Final Exams

Planning, seating, staffing and publishing
an examination session

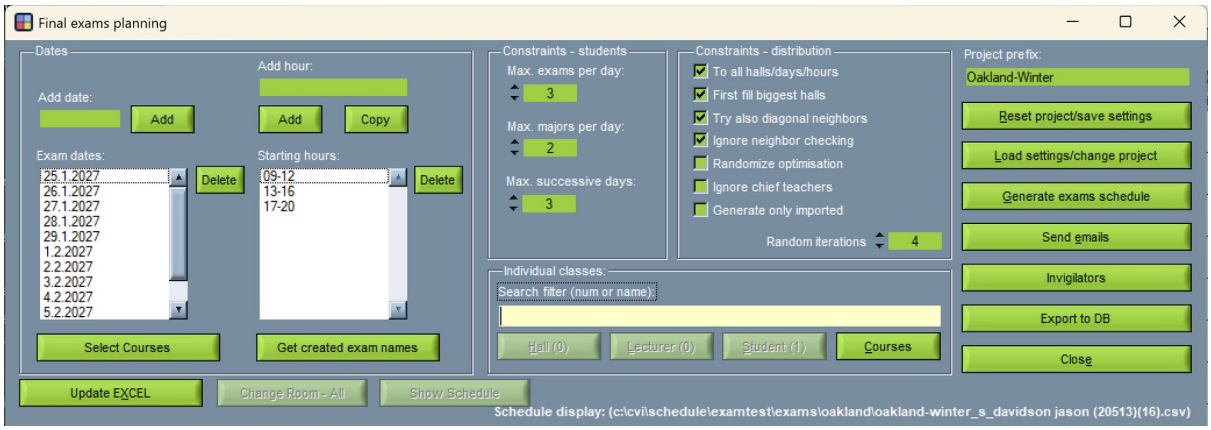
. . .

WISE TIMETABLE · AUGUST 2026

Contents

Final exam planning is a **separate scheduler**. It does not use the weekly timetable, it does not place lessons, and it does not work from groups. It takes a list of exams, a list of dates and starting hours, and a set of exam halls, and it seats **every student individually** — by row and column, in a named hall, on a date, at an hour. It then staffs each hall with invigilators and can send every person their own schedule by e-mail.

Open it with **Tools | Plan final exam**.



Part 1 — Getting the data right

Most first attempts produce an empty or half-empty plan, and almost always for one of the reasons in this part. None of them is set on the Final Exams screen itself: exam planning reads data that lives on the Rooms, Courses and Students screens. Get these four things right and the planning itself is quick.

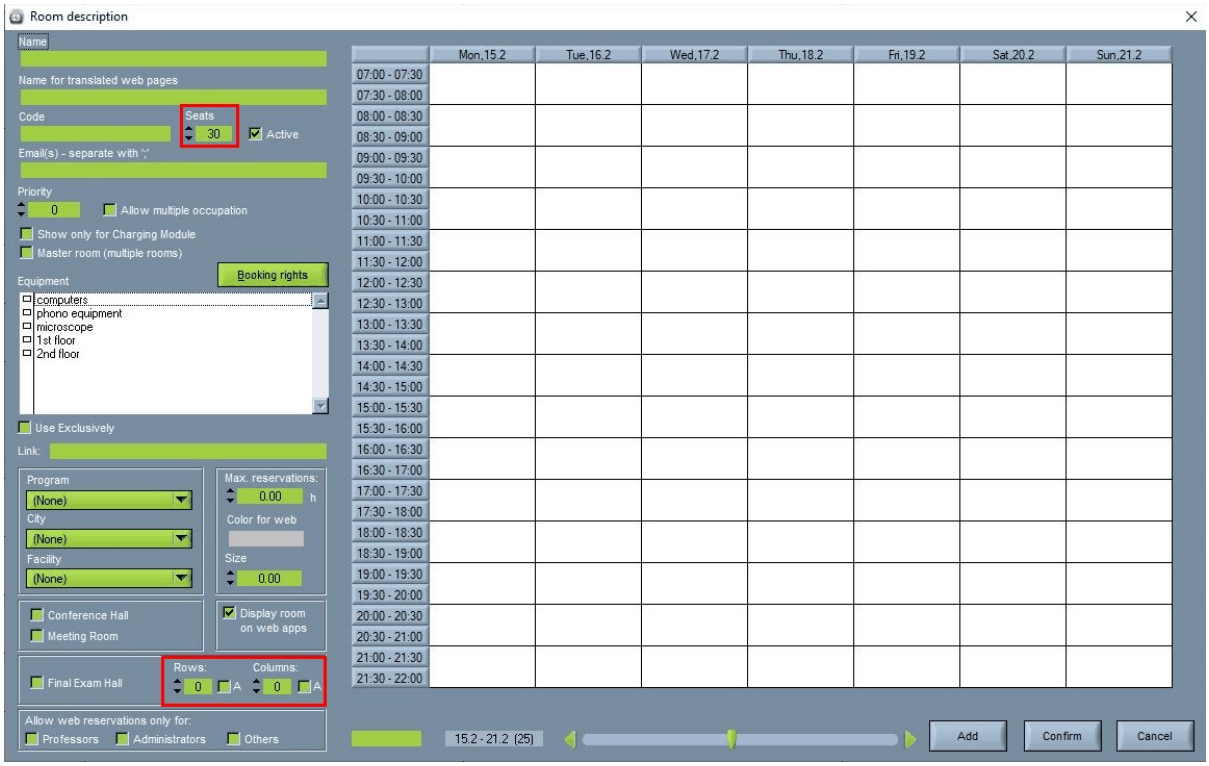
1.1 What the planner reads

It needs	Where it comes from
Exam halls and their seat grids	**Rooms** — *Final exam* tick and the seat columns/rows
Which courses sit an exam	**Select Courses** on the exam screen
Who sits each exam	**Student course registrations**
Who supervises each exam	The course's **exam supervisor** (chief teacher)
Which courses are principal subjects	The course's **Course is major** flag

Note what is *not* in that list: **groups**. A student who belongs to a group but is not registered on a course is not part of that course's exam. Group membership drives the weekly timetable; course registration drives exams. This is the single most common surprise when a school first opens the exam planner on a database that has been used for timetabling for years.

