",
  "started_at": "2024-01-15T14:30:00",
  "stopped_at": "2024-01-15T15:45:00",
  "duration_seconds": 4500,
  "status": "completed"
}
```

Note: For running sessions, calculate current duration.

Error Response (404):

```
{
  "error": "Session not found",
  "message": "No session with ID 1"
}
```

DELETE `/sessions/{session_id}`

Delete a session

Success Response (200):

```
{
  "message": "Session deleted successfully",
  "session_id": 1,
  "task_name": "coding"
}
```

Error Response (404):

```
{
  "error": "Session not found",
  "message": "No session with ID 1"
}
```

Data Structure

```
from datetime import datetime

# Global storage
sessions = []
next_session_id = 1

# Session example:
{
    "session_id": 1,
    "task_name": "coding",
    "started_at": datetime(2024, 1, 15, 14, 30, 0),
    "stopped_at": None,  # None if running
    "duration_seconds": None  # Calculated when stopped
}
```

Implementation Hints

Getting Current Time

```
from datetime import datetime

# Get current time
now = datetime.now()
# Example: datetime(2024, 1, 15, 14, 30, 45)
```

Calculating Duration

```
from datetime import datetime

started_at = datetime(2024, 1, 15, 14, 0, 0)   # 2:00 PM
stopped_at = datetime(2024, 1, 15, 15, 30, 0)  # 3:30 PM

# Calculate difference
duration = (stopped_at - started_at).total_seconds()
# Result: 5400.0 seconds (90 minutes)

# Convert to integer
duration_seconds = int(duration)
# Result: 5400
```

Using Global Variables for ID Counter

```
# At top of file (module level)
next_session_id = 1

# Inside a function where you need to increment
def some_function():
    global next_session_id  # ← Must declare to modify global variable

    current_id = next_session_id
    next_session_id += 1  # Increment: 1 → 2 → 3 → 4 ...

    # Use current_id for the new session
```

Testing Checklist

```
# 1. Start first session
POST /sessions/start {"task_name": "coding"}
→ session_id: 1

# 2. Start second session
POST /sessions/start {"task_name": "meeting"}
→ session_id: 2

# 3. Stop first session
POST /sessions/1/stop
→ Returns duration

# 4. Try to stop again
POST /sessions/1/stop
→ 400 Error: Already stopped

# 5. List all sessions
GET /sessions
→ Shows 2 sessions (1 completed, 1 running)

# 6. Filter running sessions
GET /sessions?status=running
→ Shows only session 2

# 7. Filter completed sessions
GET /sessions?status=completed
→ Shows only session 1

# 8. Get specific session
GET /sessions/2
→ Shows session 2 with current duration

# 9. Delete session
DELETE /sessions/1
→ Success

# 10. Invalid task name
POST /sessions/start {"task_name": "ab"}
→ 400 Error: Too short
```

Deliverables

1. All 5 endpoints working
2. Task name validation (3-20 chars, alphanumeric + hyphens/underscores)
3. Session ID auto-increment
4. Duration calculation (stopped and running sessions)
5. Filtering by status
6. Proper error handling (400, 404)
7. RESTful design