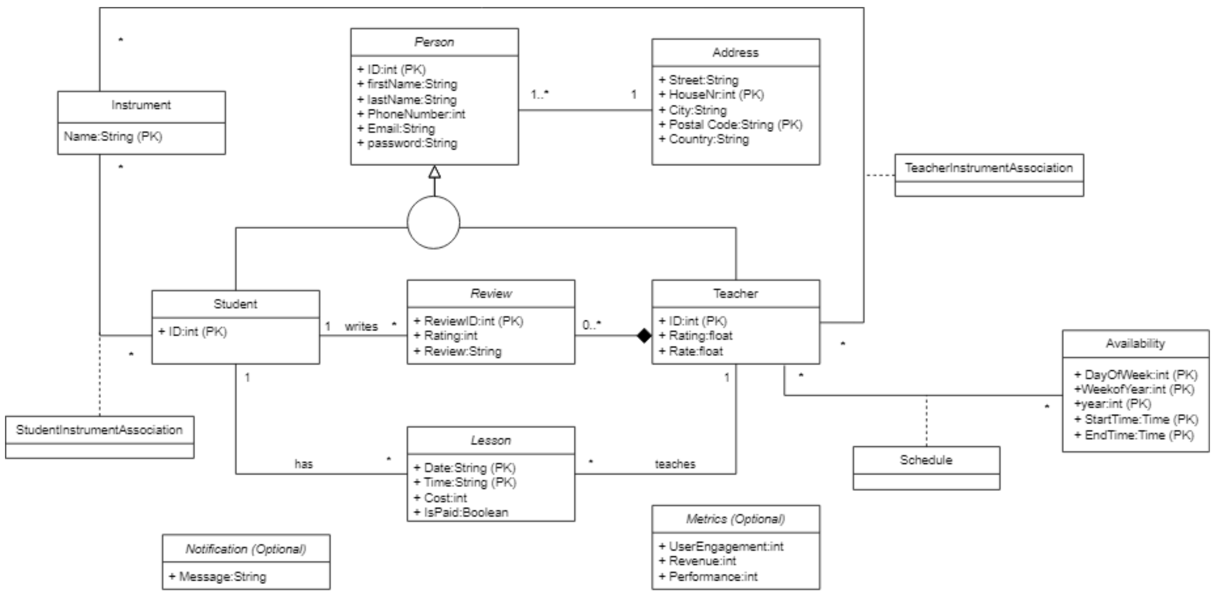
Explanation class diagram



Our class diagram has quite a couple of different classes, we did this for as much modularization as possible. Now let's explain our classes and their multiplicities, their attributes will be explained further when talking about our database schema. Let's start with person, a person can have only one address in our database, furthermore an address can have anywhere from 1 to infinite persons. Person is also the superclass for student & teacher, this relation is neither covering nor disjoint, because a person can be both a student and a teacher and there are also persons which are neither student nor teacher E.g. Admin. Teachers & students also have an association class that links them to instruments, students and teachers can have 0 to many instruments and vice versa so that is why we needed association classes for our database.
Students can also write 0 to many reviews and teachers can have 0 to many reviews. We didn't use an association class here so a student could write multiple reviews for a teacher. The same is true for lessons.
Lastly we have the schedule for the teacher. Each teacher can have as many availabilities as they want, this is why we have the association class schedule, which has the availability per day per teacher.
We also have some optional classes which are called notifications and metrics, we haven't connected these to the database, since we first want to make our structure around all of the most important classes.

Explanation database schema

address:
(House_nr, Postal_Code, Street, City,
PK(House_nr, Postal_Code));

Address has a house number, postal_code, street name and city name. Each address can be uniquely identified by their zip code and house number, so that is why that is the primary key.

person:
(PID, FirstName, LastName, PhoneNumber, Email, Password, Postal_Code NOT NULL, House_nr NOT NULL, PK(PID),
FK(Postal_Code) REF Address(Postal_Code)
FK(House_nr) REF Address(House_nr));

Person has some personal information together with a unique id and the primary key of the address, since a person can only have one address.

student: (SID, PID NOT NULL, PK(SID), FK(PID) REF Person(PID));

Student inherits all the attributes for person together with a unique student id.

teacher: (TID, Rating, Rate, PID NOT NULL, PK(TID), FK(PID) REF Person(PID));

Teacher inherits all the attributes for person together with a unique teacher id, but it also has a rating from reviews and the amount of money they charge per hour.

Availability: (DayOfWeek, WeekOfYear, Year, StartTime, EndTime, PK(DayOfWeek, WeekOfYear, Year, StartTime, EndTime));

Availability is uniquely specified for each day, the best way we found was using an integer for the day of the week and the week of the year. Start time and end time are also part of the primary key, this makes it possible to have multiple time slots available per day.