1.2 Marking rooms as exam halls

On the **Rooms** screen, open the room and tick **Final exam**. Then fill in the seat grid — the number of seat **columns** and seat **rows**.



The grid is a map of the desks, and the planner uses it literally: a student is seated at a column and a row, and that pair is printed on the seating chart and on the student's own card. A hall with no grid, or a grid of zero, is not used.

The grid is deliberately independent of the room's teaching capacity. A hall that seats ninety for a lecture holds far more desks once they are laid out in rows for an exam, and a lecture theatre with fixed benches may hold fewer. Measure the exam layout, not the lecture layout.

Up to **32 halls** can be marked. Beyond that the rest are ignored and the application log says so.

1.3 The seat grid is not how many students fit

This is the most useful single fact in this manual. Two students sitting the *same* exam are never seated next to each other, so one exam fills a pattern rather than the whole grid:

Neighbour setting	One exam can use	A 600-seat hall takes
Try also diagonal neighbors ticked	half the grid	300 students
neither neighbour option ticked	a quarter of the grid	150 students
Ignore neighbor checking ticked	the whole grid	600 students

Students sitting *different* exams may sit side by side — they are not writing the same paper — so a hall can still be filled by several exams at once.

The working rule, with the recommended settings:

> **The largest exam you can place is half of your largest hall.** An exam of 270 students needs a hall of at least 540 seats.

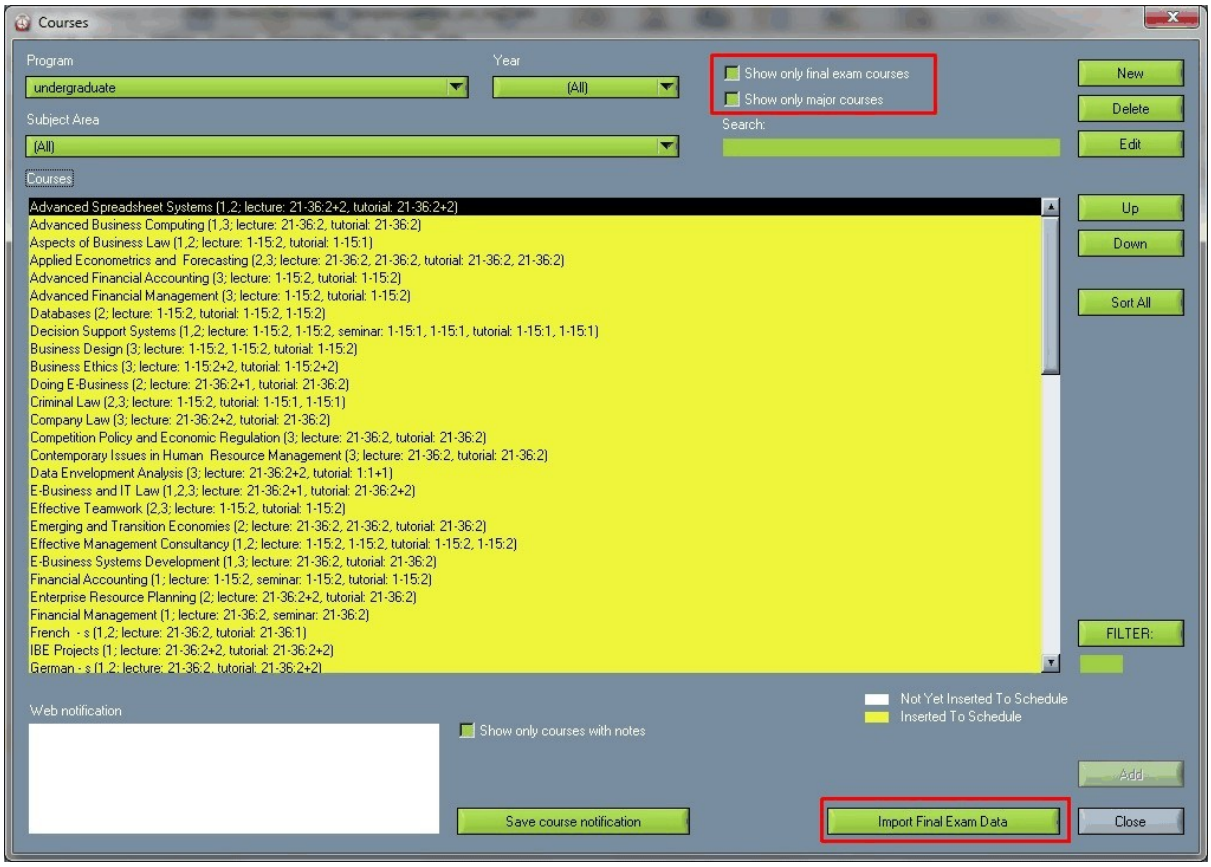
If exams keep coming back unplaced and every one of them is larger than half your biggest hall, no scheduling option will help. The halls are too small for the way you are seating people.

1.4 Preparing the courses

Which courses sit an exam is decided with **Select Courses** on the exam screen. It opens the session's course list in a text editor — one line per course, beginning with `1` for an exam and `0` for none:

```
1;Financial Accounting        (1:Accounting for Management)
@18298234-c433-…
0;International Perspectives  (2:Accounting for Management)
@9d6dfa6d-f10c-…
```

Edit the leading digit, save, close. The list is stored with the project as `<prefix>-c_list.csv`, so each session can examine a different set of courses — a January session and a resit session are two projects over the same database.



