Design Report Shotmaniacs Team 1

A picture containing sketch, drawing, origami, design

Description automatically generatedThis report is about the class diagram, use case diagram and database schema of the current implementation that will be presented in the presentation on May 31, 2023, and some of the changes that are expected to be implemented in the next sprint(s).

Table of Contents

[Class Diagram 3](#_Toc136798429)

[Use Case Diagram 5](#_Toc136798430)

[Database Schema 6](#_Toc136798431)

# Class Diagram

The image underneath is the current implementation of the class diagram. Colors are used to group up certain classes.

* Actors are green.
* Enum classes are yellow/orange.
* Any event related classes are red.
* Blue and purple are fillers; they have no meaning.

A picture containing text, diagram, screenshot, parallel

Description automatically generatedThe table Account used to be called User. When creating the DB, SQL did not allow ‘User’ as a valid table name, as it is a reserved word. As a result, to be certain that there is no confusion between the schema and class diagram, the User class is now renamed to Account to be consistent with the schema.

A screen shot of a computer

Description automatically generated with low confidenceThis class contains the maximum size of the crew **per role**. This has an effect on the circled multiplicities: Whatever the multiplicity is on the bottom 1..7, it must be the same as the top 1..7. So if an event needs photographers and assistants, it has multiplicity 2 coming from ‘event’ line and 2 that goes to the role enum, as there is no need to add seven roles for each requirement list if five of them are of maximum size 0.

There is one change (that is known at least) yet to come, which influences the link between login session and account. It would be nice to let an account have more than 1 login session active. For instance, one on the phone and one on the desktop. The class diagram does not allow for that right now. The multiplicity changes to 0..1 to 1 then.

# Use Case Diagram

A picture containing text, diagram, line, circle

Description automatically generated

The use case diagram is very minimalistic yet shows that what is possible by all the actors. Something that is not implemented yet, is registering an account, clients can register their own account, while administrators should have the possibility to create accounts for other newly reqruited admins and crew members. This is something for the later sprints.

On the right side, it is visible that the crew members and administrators are CD-generalized in staff. This is done to avoid too many added connecting lines in the box.

# Database Schema

This is the schema of the database. In the first box, all enum types are defined. Where these enums are used, is mentioned for each variable as a type. As well as some timestamps are used which are mentioned for clarity.

CREATE TYPE team AS ENUM ('core', 'club', 'core and club');

CREATE TYPE role AS ENUM ('assistant', 'data handler', 'editor', 'photographer', 'planner', 'producer', 'videographer');

CREATE TYPE account\_type AS ENUM ('administrator', 'client', 'crew member');

CREATE TYPE status AS ENUM ('to do', 'in progress', 'review', 'done');

CREATE TYPE event\_type AS ENUM ('club photography', 'festival', 'product shoot');

account(

id PK,

forename,

surname,

username

email\_address,

password,

type NOT NULL,

UNIQUE(email\_address),

UNIQUE(username)

);

has\_login\_session(

account PK,

session\_id,

expires timestamp DEFAULT current\_timestamp + interval '1' hour,

FK(account) REF account(id)

);

administrator(

id PK,

FK(id) REF account

);

announcement(

id PK,

announcer\_id NOT NULL,

title,

body,

datetime timestamp without time zone DEFAULT CURRENT\_TIMESTAMP,

FK(announcer\_id) REF administrator(id)

);

client(

id PK,

phone\_number,

FK(id) REF account

);

crew\_member(

id,

role,

team,

PK(id, role, team),

FK(id) REF account),

);

event(

id PK,

client\_id NOT NULL,

name,

description,

start timestamp without time zone,

duration,

location,

production\_manager\_id,

status NOT NULL DEFAULT ‘to do’,

type event\_type NOT NULL,

FK(client\_id) REF client(id),

FK(production\_manager) REF crew\_member(id)

);

event\_requirement(

event\_id,

role,

crew\_size,

PK(event\_id, role),

FK(event\_id) REF event(id)

);

event\_enrollment(

event\_id,

crew\_id,

PK(event\_id, crew\_id),

FK(event\_id) REF event(id),

FK(crew\_id) REF crew\_member(id)

);

Primary key ids are of type serial; this is to make it easier with implementing a default value that increments in an automatically created sequence for each row. This method is used for all PK ids in the tables. Whenever an id is an FK referencing to an already existing id PK; then integer is used instead of serial, as incrementing is implied by table it is referencing the serial from.

For many String types, the decision was made to use varchar() type instead of text. This helps limiting the length of Strings, as the maximum size of varchar prevents absurdly long columns. To not limit any security, account password is saved in text, for the few people that may want to have a crazy long password. Another decision was in the announcement table, the title is in varchar, but the body is in text. This provides a small limitation on the titles’ length, but it would break the lay out of the page if we allowed extremely long titles and that overtakes the purpose of having a body for the announcement overall.

The table administrator is as of right now technically redundant because we can use the id from account table for referencing to administrators (e.g., for announcer\_id in announcement table), this however provides more flexibility by having the option open to add attributes in the future that are specific to administrators. So, if in the next sprints the decision is made to remove the administrator table, a necessary change is to alter the announcer\_id (where it references to) in the announcement table.   
The same principle is true for the client class. However, this should still be existing, as phone numbers should only be stored for clients and not for staff members. A possibility is to have phone number attribute in the account table, but it would create null values for all staff members in that column, which is not ideal.   
Crew members have their own table that also links the id with permission type and the role. It makes sense to have a crew member table in this case. The attributes, role and team, used to be in separate association classes linked to the crew members, but it would then be difficult to show if a crew member has club in one role and core in the other. To make this possible, both were put into the crew\_member table.

Another necessary change in the next sprints when the confirmation email will be sent, is to make sure the email address is correct. A domain constraint must be added in the account table at the email column, as well as the phone number to check their validity.