Signing in with a username and password

Wise Timetable — setting up desktop sign-in, from the beginning

◆ ◆ ◆

Wise Technologies d.o.o. · Wise Timetable desktop application · August 2026

Applies to the 2026 release and later.

Contents

What this document covers

Wise Timetable can be set so that it asks for a username and a password before anyone can use it. It is switched off in a new installation, and most institutions never switch it on, which is why the procedure is easy to forget: it is done once, years ago, by whoever set the department up.

This is that procedure, written from the source code: the one setting that switches the sign-in on, where the accounts actually come from, how to create them one at a time or for everybody at once, what the program does with a password, what the person at the keyboard sees, and what to check when somebody cannot get in.

Three different passwords, and only one of them is this one

Wise Timetable has three password mechanisms that have nothing to do with each other. Confusing them wastes an afternoon, so they are worth separating before anything else.

- **Signing in to the application.** A gate at startup: no username and password, no program. This document.
- **Editing permissions by academic year.** A much older mechanism, reached from **Locking ▸ Edit permissions**, which asks for a username and password of its own and then says which academic years that person may edit. It reads a file called `permissions.xml` in the program folder and knows nothing about the sign-in described here.
- **The database lock password.** Typed when you take a shared database's lock away from a colleague who is holding it. Not an account; one shared password.

A fourth thing that is also not this: the **database connection** has its own username and password. That is the credential the program uses to reach MariaDB or SQL Server, it is configured with the connection, and it is unrelated to who is sitting at the keyboard.

How the sign-in works, in one page

- It is **off by default** and switched on by one line in one file, **on each workstation**.
- There is **no separate list of users**. The accounts are the **lecturer records in the database that is open**: the username is the e-mail address on that record, the password is the password on that record, and the record must carry the **Administrator** role flag.
- The prompt appears at startup, after the program has loaded its data and before anyone can touch it. The window has **no close box**, and **Cancel exits the program** — it is a gate, not a suggestion.
- On success, the address that was typed becomes your **User** name for the rest of the session. That is the name a colleague sees on the database lock and on course ownership in multi-user planning, and the field that holds it becomes read-only, so nobody can put somebody else's name on their own edits.

Everything below is the detail behind those four points.

Switching it on

The setting lives in **`wtt.scheme`**, in the per-user settings folder:

```
%APPDATA%\WiseTimetable\wtt.scheme
```

On a Wise Timetable Basic installation the folder is `WiseTimetableBasic` instead. On an installation that has not yet run a current build the settings may still be in `C:\ProgramData\WiseTimetable\`; a current build migrates that folder into `%APPDATA%` on its first run, so edit the `%APPDATA%` copy.

Add — or find — the `[Advanced]` section and put the setting in it:

```
[Advanced]
EnableDesktopClientLogin=1
```

- `1` switches the sign-in on. `0`, or the line missing altogether, is the default and means off.
- The file is read **once, at startup**. Restart Wise Timetable afterwards.
- `wtt.scheme` also holds the colour scheme and a good many other advanced settings. Add this line; leave the rest of the file alone.

It is a workstation setting, not an institution setting

This is the part that surprises people. The setting is in a file in one Windows user's profile on one machine. It is not stored in the database and it does not travel with it. A planner whose workstation does not have the line will not be asked to sign in, on the same database, at the same institution, on the same day.

So switching the sign-in on for an institution means putting the line on every workstation that is meant to be gated — by editing each `wtt.scheme`, by shipping a prepared one, or by having the installer place it. If the point of the exercise is that nobody reaches the timetable without identifying themselves, a single machine without the line defeats it.

Preparing the accounts

An account is not created on a screen of its own. It is three fields on an ordinary lecturer record, and all three have to be right:

Field	Where	What it has to be
E-mail	Lecturer ▸ E-mail	The username. Must not be empty.
Password	Lecturer ▸ Password	Must not be empty.
Role: Administrator	Lecturer ▸ Role	Must be ticked.

Open a lecturer for editing in the usual way and all three are on the same screen.