The exam supervisor of a course — its chief teacher — is used twice: the person is written on the exam as its supervisor, and the planner will not put one supervisor in two different halls in the same period. **Ignore chief teachers** on the exam screen switches that rule off.

Major courses. In the same **Final Exam Settings** dialog a course can be ticked **Course is major**, and a course part **Course part is major**.

This is a property of the *course*, not of the exam. It marks the course as a **principal subject** — one of the ones a student's programme is built around — and the same flag is used elsewhere in the application: **Show only major courses** filters the course picker by it, and a major course is listed with an `M` beside its name.

The exam planner then treats it as a second, tighter limit. *Max. majors per day* counts only these, so a student is not made to sit two of their main subjects on one day even where the general limit would allow it.

Mark the subjects that carry the programme, not simply the ones that are hard. Marking everything major makes the limit meaningless and the session much harder to place.

Course parts with their own exam. A part of a course — lectures, seminar, tutorial — can be flagged to sit its own exam. It then appears as a second exam entry for that course, and the students who are registered *on that part* sit both it and the course exam. Use it where the practical and the written paper are genuinely separate sittings.

1.5 Student registrations — the part that is usually missing

A registration is a pair: **student** → **course**, optionally narrowed to a **course part**. Registrations can be entered by hand on the Students screen, imported from CSV, or brought in from a student information system.

By hand: open the student, and tick the courses. For a whole branch at once, the Students screen can add a branch's default courses to a student who has none.

1.6 Importing students and registrations from CSV

This is the fastest way to make an existing database exam-ready, and it is what a school with a student information system will use.

Data | Import students from CSV, or the import button on the Students screen. The file is read line by line; a line beginning with # is a comment.

The main format

```
studentNumber;year;surname;name;email;group;ldapName;code1;code2;code3;…
```

Field	Meaning
`studentNumber`	enrolment number — the key the import matches on
`year`	year of study
`surname`, `name`	as they should appear on the seating chart
`email`	needed only if the student is to receive their schedule
`group`	timetable group; may be left empty
`ldapName`	web login name; may be left empty
`code1…`	**one field per course registration** — any number of them

Everything from the eighth field onwards is a **course registration**. Each is matched first against the **course code**, and if no code matches, against the **course name** exactly. A code that matches neither is skipped silently, so check a small file first.

To register a student on one **part** of a course rather than the whole course, put the execution type's code in brackets:

```
20001;1;Smith;Emily;emily.smith@example.edu;BAM1-1;;FIN101;ECO102;ECO102(LAB)
```

That student sits Financial Accounting, Macroeconomics, and the laboratory part of Macroeconomics as its own exam.

> When a line carries any course fields at all, the student's existing registrations are **replaced**, not added to. A file with no course fields leaves existing registrations alone. This is what makes a re-import safe at the start of a session and destructive in the middle of one.

The short format

A line whose second field is not a number is read as:

```
ID;Name;Surname;Email;Program;Year;Subject Area;Short Name
```

This carries no registrations — it creates or updates the people only.

After importing

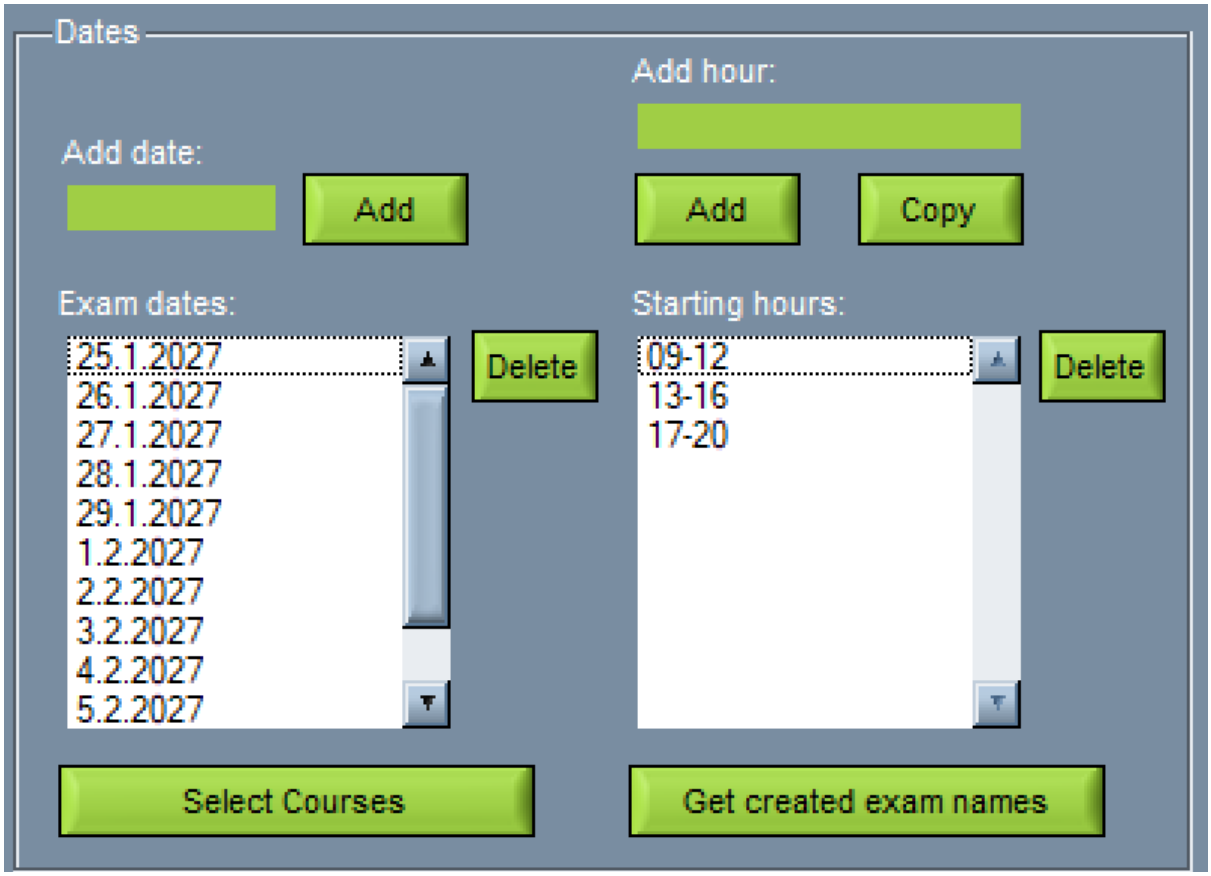
Check three things before planning:

1. **A course that nobody is registered on** is skipped, and the generate reports how many such entries it found. If that number is not zero, either the course should not be in the exam list, or its codes did not match.
2. **Enrolment counts per exam** are shown in the result table. A course with far fewer students than expected usually means some registrations matched a course *name* that differs by a space or a dash.
3. **No more than 768 students** may be registered on one exam, and no more than 16384 students may take part in one session. Both limits are written to the log if they are reached.

Part 2 — Planning the session

2.1 Dates and starting hours

Dates and hours are the left-hand half of the screen. The two lists are independent: **Exam dates** holds the days, **Starting hours** holds the hours of the date selected beside it.



Add date — type the date and click **Add**; it appears in **Exam dates**. Select one and click **Delete** to remove it.

Write the date as **day.month.year with no spaces** — `25.1.2027`, never `25. 1. 2027`. The date is read back as a single word, and the successive-days rule works out which dates are calendar neighbours by reading it, so a malformed date silently weakens that rule.

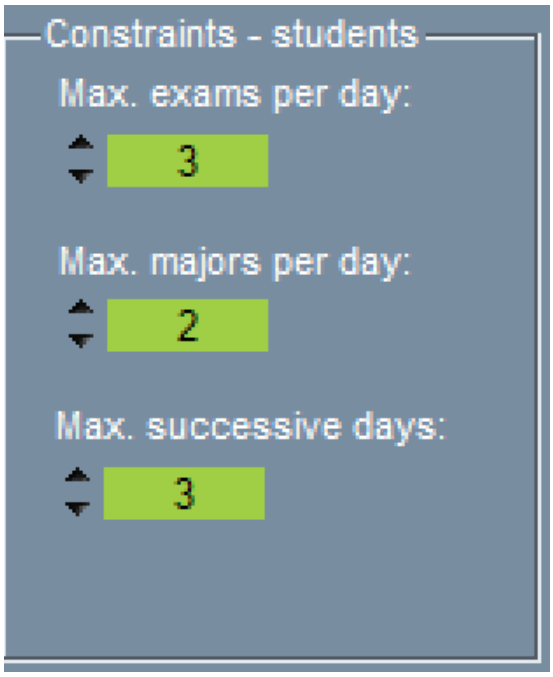
Add hour — select a date, type the starting hour, click **Add**. One date can carry several starting hours. **Copy** gives the selected date the same hours as the previous one — the quick way to build a two-week session. Select an hour and **Delete** to remove it.

A date and a starting hour together make a **period**, and the period is the unit the planner places exams into. Twelve dates with three hours each is 36 periods.

Avoid a colon in the hour. The hour becomes part of the name of every result file, where a colon is not allowed, so `09:00` is stored as `09-00` and the screen and the file names stop agreeing. `09-12` or `0900` reads the same everywhere.

Maximum **12 dates** and **5 starting hours per date**.

2.2 Constraints — students



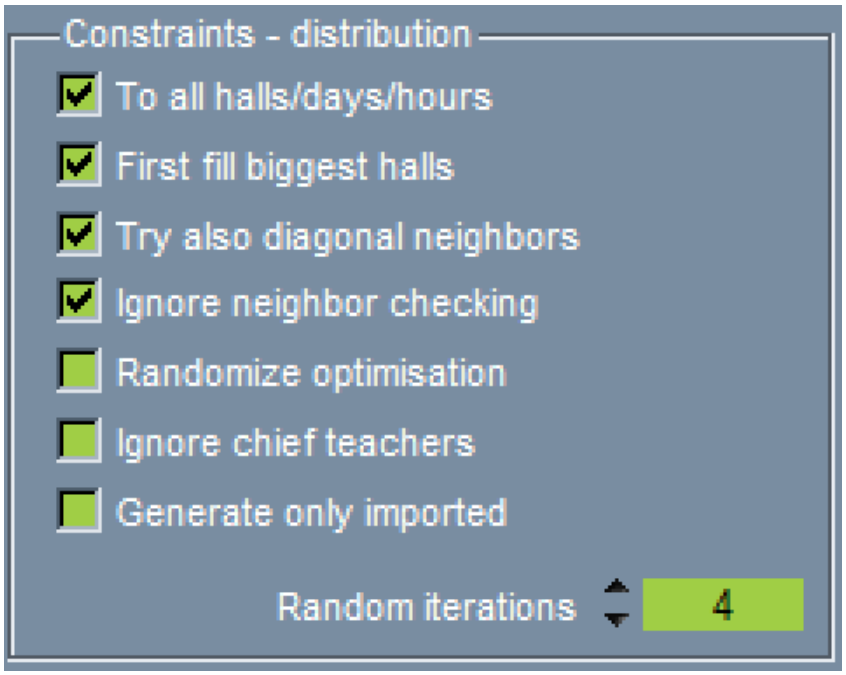
These three protect the student and are the usual reason a session is tight.

Max. exams per day — how many exams one student may sit on one date, no matter the hours.

