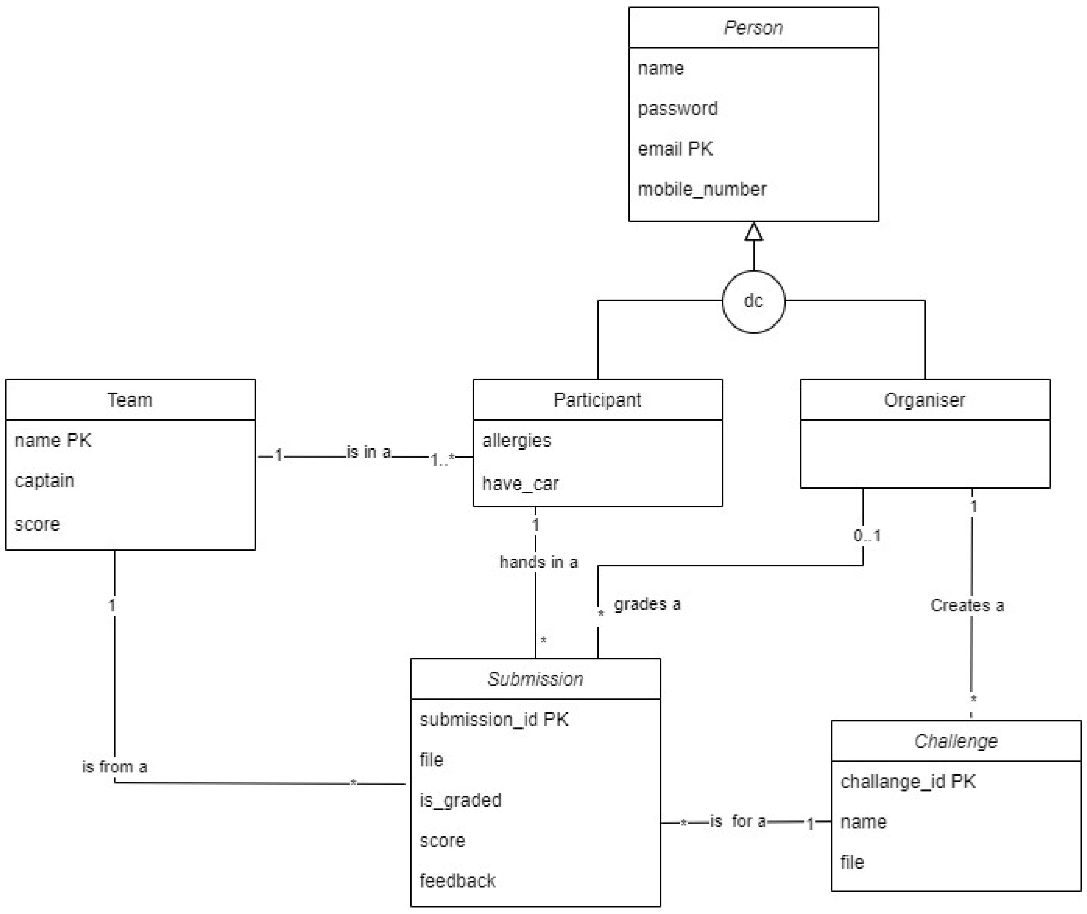
Class Diagram, SQL schema & Use case Diagram

Class Diagram:



Above is a class diagram representation of our database. First of all, Person represents all users of our system, consisting of only participants and organizers. A user can't both be an user and an organizer. Thus the relation is both covering and distinct. For a person we store a name, password, email and mobile number, with the email being a primary key as no two users are allowed to have used the same email for sign up. When a user is a participant we also store allergies and if the user has a car. A participant is always in one team. A team consists of a minimum of 1 person and a maximum of 5 people. For a team we store the name, which is the primary key, thus no two teams are allowed to have the same name, the captain and the score. The captain is the founder of the team and can delete members of their team. A participant can hand in multiple submissions, and a submission is always from one team and one participant. A submission has a submission_id, a file, and a boolean if it is graded. It can be graded or not by an organizer, if it is graded a score and possible feedback is added. The submission is for one certain challenge. There are many submissions for every challenge. A challenge has a challenge_id, a name and a file. Of Course, there are many kinds of challenges, which are differentiated by the challenge_id. A challenge id for crazy88 would for example be c00001 and for a puzzle p00001. A challenge is created by one organizer.

SQL schema:

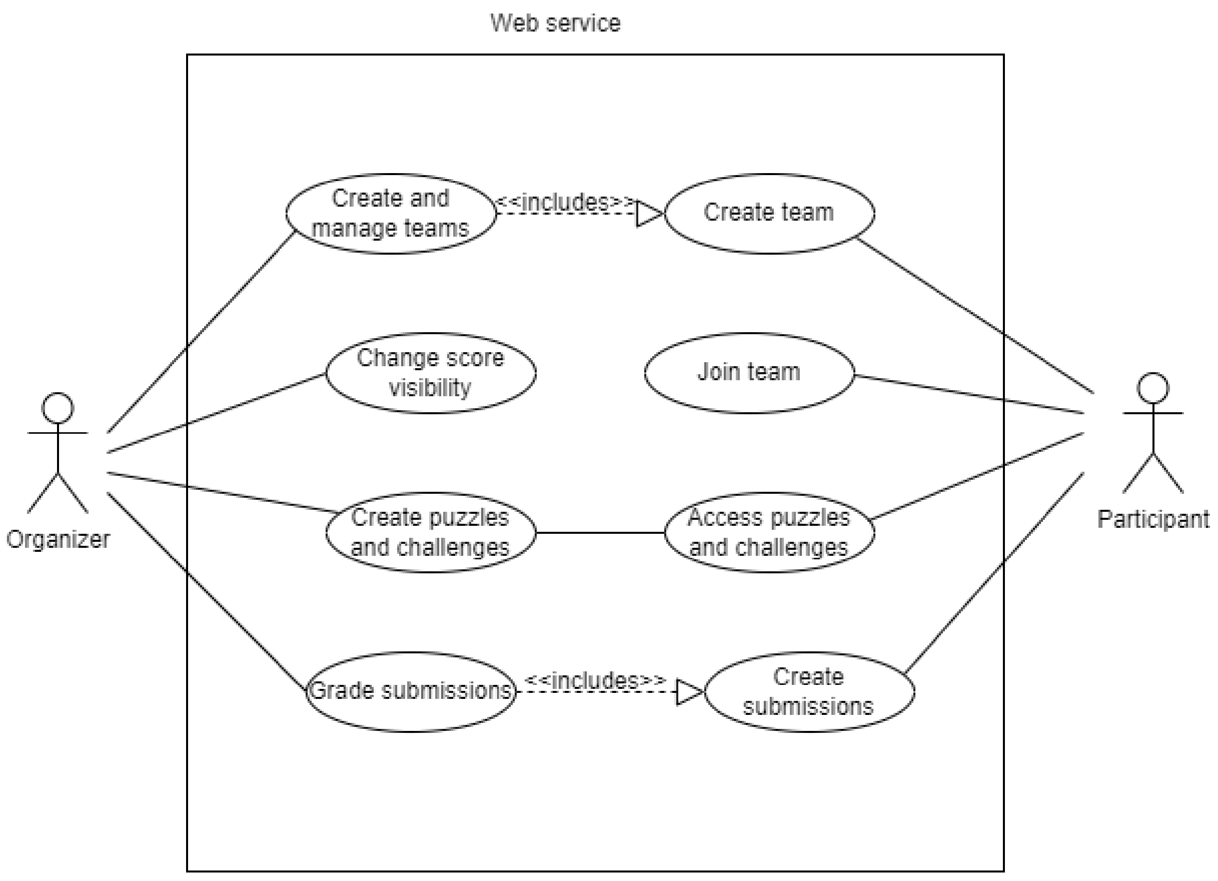
```
Person(name, password, email, mobile_number, is_participant, team, allergies, have_car)
        PK(email),
        FK(team) REF Team(name));
        CONSTRAINT if_participant_then_team_is_not_null
                CHECK ( (NOT is_participant) OR (team IS NOT NULL) ) ) ;

Team(name, captain, score,
        PK(name));
        CHECK (name IN (SELECT team FROM Person)));

Submission(submission_id, file, is_graded, score, feedback, team NOT NULL, challenge
NOT NULL, handed_in_by NOT NULL, graded_by,
        PK(submission_id),
        FK(team) REF Team(name),
        FK(challenge) REF Challenge(challenge_id),
        FK(handed_in_by) REF Person(email),
        FK(graded_by) REF Person(email));

Challenge(challenge_id, name, file, created_by NOT NULL,
        PK(challenge_id)
        PK(created_by) REF Person(email));
```

Above is our database schema. Most tables speak for themselves, so only an explanation for the checks will be provided. For the person table we have a check to ensure that if the person is a participant, the participant is in a team (thus team is not null). For the Team we check if their name is in the Person table to ensure that every team has a minimum of 1 member.

Actual Code:

```
DROP TABLE Team, Person, Challenge, Submission;

CREATE TABLE Team
        (name VARCHAR,
        captain VARCHAR,
        score int,
        PRIMARY KEY(name));

CREATE TABLE Person
        (name VARCHAR,
password VARCHAR,
email VARCHAR,
mobile_number VARCHAR,
is_participant BOOLEAN,
team VARCHAR,
allergies VARCHAR,
have_car BOOLEAN,
        PRIMARY KEY(email),
        FOREIGN KEY(team) REFERENCES Team(name),
        CONSTRAINT if_participant_then_team_is_not_null
                CHECK ((NOT is_participant) OR (team IS NOT NULL)));

CREATE TABLE Challenge
        (challenge_id int,
        name VARCHAR,
        file VARCHAR,
        created_by VARCHAR NOT NULL,
        PRIMARY KEY(challenge_id),
        FOREIGN KEY(created_by) REFERENCES Person(email));

CREATE TABLE Submission
        (submission_id int,
        file VARCHAR,
        is_graded Boolean,
        score int,
        feedback VARCHAR,
        team VARCHAR NOT NULL,
        challenge int NOT NULL,
        handed_in_by VARCHAR NOT NULL,
        graded_by VARCHAR,
        PRIMARY KEY(submission_id),
        FOREIGN KEY(team) REFERENCES Team(name),
        FOREIGN KEY(challenge) REFERENCES Challenge(challenge_id),
        FOREIGN KEY(handed_in_by) REFERENCES Person(email),
        FOREIGN KEY(graded_by) REFERENCES Person(email));
```

Use case Diagram:



We model our system with one use case diagram. We have two actors, organizer and participant. A participant can create and join teams, get access to the challenges, and create submissions. The admins can confirm and manage the teams, change the state of the scoreboard, create or modify challenges, and grade the incoming submissions from the players.