📅 UMT Timetable Scheduler: Features Document

🚀 1. High-Level Features:

1.1 Upload & Read Excel Files

- **Main Excel** (must include sheets like **Roadmap**, **Rooms**, **StudentCapacity**, and optionally **Electives** & **SpecialLabs**)
- **Cohort Excel** (optional for special courses that need "cohort" constraints)

The app reads these Excel files to figure out:

- **Which courses** are offered in each semester
- **How many rooms** are available and their **types** (theory or lab)
- **How many students** are in each semester
- **Special labs** needed for particular courses (if any)
- **Electives** courses, each with possible "theory" or "lab" scheduling options

1.2 Automated Timetable Generation

- Generates a full timetable for **main courses** by semester and by section.
- Generates an **electives** timetable after the main timetable is complete (if electives are defined).

1.3 Room & Day/Timeslot Management

- You can **add or remove rooms** right inside the app.
- The system tracks **used vs. free timeslots** in each room (so it knows how many slots remain).

1.4 Cohort Course Handling

- Special “cohort” courses can be scheduled on specific days/times that are **predefined** in a separate **Cohort Excel**.
- Each cohort entry can have its own capacity and timeslots.

1.5 Outputs & Downloads

- After generating, you can **download** Excel files:
    - **Main Timetable** for all scheduled semesters/sections
    - **Remaining Capacity** for each room (free vs. used timeslots)
    - **Electives Timetable** (for electives specifically)
    - **Remaining Capacity After Electives**

Example: After scheduling, you see that **Room A** is used 10 timeslots across Monday–Friday and still has 5 free slots. You can see this in the downloaded “Remaining Capacity” workbook.

🔍 Consistency Checks:

Before building the schedule, the app performs several **validations** to ensure the input data makes sense. These are carried out in the `consistency_check.py` and some in `data.data_io`:

1. **Required Sheets Check**
    - The **main Excel** must contain:
        - `Roadmap`
        - `Rooms`
        - `StudentCapacity`
    - Optionally can have:
        - `Electives`
        - `SpecialLabs`
    - Any missing sheet? The app stops and shows an error.
2. **Roadmap Sheet Validation**

- Checks for columns: `semester`, `course_code`, `course_name`, `is_lab`, `times_needed`
- Ensures `course_code` is unique within each semester.
- Enforces `is_lab` be **"true"** or **"false"** only.
- Prevents `times_needed = 3` because we only allow 1, 2, or other custom times, but not 3.

3. **Rooms Sheet Validation**
    - Checks columns: `room_name`, `room_type`
    - Room type can only be **"theory"** or **"lab"**.
    - Room name can’t be blank.
4. **StudentCapacity Sheet Validation**
    - Checks columns: `semester`, `student_count`
    - Makes sure student counts are **positive integers**.
    - Each semester only listed **once**.
5. **Electives Sheet Validation** (if present)
    - Checks columns: `elective_code`, `elective_name`, `sections_count`, `can_use_theory`, `can_use_lab`
    - Ensures `sections_count` > 0.
    - **Prevents** both `can_use_theory` and `can_use_lab` from being the same (they must differ, e.g. an elective is either strictly theory or strictly lab or can handle both).
6. **Cohort File Validation**
    - Checks for columns like `CohortSemester`, `CourseCode`, `CourseName`, `Section`, `Capacity`.
    - Ensures capacity is **positive**.
    - Splits any row with capacity > 50 into multiple sub-sections, e.g. if capacity = 120, it splits into sections of 50, 50, and 20.
7. **Capacity Consistency**

- If a main course is declared as “cohort-based” in the main Excel (by pairing it in the Cohort Excel), the sum of all cohort section capacities must meet or exceed the total seats needed.
- **Example**: If semester 3 has 100 students ⇒ 2 sections of 50 each ⇒ total seats needed = 100. The cohort file for that course must have enough capacity across its cohort sub-sections (e.g. 60 + 50 = 110, which is enough).

If **any** check fails, the app **stops** and displays an error message so you can fix your Excel files first.