Max. majors per day — the same limit counted only over courses marked **Course is major** (see 1.4). It is what keeps two principal subjects off the same day; 1 is the usual setting.

Max. successive days — the longest run of **consecutive calendar days** on which one student may have exams. Set to 3, it permits Monday–Tuesday–Wednesday and refuses the Thursday. It is a rest rule, not a counting rule: two exams in one day are governed by *Max. exams per day*. A weekend breaks a run, because the rule reads the real calendar.

2.3 Constraints — distribution



To all halls/days/hours — plan the whole session at once, choosing freely among all periods. With it off, the planner works period by period in order, filling each before moving on.

First fill biggest halls — send each exam to the **largest hall that can hold it**, and let several exams share a hall. With it off, exams go to the **emptiest** hall instead, which spreads them out. Note what "largest" means in practice: the biggest hall keeps winning while it has room, so the smaller halls are overflow rather than an equal choice. That is usually what a school wants, and it is why a session can run with only one hall in use.

Try also diagonal neighbors — allows two students of the same exam to sit diagonally, banning only the four orthogonal neighbours. **Leave this ticked** unless you have a reason not to: with it off, diagonals are banned as well and every exam is limited to a quarter of its hall.

Ignore neighbor checking — no neighbour rule at all; an exam may use every seat. Only sensible where seating is not an anti-copying measure.

Randomize optimisation with **Random iterations** — run the search several times from different orders and keep the best result. With it off the search runs once, deterministically, taking the largest exams first — which is often the better plan, because large exams have to be placed early.

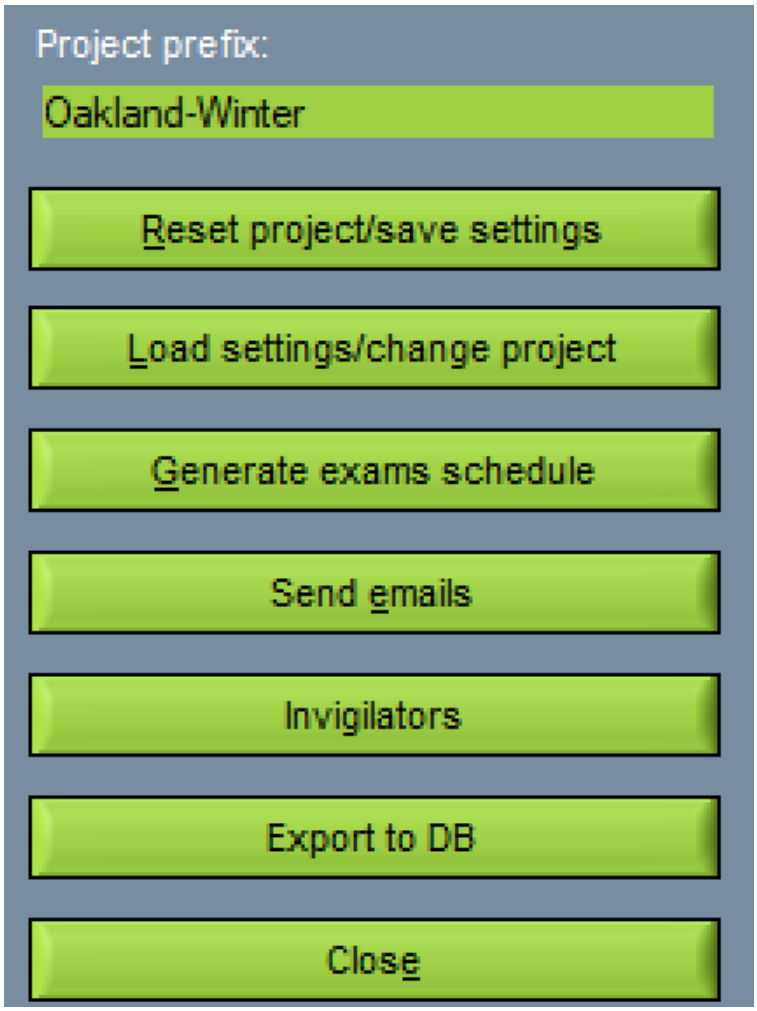
Ignore chief teachers — drops the rule that one supervisor cannot be in two halls at once.

Generate only imported — plans only exams that came from an external system.

2.4 Generating

Name the session in **Project prefix**, then click **Generate exams schedule**. The first generate asks where to keep the project; every file it writes goes there, named after the prefix.

Do not put an exam project in Program Files — Windows will not let the application write there. A folder under Documents, or on the school's shared drive, is the right place.



When it finishes, the application says what it produced: how many exams were placed, over how many dates and in how many halls, and — if some could not be placed — how many and how many student registrations they affect. The schedule appears in the table at the bottom of the window.

2.5 Reading the result

The four buttons under **Individual classes** re-cut the same session:

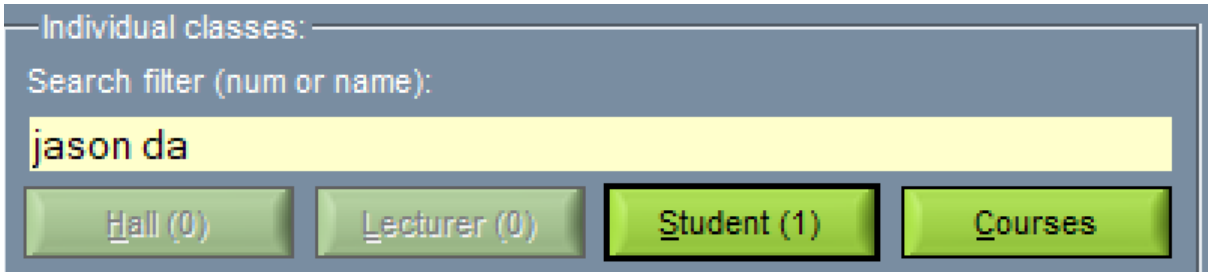
- **Courses** — one line per exam: students, hall, date and hour, supervisor, and `M` where the course is a major subject
- **Hall** — the seating chart of one hall in one period, student by student, with row and column
- **Lecturer** — one lecturer's invigilation duties
- **Student** — one student's personal exam card