The e-mail address is the username

The comparison ignores case, so `A.Novak@uni.si` and `a.novak@uni.si` are the same account. A record with no e-mail address can never sign in, because there is nothing to type.

Avoid two lecturer records carrying the same address. The program checks the records in order and stops at the first one whose address matches, so which of the two decides the password is not something anybody should have to reason about.

The Administrator role is what grants entry

The **Role** on a lecturer record is three independent tick boxes — **Professor**, **Administrator** and **Other**. The sign-in accepts a record only if **Administrator** is ticked. Professor alone is not enough, and neither is Other.

That box does double duty, and it is worth knowing before ticking it: the same role also governs booking approval, where Professor and Administrator may approve reservations for their own section and Other may not. Ticking Administrator so that somebody can sign in also gives them that. In practice the people who are given the desktop application are the people who run the schedule, so the two usually want the same answer — but decide it rather than discover it.

The password field

Type the password into the **Password** field on the lecturer screen. **Update** saves it and **Clear** removes it. The field accepts up to 31 characters.

An empty password means the person cannot sign in to the desktop application at all. (On the web application an empty password means something different — there it lets the lecturer in with just the e-mail address. The desktop gate has no such shortcut.)

Giving everybody a password at once

Setting passwords one at a time is fine for three planners and unreasonable for a faculty. Two keyboard shortcuts on the main workspace do it in bulk.

	What it does
Shift + F12	Generates a password for every lecturer who **does not have one**. Existing passwords are left exactly as they are.
Shift + Insert	Generates a **new** password for **every** lecturer, replacing whatever is there.

The custom operation `GENPASS`, run from **Data ▸ Custom operation**, does the same as Shift + F12 and is the way to do it as part of an import routine rather than by hand.

Shift + Insert is the destructive one. It replaces passwords that are working, including those of people who are signed in elsewhere and those a lecturer chose themselves on the web application. Use Shift + F12 unless replacing everything is genuinely what is wanted.

The file this produces

Either shortcut writes a file:

```
%APPDATA%\WiseTimetable\TutorPasswords.csv
```

One line per lecturer, semicolons between the fields:

```
code;first name;surname;e-mail;password
```

Every lecturer is listed, not only the ones that changed — so the file is a complete distribution list, which is what makes it useful and also what makes it dangerous. A generated password is built from the surname and the initials of the given names with four random characters on the end, so it is short enough to read out and unguessable enough to be worth having.

Two practical points:

- **The passwords in that file are in clear text.** It is written into a profile folder on the machine that generated it. Distribute the lines to the people they belong to and then delete the file. Do not leave it on a shared drive, and do not mail it round as an attachment to everybody.
- **Close it in Excel first.** The file is opened for writing each time, and if Excel is holding it the write fails and the program carries on without it — the passwords are still set, but the list you needed does not appear.

Generating passwords changes the data, so save afterwards in the ordinary way.

How a password is stored

By default Wise Timetable stores the **MD5 digest** of the password — 32 hexadecimal characters — and never the password itself. When somebody signs in, the program hashes what was typed and compares the two digests.

The lecturer edit screen keeps that consistent on its own: when it loads a password that is **not** already 32 characters long, it hashes it. So a password that arrived in clear from a CSV import or an outside feed turns into a digest the first time the record is opened and saved through that screen.

This produces the one behaviour that catches people out after a bulk generation: **the password in `TutorPasswords.csv` is the clear text**, and the stored value becomes a digest once the record goes through the edit screen. If somebody cannot sign in with the password from the CSV, that mismatch is the first thing to check, not the last.

`ForcePlainPasswords`, and when not to use it

```
[Advanced]
ForcePlainPasswords=1
```

With this set, nothing is hashed: the stored value **is** the password, and the sign-in compares the two as plain text. It exists for an installation whose lecturer passwords are maintained by an outside system that stores them in clear, so that hashing on our side would guarantee a mismatch.

It is off by default and should stay off unless that is the situation. Note also that the setting was unusable until it was repaired: on older builds, switching it on meant nobody could sign in at all, because the value being compared was never filled in. If an installation has this line and a history of “the login has never worked”, that is very likely the explanation.

