Assessment Timetable Scheduler

Technical Specification

Dublin City University

COMSCI3

Student: Jingyuan Wu | Supervisor: Darragh O’Brien

1. Introduction

1.1 Overview

This document covers the design, implementation, and testing of the Assessment Timetable Scheduler — a desktop application that helps DCU module coordinators schedule 4th-year project presentations. I used Google OR-Tools because it supports constraint programming and was suggested in lectures.

The motivation is that scheduling these assessments by hand is slow and easy to get wrong. When you have 20+ projects and each one needs a panel of 2-3 lecturers who all have different availability, working out a valid timetable manually can take hours. This project tries to automate that.

1.2 Objectives

- Build a GUI that allows coordinators to enter all data without editing JSON files directly
- Implement a CP-SAT solver that handles hard constraints: room capacity, lecturer availability, panel requirements, supervisor assignment
- Add a weighted soft objective so the output is a good schedule, not just any valid one
- Provide JSON and CSV export so the output can be shared or processed further
- A command-line interface for scripted/batch use

1.3 Scope and Limitations

This system is built specifically for DCU 4th-year project presentations. It does not handle exams, lab assessments, or other institutions. The solver is designed to perform well on typical DCU-scale inputs (roughly 10-30 projects, 5-15 lecturers, 20-40 timeslots). Very large inputs may be slow due to the growth in CP-SAT model variables.

2. Requirements

2.1 Functional Requirements

ID	Requirement	Priority
FR-01	Add, edit and delete lecturers; mark availability per timeslot	Must have
FR-02	Add, edit and delete students; assign to projects	Must have
FR-03	Add timeslots individually or in batches (full day generation)	Must have
FR-04	Set panel size, supervisor constraint, number of rooms	Must have
FR-05	Run CP-SAT solver and display results in a table	Must have
FR-06	Show plain-English error messages when schedule is not feasible	Must have
FR-07	Export result as JSON or CSV	Should have

ID	Requirement	Priority
FR-08	Mark certain slots as lunch slots to be penalised by the solver	Should have
FR-09	Soft objective to balance panel workload across lecturers	Should have
FR-10	Command-line interface for batch use	Could have

2.2 Non-Functional Requirements

- Solver should return a result within the configured time limit (default 10 seconds)
- GUI should not freeze while the solver is running
- Application should run on Windows, macOS, and Linux without changes
- The JSON config format should be readable without the application

3. System Architecture

3.1 Layer Overview

The code is split into distinct layers so each part can be tested and changed independently. The GUI and CLI both use the same solver and model code.

Layer	Module(s)	Responsibility
Data model	models.py	Python dataclasses for all domain objects
File I/O	io_json.py	Load/save JSON with structural validation
Pre-solve checks	solver/precheck.py	Catch obvious errors before solver runs
Solver	solver/slice1.py, slice2.py, slice3.py, api.py	CP-SAT constraint models
GUI	ui_tk/app.py and tabs/	tkinter four-tab interface
CLI	cli.py	Command-line wrapper using same solver code

3.2 Data Model

All domain objects are Python dataclasses (Python Software Foundation, 2024). Entities are kept flat and linked by ID string rather than nested, following supervisor feedback.

Class	Key fields	Notes
TimeSlot	id, date, start, end, label	frozen=True so it cannot be modified after creation
Lecturer	id, name, available_slot_ids, max_per_day	max_per_day limits how many panels per day

Class	Key fields	Notes
Student	id, name, unavailable_slot_ids	Unavailable slots are blocked in the solver
Project	id, title, student_ids, supervisor_lecturer_id	Links by ID, not nested objects
Constraints	rooms, panel_size, lunch_slot_ids, weights, solver	All solver parameters together
Config	meta, timeslots, lecturers, students, projects, constraints	Top-level container

3.3 Solver Design

Three Slices

The solver has three levels. Slice 1 is the simplest. Each slice adds more variables and constraints. This approach made debugging easier — slice 1 could be verified on its own before adding panel assignment.

Slice	Variables added	Constraints added	Objective
Slice 1	x[p,t,r] — project p in slot t, room r	Each project once; at most 1 project per room/slot; student unavailability	Minimise last slot used
Slice 2	y[p,l] — lecturer l on panel for project p; z[p,l,t,r] = x AND y	Panel size; supervisor in panel; no lecturer double-booked; availability; max_per_day	Same as slice 1
Slice 3	count[l], max_c, min_c, imbalance, lunch_penalty	All of slice 2	Weighted: span + workload balance + lunch penalty

Binary Product Linearisation

The z = x AND y conjunction is linearised using three standard linear constraints (Williams, 2013, p.147):

- $z <= x$
- $z <= y$
- $z >= x + y - 1$

This guarantees z = 1 if and only if both x and y are 1, with no non-linear terms required.

Workload Balance

Slice 3 counts how many panels each lecturer appears in, then minimises (max count - min count). OR-Tools add_max_equality and add_min_equality are used for this (Google LLC, 2024). This avoids the need for a big-M formulation.