The **student view** is the one to show a registrar, because it is what the student will hold on the morning of the exam:

	1.Course	2.Room	3.Seat Row	4.Seat Column	5.Day(Hour)	6.Major exam
1	International Business Environment	B-008	14	3	25.1.2027(09-12)	M
2	Aspects of Business Law	B-008	14	3	25.1.2027(13-16)	0
3	Enterprise Resource Planning	B-008	17	3	26.1.2027(13-16)	0
4	Total Quality Management	B-008	4	3	25.1.2027(17-20)	M
5	Psychology & Work	B-008	12	3	26.1.2027(09-12)	0
6	Written Business English	B-008	11	3	26.1.2027(17-20)	0
7	Physics	B-008	10	3	27.1.2027(09-12)	0
8	jason.davidson@students.example.com					

Course, room, **seat row**, **seat column**, day and hour, and `M` where the course is a major subject. The last line carries the student's e-mail address — the same address the file is sent to. Everything a person needs to walk into the right hall and sit in the right chair is on one page.

Finding one person. Type a name or a student number into **Search filter**. The four buttons then show how many matches each view holds — `Student (1), Hall (0)` — and clicking one opens it. It is the quickest way to answer "where do I sit?" at the counter.



Select a cell and click **Show Schedule** to open the individual schedule behind it — click a hall in the course view, for example, and the seating chart for that hall and period opens.

Anything you edit directly in the table needs **Update EXCEL** to be written back to the project files.

2.6 When exams cannot be placed

An exam that could not be placed shows **"too many/overlapping students"** instead of a hall. That one message covers several different causes; this table separates them.

What you see	What it really means	What to change
Every unplaced exam is larger than half your biggest hall	The seating pattern, not the calendar — no hall can hold that many with a free seat between them	A larger seat grid, **Ignore neighbor checking**, or split the exam
The large exams fail, the small ones are placed	The large ones were scheduled too late, when every period already held one of their students	Turn **Randomize optimisation** off, or raise **Random iterations**
A few exams fail regardless of size	Their students have no free period left	Add a date or a starting hour
Failures appear only when limits are tight	*Max. exams*, *majors* or *successive days* is binding	Relax one of the three, or add dates
Only one hall is ever used	Normal with **First fill**	Untick it to spread exams

What you see	What it really means	What to change
	biggest halls while that hall has room	across halls
An exam reports zero students	Nobody is registered on it	Fix the registrations, or take the course out of the exam list

Why large exams are the fragile ones. A period is refused for an exam if **any single one** of its students is already sitting an exam in that period, or is at a daily limit, or would break the successive-days rule. An exam with 240 students drawn from several branches touches so much of the cohort that, once the schedule is half full, every period contains at least one of them. Large exams have to go in early — which is exactly what the deterministic search does.

2.7 Changing a hall after generating

Select an exam and use **Change Room** to move it. The hall is rewritten in every file that mentions it, so the rest of the plan stays valid and nothing needs regenerating. This is how a room that turns out to be unavailable is handled a week before the session.

2.8 Saving, loading, resetting

Both live in the button column shown in 2.4.

Load settings/change project opens a saved session — pick its `<prefix>_settings.txt`; the project name and directory follow from it.

Reset project/save settings stores the session — and **deletes the previous files of the same prefix** before writing new ones. Use a new prefix whenever you want to keep an earlier plan.

Part 3 — Invigilators

An exam schedule that nobody is rostered to supervise is only half a plan. Two mechanisms work together: the planner assigns the course's own supervisor automatically, and the **Invigilators** screen lets you add the rest by hand.

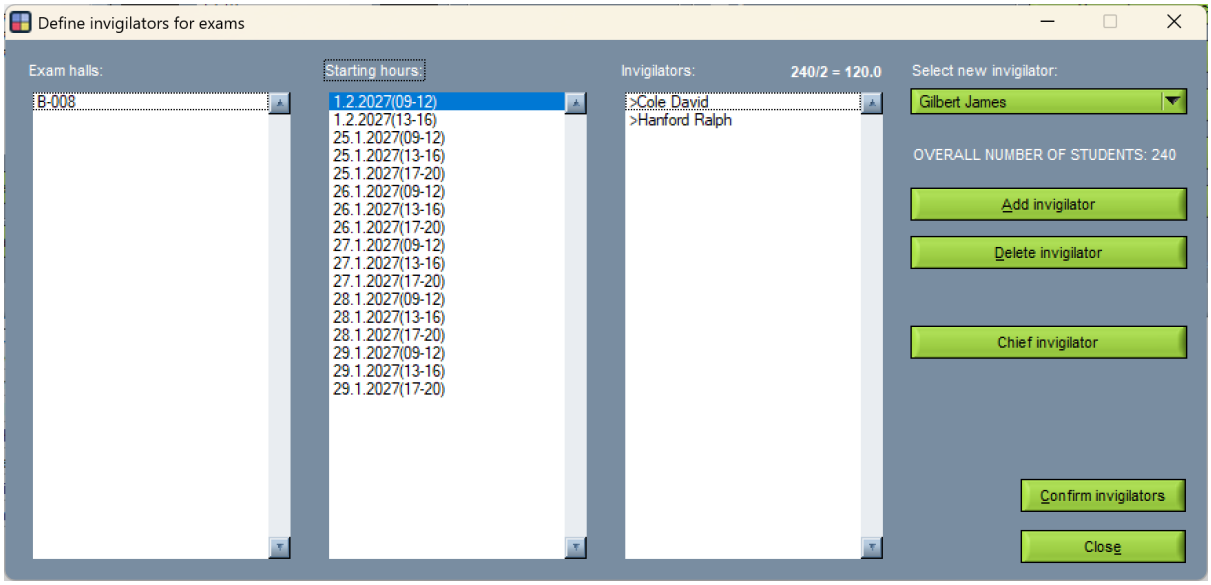
3.1 What the planner does on its own

Each exam carries its course's **exam supervisor**, and that person is treated as being present in the hall. The planner will not place two exams with the same supervisor in two different halls in the same period — one person cannot invigilate in two rooms at once. Several exams sharing a hall may share a supervisor; the rule only prevents being in two places.

