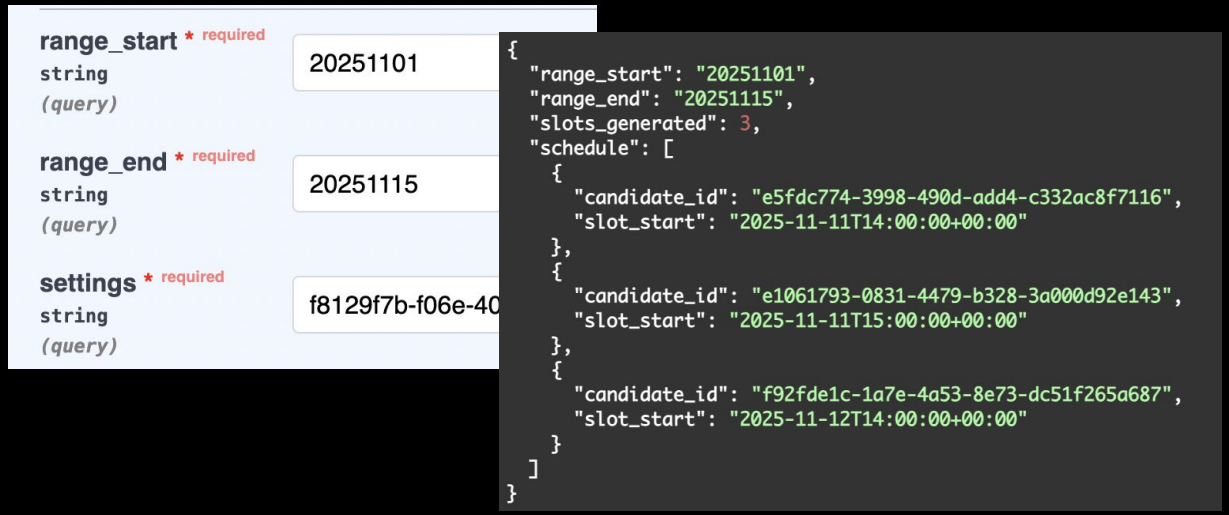

Interview Calendar - Backend

Mahdi Almosawi

Contributions

- Database configuration with supabase
- API endpoints, including:
  - Get_interviews
  - Scheduler
  - Get_settings
- Automatic interview scheduling algorithm

Artifacts

```
    while cur <= end:
        day_start = cur.replace(hour=workday_start, minute=0)
        day_end = cur.replace(hour=workday_end, minute=0)
        all_days.append((day_start, day_end))
        cur += timedelta(days=1)

    print("📅 Days expanded:")
    for d_start, d_end in all_days:
        print("  →", d_start, "to", d_end)

    return all_days

def schedule_candidates(records, range_start, range_end, int_id, buffer_minutes, interval_minutes):

    """
    Input:
      - records: JSON from GET /availability
      - day_start/day_end: scheduling window

    Output:
      - list of {candidate_id, slot_start}
    """

    # Step 1: Parse availability JSON
    candidate_availability, interviewer_intervals = parse_availability_json(records, int_id)

    all_days = expand_date_range(range_start, range_end)
    print("trying to generate time slots")
    # Step 2: Generate 30-minute time slots
    all_slots = []
    slot_index_to_datetime = {}
    idx = 0
    for day_start, day_end in all_days:
    # Use the interval_minutes parameter
        slots = generate_time_slots(day_start, day_end, interviewer_intervals, buffer_minutes, interval_minutes)
        for s in slots:
            all_slots.append(s)
            slot_index_to_datetime[idx] = s
            idx += 1


    # Step 3: Build bipartite graph
    G = build_bipartite_graph(candidate_availability, all_slots)
    left_nodes = [n for n in G.nodes if n.startswith("cand:")]



    # Step 4: Run Hopcroft-Karp matching (NetworkX)
    matching = nx.algorithms.bipartite.matching.hopcroft_karp_matching(G, left_nodes)


    # Step 5: Convert to usable schedule
    schedule = []
    for cand_node in left_nodes:
        if cand_node in matching:
            slot_node = matching[cand_node]
            if slot_node.startswith("slot:"):
                slot_idx = int(slot_node.split(":")[1])
                cid = cand_node.split("cand:")[1]
                schedule.append({
                    "candidate_id": cid,
                    "slot_start": slot_index_to_datetime[slot_idx].isoformat()
                })

    # Sort schedule by datetime
    schedule.sort(key=lambda x: x["slot_start"])
    return schedule
"""
```

[My contributions on the roadmap]

[My pushes]

[Backend Repo]

[README]

Updated skills:

1. Last four digits of student ID: 7611
2. First Name and Last Initial: Mahdi A
3. Skills: Python, Java, C, Web Design (HTML/CSS, Javascript), Git, backend development (fastAPI, SQL), and cooking
4. Preferred Roles: I intend to become a teacher at the highschool level, if that is what the question is asking