4. Implementation Notes

4.1 Technology Stack

Component	Technology	Version
Language	Python	3.10+
Solver	Google OR-Tools CP-SAT	9.9+
GUI toolkit	tkinter / ttk	Standard library
Testing	pytest	9.0+
Packaging	setuptools	68+
CI	GitHub Actions	—

4.2 Availability Grid

The availability tab uses a Canvas widget instead of Button widgets. With 10 lecturers and 20 timeslots, that is 200 widgets, which is slow in tkinter. The canvas draws rectangles and uses mouse event coordinates to detect clicks (Stack Overflow, 2014). The canvas.canvasx() and canvas.canvasy() calls are needed to convert from screen coordinates to canvas coordinates after scrolling.

4.3 Pre-solve Checks

The precheck module runs before the solver to catch obvious problems and give plain-English error messages. This includes: not enough room/slot capacity for the number of projects; panel size bigger than number of lecturers; supervisor with no available slots; references to unknown slot IDs. Without this, the solver just returns INFEASIBLE with no explanation.

4.4 Known Issue During Development: Supervisor Bug

The early prototype (legacy/demo.py) had the supervisor constraint written as model.add(y[supervisor] == 0), which forces the supervisor out of the panel — the opposite of the requirement. This bug was found during slice 2 testing and is documented in legacy/README.md. The fix is model.add(y[supervisor] == 1).

5. Testing

5.1 Test Summary

Automated tests use pytest. There are 29 tests in total. GitHub Actions runs all tests on every push.

Module	Tests	Coverage
test_io.py	5	JSON roundtrip, missing key, null meta, duplicate IDs, old num_search_workers field
test_precheck.py	6	Valid config passes, capacity error, panel too large, empty supervisor slots, unknown lunch slot, ensure_ok raises
test_solver_slice1.py	4	Feasibility, all projects scheduled, no room conflicts, student unavailability
test_solver_slice2.py	7	Feasibility, all projects scheduled, panel size, supervisor in panel, availability, infeasible, max_per_day
test_solver_slice3.py	7	Feasibility, all projects, supervisor, lunch avoidance, zero weights, stats returned, infeasible

5.2 Selected Test Cases

test_roundtrip

Creates a Config object, saves it to a temporary JSON file, loads it back, and checks that all key fields are preserved. This catches any serialisation or deserialisation bugs.

test_slice2_supervisor_in_panel

Runs slice 2 on a 2-project config where must_include_supervisor is True. Checks that for every entry in the result, the project's supervisor ID appears in panel_lecturer_ids. This test was added after the supervisor bug (Section 4.4) was found in the prototype.

test_slice3_lunch_avoidance

Sets lunch weight to 50 (high) and span weight to 0. With 2 projects and 3 timeslots, the solver has enough room to avoid the lunch slot completely. The test checks that no entries are assigned to the designated lunch slot.

test_slice2_infeasible_when_no_availability

Clears all lecturer availability. must_include_supervisor is set to False first, otherwise the precheck layer would raise a PrecheckError before the solver even runs. Then checks the solver returns INFEASIBLE. This distinction was discovered while writing the test — a note is in the test file.

5.3 User Testing

I asked one classmate to try the system without instructions. The participant was given no instructions and asked to complete a series of tasks.

Task	Outcome	Notes
Create slots using the batch generator	Completed	Participant found the button easily and used the default values
Add two lecturers and mark availability	Completed with confusion	Clicked header row first. Fixed when realised only cells respond to clicks
Add a project and set supervisor	Needed two attempts	Forgot to click 'Set supervisor' after adding the project the first time
Run the solver (Slice 3)	Completed	Result appeared in about 1 second. Status showed OPTIMAL
Export result as CSV	Completed	File opened correctly in Excel. Column headers were clear
Feedback	—	Participant asked what 'panel size' means — suggested adding a tooltip

Overall the tasks were completed successfully. Main usability problem was that the availability grid requires entities to be added first, which is not obvious. A tip was added to the welcome screen to address this.

6. Known Limitations and Future Work

- No undo/redo — if a lecturer is deleted by mistake, it must be re-added manually
- Timeslots are not checked for overlap within the JSON
- Large inputs (30+ projects, 15+ lecturers) may be slow because CP-SAT variable count grows with P x L x T x R
- No GUI integration tests — automated tests only cover the backend

Possible future improvements: undo/redo using a command pattern; import availability from CSV; visual day-view of the timetable; multi-day schedules with date-range selector.

7. References

Google LLC (2024) OR-Tools CP-SAT Python API Reference. Available at: https://developers.google.com/optimization/reference/python/sat/python/cp_model [Accessed: February 2026]

Google LLC (2024) OR-Tools GitHub repository. Available at: https://github.com/google/or-tools [Accessed: February 2026]

Python Software Foundation (2024) dataclasses — Data Classes. Python 3.12 Documentation. Available at: https://docs.python.org/3/library/dataclasses.html [Accessed: January 2026]

Python Software Foundation (2024) json — JSON encoder and decoder. Python 3.12 Documentation. Available at: https://docs.python.org/3/library/json.html [Accessed: January 2026]

Stack Overflow (2014) Adding a scrollbar to a group of widgets in tkinter. Available at: https://stackoverflow.com/questions/3085696 [Accessed: January 2026]

Stack Overflow (2012) How to create binary variables in GLPK. Available at: https://stackoverflow.com/questions/10792603 [Accessed: January 2026]

Williams, H.P. (2013) Model Building in Mathematical Programming. 5th edn. Chichester: Wiley.