Ignore chief teachers switches this off. Use it when supervisors are assigned centrally afterwards and the course's own lecturer is not expected to attend.

3.2 Adding invigilators

Click **Invigilators**. The window **Define invigilators for exams** opens, listing only the halls that are actually in use and, for each, the periods it is used in.



1. Select the hall under **Exam halls**.
2. Select the period under **Starting hours** — a hall is staffed per period, not per day, because the people change between sittings.
3. Pick a person under **Select new invigilator** and click **Add invigilator**.
4. Repeat for as many as the hall needs, then move to the next period.

Delete invigilator removes the selected one. **Chief invigilator** marks the person in charge of that hall — the one who holds the papers, reads the instructions and signs the attendance sheet. **Confirm invigilators** writes the assignment into the project files; nothing is stored until you do.

A name shown with a **>** in front of it is the course's own exam supervisor, placed there by the planner rather than by you. It **cannot be deleted** here while *Ignore chief teachers* is off — removing the person responsible for the paper is not an editing operation, it is a change

to the exam. Change the course's supervisor, or tick *Ignore chief teachers*, if that is what you mean.

3.3 How many invigilators a hall needs

Above the list, two numbers answer that at a glance for the selected hall and period:

- **OVERALL NUMBER OF STUDENTS** — how many people are sitting there
- the ratio beside **Invigilators** — students divided by invigilators, so `240/2 = 120.0` means each of the two is watching a hundred and twenty candidates

This is a count, not a rule. Wise Timetable does not enforce a ratio, because every institution has its own — but the number makes an understaffed hall visible weeks before the session rather than on the morning.

3.4 What invigilators receive

Every person assigned to a hall gets a `_p_` file in the project directory — their own duty list, one line per exam: the exam, the number of students, the hall, the date and hour, and whether the course is a major. That file is the attachment they receive if you send the schedule by e-mail (Part 4).

Part 4 — Sending the schedule out

The exam plan is only useful once each person has their own copy. Wise Timetable sends every student and every lecturer **their own file**, not the whole timetable.

4.1 What is sent to whom

Recipient	Attachment	Contents
Student	`<prefix>_s_<name>(n).csv`	their exams: course, room, seat row, seat column, day and hour, `M` for a major subject
Lecturer	`<prefix>_p_<name>.csv`	their invigilation duties: course, number of students, hall, day and hour

Only people who **have an e-mail address in the database** appear in the sending list. A student without one is not listed and receives nothing — so if fewer names appear than expected, the addresses are missing, not the schedules.

4.2 Mail settings

Sending uses the same mail configuration as the rest of the application, under **Settings ▸ Advanced**:

Setting	Meaning
`EmailSMTPServer`	the outgoing mail server, e.g. `smtp.gmail.com`
`EmailSMTPPort`	the port; **587** by default
`EmailUsername`	the account that authenticates
`EmailPassword`	its password — for Google and Microsoft accounts this must be an **app password**, not the account password
`EmailFrom`	the address the mail comes from
`EmailSenderName`	the name shown beside it
`EmailIsBodyHTML`	whether the message body is HTML
`AlwaysUseCCRecipient`	an address that receives a copy of everything — useful for an archive
`EmailTargetEmailWhenUnknown`	where a message goes when the person's own address is missing
`EmailDelayBetweenTwoEmailsMs`	a pause between messages