⚙️ Constraints Covered by the Solvers:

Main Courses Solver:

The main scheduling magic happens in `scheduling/solver.py` with the function `schedule_timetable(...)`. Here are the key constraints in **plain language**:

1. **Room Availability**
    - A room (theory or lab) can only host **one class** in a given timeslot/day.
    - If a room is already used in the previously saved usage data, it’s not available for that timeslot.
2. **Timeslots**
    - We have **theory timeslots**: 8:00–9:15, 9:30–10:45, 11:00–12:15, 12:30–1:45, 2:00–3:15, 3:30–4:45, 5:00–6:15.
    - We have **lab timeslots**: 8:00–10:30, 11:00–1:30, 2:00–4:30, 5:00–7:30.
    - Theory timeslot **#3 is skipped on Fridays** (to avoid a midday conflict, e.g. prayers/lunch).
3. **Number of Sessions Needed**
    - If a course says `times_needed = 2` (e.g., 2 sessions of theory per week), the solver ensures it is scheduled exactly **2 separate days**.
4. **No Consecutive Day Theory Sessions**

- If a theory course needs 2 sessions, those sessions **cannot** be on back-to-back days (e.g., Monday & Tuesday). They must have a day gap at least.

5. **No Overlap in a Single Section**
    - A single section can't have two classes at the same day/time.
    - If you have a lab slot, it also prevents overlap with the corresponding theory timeslots that clash in minutes (for example, a 2.5-hour lab overlaps with multiple theory slots).
6. **Special Labs**
    - If the course is marked as needing a special lab, it **only** gets scheduled in **those** specific rooms (listed under "SpecialLabs").
7. **Cohort Courses**
    - If a course is listed in the **cohort file**, it gets scheduled at **fixed** times/days from that file.
    - The solver ensures that capacity in those cohort sections can accommodate the standard sections in the main timetable.
    - The app merges them so that a normal "Section S3A1" plus "Cohort C08-A" does not overlap timeslots in any impossible way.
8. **Student Day Span**
    - Each section's classes in a single day cannot exceed an **8-hour span** from the first to the last class.
    - For example, if a section has a class at 8:00 AM and another at 5:00 PM on the same day, that's a 9-hour gap — not allowed. The app ensures it stays within an 8-hour range.

Example:

If **BSAI Semester 1** has 50 students, it forms **Section S1A1**. The app sees it needs **2 theory sessions** for "Intro to Management" and 1 lab session for "Computer Applications." It ensures:

- The 2 theory sessions are on separate, non-consecutive days (e.g., Monday & Wednesday), each in a theory room.

- The lab session is on a different day (e.g., Friday) in an available lab slot, possibly skipping the 12:30–1:45 Friday slot for theory.
- No daily time overlaps for that single section.

Electives Solver:

Electives scheduling is handled in `scheduling/electives_solver.py`. Key points:

1. **Choice of Theory vs. Lab**
    - Each elective course is flagged if it "**can use theory**" or "**can use lab**."
    - The solver decides which type to use based on availability (the course can't be both).
2. **Timeslots**
    - If it picks **theory**, it will schedule **2** distinct (non-consecutive) days/timeslots.
    - If it picks **lab**, it schedules **1** lab slot.
    - **Again**: timeslot #3 for theory is skipped on Fridays.
3. **Room & Timeslot Availability**
    - The solver checks leftover free slots after the main timetable.
    - No double-booking.

If it **cannot** find a feasible solution for some electives due to not enough free timeslots, it will let you know so you can add more rooms or reduce sections.

Example:

Elective "Creative Writing" might say:

- `can_theory = True`, `can_lab = False`, `sections_count = 2`
  This means we need 2 different sections, each must have 2 theory sessions in free theory slots that are not on consecutive days. If the solver can't find enough open theory slots, it fails.

📈 4. Future Possibilities (What Could Be Added)

1. **Teacher Availability**:
    - Let’s say certain instructors only teach on certain days. We can expand the solver constraints to handle teacher availability.
2. **Preferred Times**:
    - Some courses prefer morning or afternoon slots. This can be added as a “soft constraint.”
3. **Max Classes per Day**:
    - If we want to limit a section to a maximum of 3 classes/day, we can add that constraint.
4. **Evening Classes**:
    - If the university might have 7 PM–9 PM slots, these can be extended easily in the timeslots definitions.
5. **Automatic Room Capacity**:
    - So far, we only do “room type” matching, not actual seat counts. If needed, we can tie each room with capacity to ensure a room has enough seats for that section.