Schedule: (Teacher NOT NULL, DayOfWeek NOT NULL, WeekOfYear NOT NULL, Year NOT NULL, StartTime NOT NULL, EndTime NOT NULL, PK(Teacher, DayOfWeek, WeekOfYear, Year, StartTime, EndTime),
FK(Teacher) REF Teacher(tid),
FK(DayOfWeek) REF Availability(DayOfWeek),
FK(WeekOfYear) REF Availability(WeekOfYear),
FK(Year) REF Availability(Year),
FK(StartTime) REF Availability(StartTime),
FK(EndTime) REF Availability(EndTime));

Schedule is a pretty big class that keeps track of the availability for each teacher for each timeslot. Some old data should be deleted here once in a while to keep it from getting too big.

instrument: (name, PK(name));

Instrument just has a name, which is unique.

studentInstrumentAssociation: (Student, Instrument, PK(Student, Instrument),
FK(Student) REF Student(SID),
FK(Instrument) REF Instrument(Name));

This is for keeping track of which student has which instrument.

teacherInstrumentAssociation: (Teacher, Instrument, PK(Teacher, Instrument),
FK(Teacher) REF Teacher(TID),
FK(Instrument) REF Instrument(Name));

This is for keeping track of which teacher has which instrument.

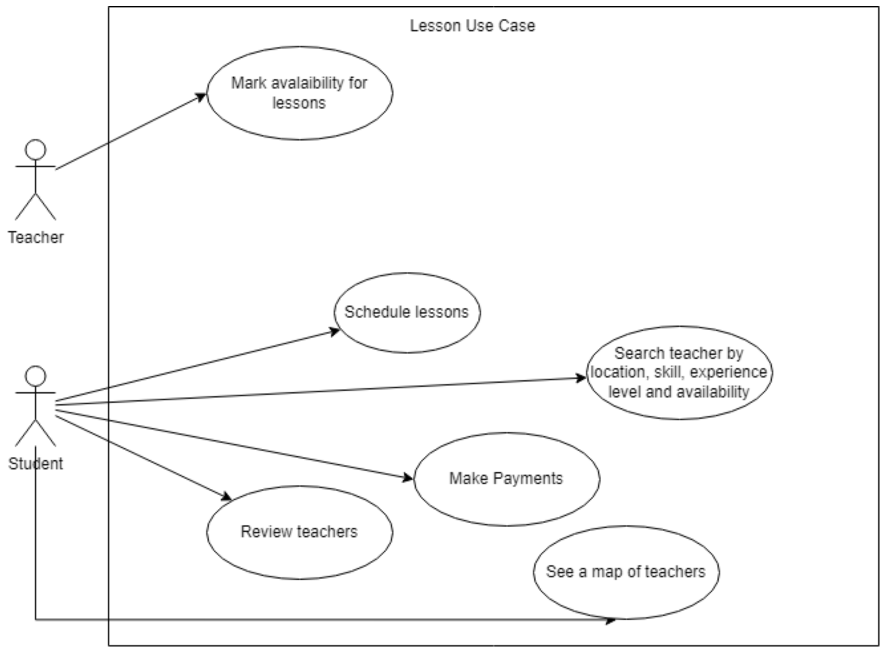
Review: (rid, score, message, Student NOT NULL, Teacher NOT NULL, PK(RID)
FK(Student) REF Student(SID),
FK (Teacher) REF Teacher(TID));

Review has a unique id to keep track of all individual reviews, they can also have a message in them. All reviews also need both a student and a teacher.