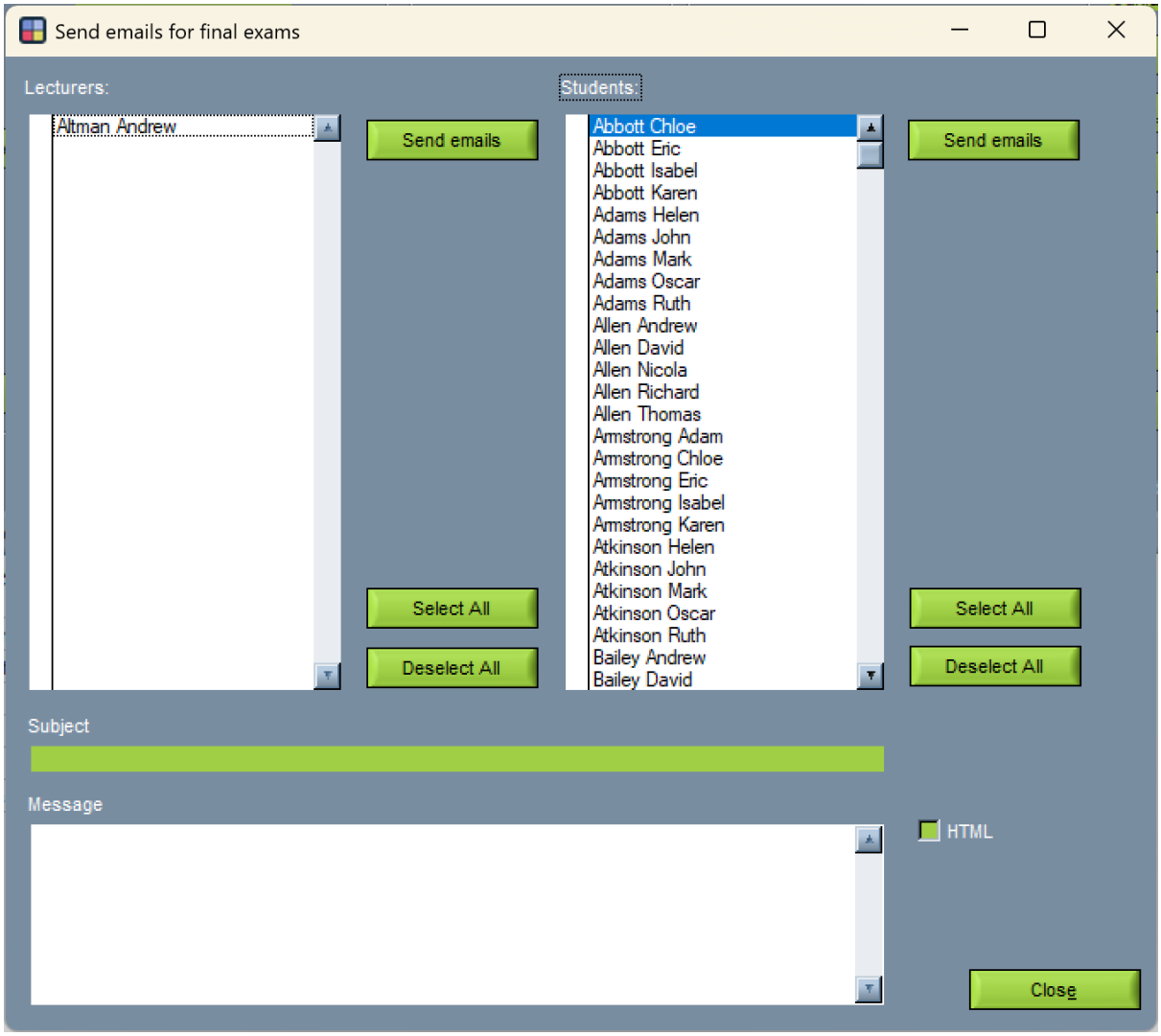
SSL is enabled automatically on every port except 25. Port 25 is treated as a plain internal relay.

That last delay setting matters more than it looks. Sending several hundred personal schedules in a burst is exactly the shape of traffic that a mail provider treats as spam; the whole run can be throttled or the account temporarily locked. If a large session is going out through a hosted mailbox, set a delay of a few hundred milliseconds and let it take its time.

Per-programme overrides exist for schools that send from different addresses per faculty — the same keys with `Override` appended.

4.3 The sending screen

Click **Send emails**. The window **Send emails for final exams** opens.



It has **two independent lists** — Lecturers on the left, Students on the right — built from the project's `_p_` and `_s_` files. Each has its own **Select All**, **Deselect All** and **Send emails** button, so lecturers and students are two separate operations: you can send the invigilation duties today and the student cards after a final check.

Write a **Subject** and a **Message** at the bottom. Both are the same for everyone — the personal part is the attachment. Tick **HTML** if the body is written as HTML rather than plain text.

> The screenshot above is worth reading as a diagnostic. The student list is full while the lecturer list holds a single name — because only one lecturer in that database has an e-mail address. **The list length tells you how many addresses you have**, not how many people are involved.

4.4 Before sending a real session

- **Send yourself one first.** Put your own address on a test student, generate, and send only that one. It costs a minute and shows you exactly what arrives.
- **Check the sender address** is one the recipients will recognise and can reply to. A schedule from a no-reply address generates telephone calls.
- **Do not send from the demo database.** The e-mail addresses in a sample or a copied database may be real people's.
- **Send after the plan is final.** A second send is a second schedule in someone's inbox, and people act on the first one they read.

Part 5 — Reference

5.1 Files a project writes

Everything lands in the project directory, prefixed with the project name.

Name	Contents
`<prefix>_settings.txt`	the session: dates, hours, limits, options
`<prefix>-c_list.csv`	which courses sit an exam
`<prefix>_c_schedules.csv`	the whole schedule, one line per exam
`<prefix>_c_schedules_ids.csv`	the same keyed by id, for integration
`<prefix>_r_<hall>_<date(hour)>.csv`	one seating chart per hall per period
`<prefix>_s_<student>(n).csv`	one card per student
`<prefix>_p_<lecturer>.csv`	one duty list per lecturer
`<prefix>_students_integration.csv`	seat-level export for a student information system

A student file is written for **every** student in the session, so a full university session leaves several hundred small files in one directory. That is normal, and it is why each session gets its own prefix.

5.2 Limits

	Maximum
Exam dates	12
Starting hours per date	5
Periods in a session	60
Exam halls	32
Exams in a session	1024
Students registered on one exam	768
Students in a session	16384
Invigilators per hall	64

Reaching one of these is written to the application log naming the limit that was hit, so a session that quietly lost the tail of a list can be identified afterwards rather than being discovered by a student who has no seat.

5.3 A first session, end to end

For a school setting this up for the first time, in order:

1. Mark the exam halls and enter their seat grids. **Rooms**.
2. Check that the largest exam you expect is no more than **half** of your largest grid.
3. Import or enter student course registrations. **Part 1.6**.
4. Set the exam supervisor on each course, and tick **Course is major** on the principal subjects.
5. Open **Tools | Plan final exam**, name the project, add the dates and hours.

6. **Select Courses** — turn off anything that does not sit an exam this session.
7. Set the three student limits. Start with 2 exams, 1 major, 2 successive days; relax them only if the session will not fit.
8. **Generate exams schedule**, and read the summary it gives you.
9. Assign **Invigilators** per hall and period.
10. Send yourself a test message, then **Send emails**.

Steps 1 to 4 are done once and then maintained. Steps 5 to 10 are the session, and take an afternoon the first time and an hour after that.