What the person at the keyboard sees

The prompt is a small window with two fields, **User (email)** and **Password**. The password is masked with asterisks and accepts up to 64 characters.

- The address from the **last successful sign-in on that machine** is filled in and the cursor starts in the Password field. It is remembered in `%APPDATA%\WiseTimetable\tmpUserName.txt`, a single line, per Windows profile. Delete that file if a shared machine should stop offering the last person's address.
- **There is no close box.** Dismissing the window with the window manager's X would drop somebody straight into the application without ever having identified themselves, so the box is not drawn.
- **Cancel exits the program**, after asking for confirmation. That is the only way past the prompt without a valid password.
- A wrong username or password produces an error message and nothing else changes; the prompt stays and can be tried again. There is no lock-out and no attempt counter.
- Every failed attempt is written to the Wise Timetable event log with the username that was tried. **No password is ever written to the log** — what is recorded is the fact and the name, which is what support needs to answer "it says my password is wrong".

What changes once somebody has signed in

The address that was typed becomes the session's **User** name.

- **Settings ▸ Miscellaneous ▸ General ▸ User** shows it and is **greyed out**. Without the sign-in that field is ordinary free text, and a new installation fills it with *Wise user*.
- It is the name a colleague sees in *Warning! Database is LOCKED by …*, in *User … claimed the course …*, and against course ownership in multi-user planning.

That last point is the practical argument for switching the sign-in on at all. Multi-user planning identifies who owns a course and who holds the lock by that name, and where the field is free text the name is only as reliable as the person who typed it. The sign-in makes it something the person cannot choose.

Multi-user planning itself is described in its own manual, *Working on the same timetable at the same time*, published at **wise-tt.com**.

When somebody cannot get in

What you see	What to check
The prompt never appears	The line is missing from **that machine's** `wtt.scheme`, or was added to a different Windows profile, or to a copy of the file in the install folder rather than in `%APPDATA%`. Or the program was not restarted.
The prompt appears, a correct password is rejected	**Administrator** is not ticked on the lecturer record. Or the e-mail on the record is empty or is not the address being typed. Or the stored password is clear text while `ForcePlainPasswords` is off — or a digest while it is on.
It works for one person and not another	Compare the two lecturer records field by field: e-mail, password, Administrator. It is almost always the role box.
It worked yesterday	Somebody ran **Shift + Insert** and replaced every password, or the lecturer record was edited and saved. Check `TutorPasswords.csv` for its timestamp.
Nobody at all can get in	The accounts live in the database, so anyone still signed in — on any machine — can correct a record. If nobody is, contact support: we can get an authorised person back in without anybody editing the database by hand.

One more thing to hold in mind while diagnosing: the accounts checked are those of the **data that is loaded**. The prompt appears after the program has opened its database or file, so an installation that switched database will be checking the new one's lecturer records, not the old one's.

What this mechanism is, and what it is not

It is worth being plain about this, because a sign-in prompt invites people to assume more than it delivers.

It is a gate on the application. It establishes who is at the keyboard and keeps casual hands out of the schedule. That is a real and useful thing: in a department where three people edit one database, knowing whose name is on the lock and on a claimed course is worth having.

It is not a permission system. Everyone who gets past the prompt has the same rights inside the program. The only rights mechanism in the desktop application is the year permissions described at the top of this document, and it is separate.

It is not protection for confidential data. MD5 is a digest, not a defence: it is unsalted, it is cheap to attack with commodity hardware, and the digests sit in the same database as the timetable they are protecting. Anyone who can read the database can work on them at their leisure. Where the data genuinely matters, the controls that count are the ones on the database and on the machines that reach it — not this.

Read the sign-in as *identification*, not as *security*, and it is doing exactly the job it is good at.

Related documents

- **Working on the same timetable at the same time** — multi-user planning, and what the User name does once several planners share a database.
- **Wise Timetable — Importing Companion** — the complete import reference, including the lecturer imports that can carry an e-mail address and a password.

Both are published free at **wise-tt.com**.