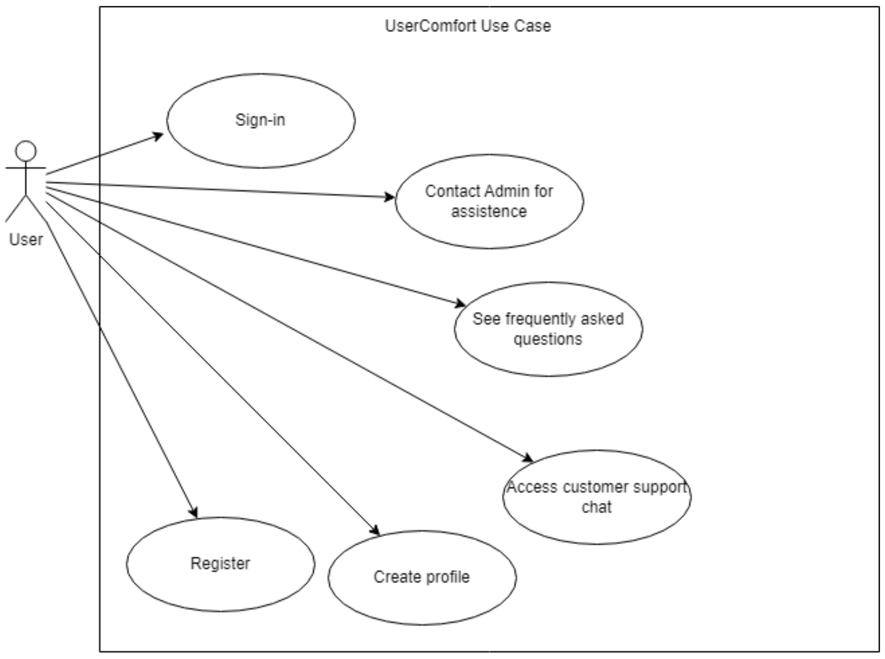
Lesson: (Date, Time, Cost, IsPaid, Student NOT NULL, Teacher NOT NULL, PK(Date, Time, Student, Teacher)
FK(Student) REF Student(SID),
FK (Teacher) REF Teacher(TID));

Lesson has some information like the date, time and cost, it also has a boolean called isPaid to keep track of the payment. Just like review, a lesson also needs a student and a teacher.

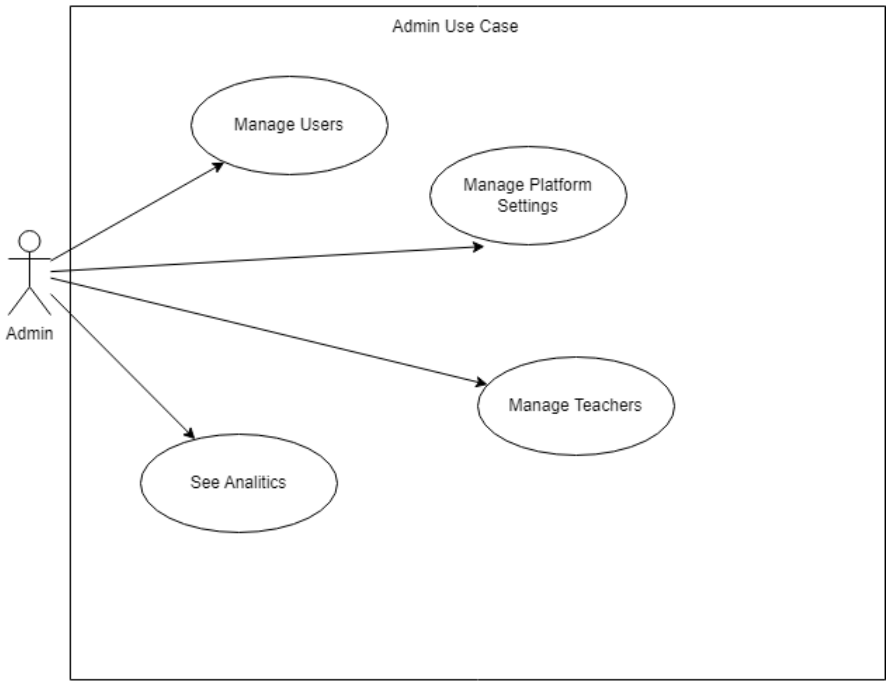
Use-case diagrams explained


Lesson Use Case
Mark avalaibility for lessons
Teacher
Schedule lessons
Search teacher by location, skill, experience level and availability
Student
Make Payments
Review teachers
See a map of teachers

This use-case diagram shows you all the actions students and teachers can take while making lessons. students can review teachers after a lesson, schedule lessons inside the availability of the teacher, make payments for lessons, see a map with all the teachers and search for teachers directly with some filters.
Teachers can mark their availability and they can also cancel lessons that have been scheduled by students.


UserComfort Use Case
User
Sign-in
Contact Admin for assistence
See frequently asked questions
Access customer support chat
Register
Create profile

This use-case diagram looks at the profile and user comfort use cases. As a user, you can register a new account, you can sign in if you already have an account, you can contact an admin through a live chat for assistance, you can look at a page for frequently asked questions and you can also view your profile when logged in and alter it by making a description or adding a video showing their skills.



Our last use-case diagram shows us what the admin can do. They can manage users, by changing some information or blocking their account. They can change other settings on the application, by disabling certain features, they can also manage teachers the same way as students and they have access to an analytics screen. These use cases are more optional than the other ones, so this will only be implemented when everything else is finished.