Timetabling Algorithm Planning

Existing APIs

1. OptaPlanner

- **Best For**: Complex scheduling with constraints.
- **Features**:
  - Handles complex university timetabling problems.
  - Supports hard and soft constraints like avoiding class overlaps, room capacity limits, and teacher availability.
  - Can find optimal schedules based on a variety of inputs.
- **Use Case**: Perfect for scheduling university classes as it allows you to input multiple constraints such as room sizes, time slots, professor schedules, and student preferences.
- **Documentation**: OptaPlanner

2. Untis (WebUntis API)

- **Best For**: Class and room scheduling at schools and universities.
- **Features**:
  - Allows creating, managing, and adjusting timetables.
  - Handles both class and room schedules with integrated resource management.
  - Supports automated timetabling with conflict detection.
- **Use Case**: Suitable for large institutions where multiple courses, rooms, and instructors must be managed.
- **Documentation**: WebUntis API

3. Timetabler (Edval)

- **Best For**: Educational institutions.
- **Features**:
  - Advanced scheduling algorithms for university courses.
  - Manages constraints like student subject choices, teacher availability, and room requirements.
  - Can optimize for minimal clashes and best fit for resources.
- **Use Case**: Ideal for creating class schedules that are optimized for both students and faculty.
- **Documentation**: Edval Timetabler

4. Classroombookings API

- **Best For**: Room and resource scheduling.
- **Features**:
  - Simple API for room and resource booking.
  - Handles conflict resolution and availability checking.
- **Use Case**: Good for handling room assignments and availability when scheduling university classes.
- **Documentation**: Classroombookings

5. Unitime

- **Best For**: Large-scale university scheduling.

- **Features**:
  - Open-source solution for course timetabling, exams, and student scheduling.
  - Handles complex constraints including room, teacher, and student availability.
  - Highly customizable for university needs.
- **Use Case**: Unitime is ideal for universities that need a customizable, scalable timetabling solution.
- **Documentation**: Unitime

Using an Existing API Vs. Writing Our Own

- Existing API probably more robust and has more features.
- We can customise our own api to better fit client needs.
- I think I could write my own faster than it would take me to figure out how to connect the existing api.
- Would we get more marks for writing our own? (Probably not)
- I would have more fun writing my own lol.

Writing Our Own

Brute force doesn’t seem at all feasible. I think we will have to use a heuristic, such as:

- Simulated annealing (I have experience with this)
- Genetic algorithms

Fitness function:

- Non-negotiable constraints:
  - Can’t have two classes in same room at same time.
  - Can’t have teaching staff or student allocated two classes at same time.
  - etc.
- Other factors:
  - Nicer to start at whole increments of hours e.g 10am >> 10:15
  - Preferred start time e.g. 12pm >> 8am

Biggest challenge is how to weight factors.

How to represent a candidate timetable:

- Can’t just constantly read and write to the db.
- 2d matrix with rows corresponding to a time window and cols corresponding to a room. A single class would span across multiple rows. Store other constrains in other data structures e.g. hashmap of which classes can’t run at the same time.

Supabase offers support for 8 languages, of those I think we should consider:

- Python
  - What we’re the most familiar with.
  - Might be able make use of sklearn (although I think it might be hard to get the data into a suitable form).
  - Slow. Performance matters a lot because it allows us to consider more candidate timetables.
- Js
  - Already using for frontend.

    - Also slow.
- C#
    - By the far the fastest.
    - Seems hard to deploy.

Leaning towards Js