

Database Management Systems II

ECE 541

12. Application using XQuery

Authors:

Chatzimichail Antonios-Kyrillos 2657

Liagka Maria 2558

Date: 6th January 2022

Installation Guide

1. Open the folder: XqueryApplication_dist

*****DO NOT CHANGE ANYTHING inside "XqueryApplication_dist" folder*****

2. To run the project from the command line, go to "XqueryApplication_dist" folder and type the following:

```
java -jar "XqueryApplication.jar"
```

or just double-click the file with the name:

XqueryApplication.jar

Tip: *If you need to recover the initial database, please copy the ".xml" files from "xml_bp" folder to "xml" folder, as well as the contents from "photos_bp" to "photos".*

Introduction

We have created a photography game where you can sign up for free and upload your own photos. You have the chance to discover and participate in various challenges with different themes. The goal is to collect votes-points and level up. Every challenge has its leaderboard, so you can check your position at any time. The game is based on a database with users, photos and challenges.

User Guide

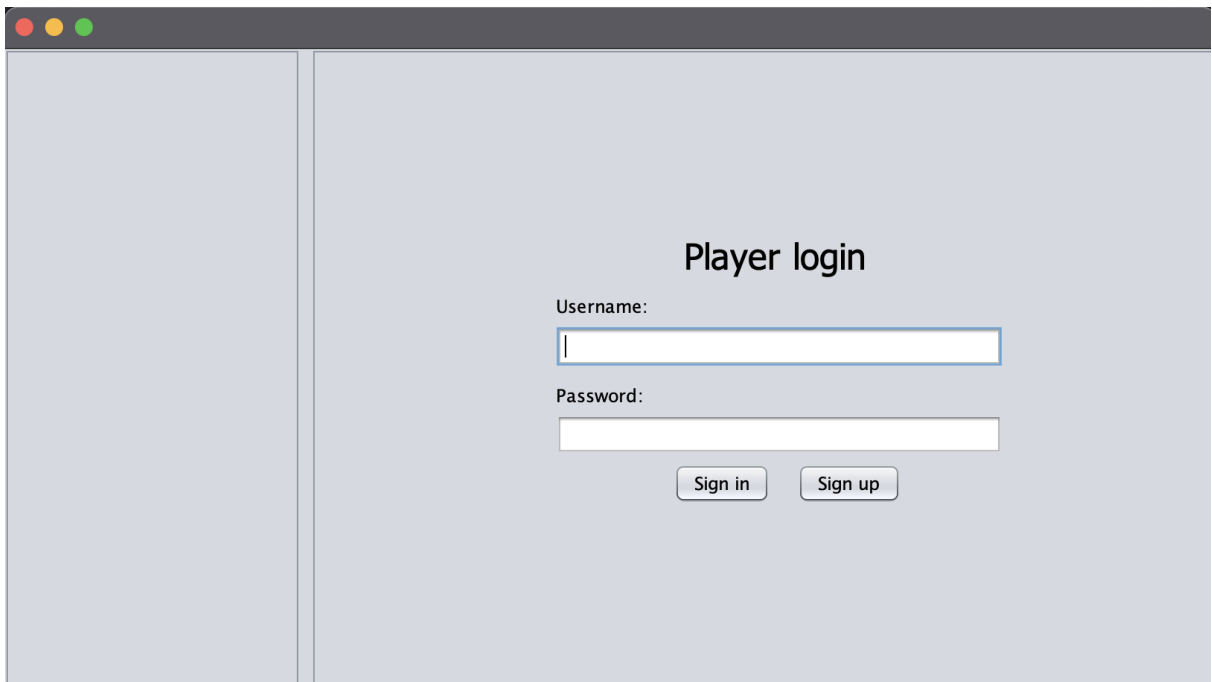
Run the application by typing on the command line:

```
java -jar "XqueryApplication.jar"
```

or just double-click the file with the name:

XqueryApplication.jar

You will see the following window:

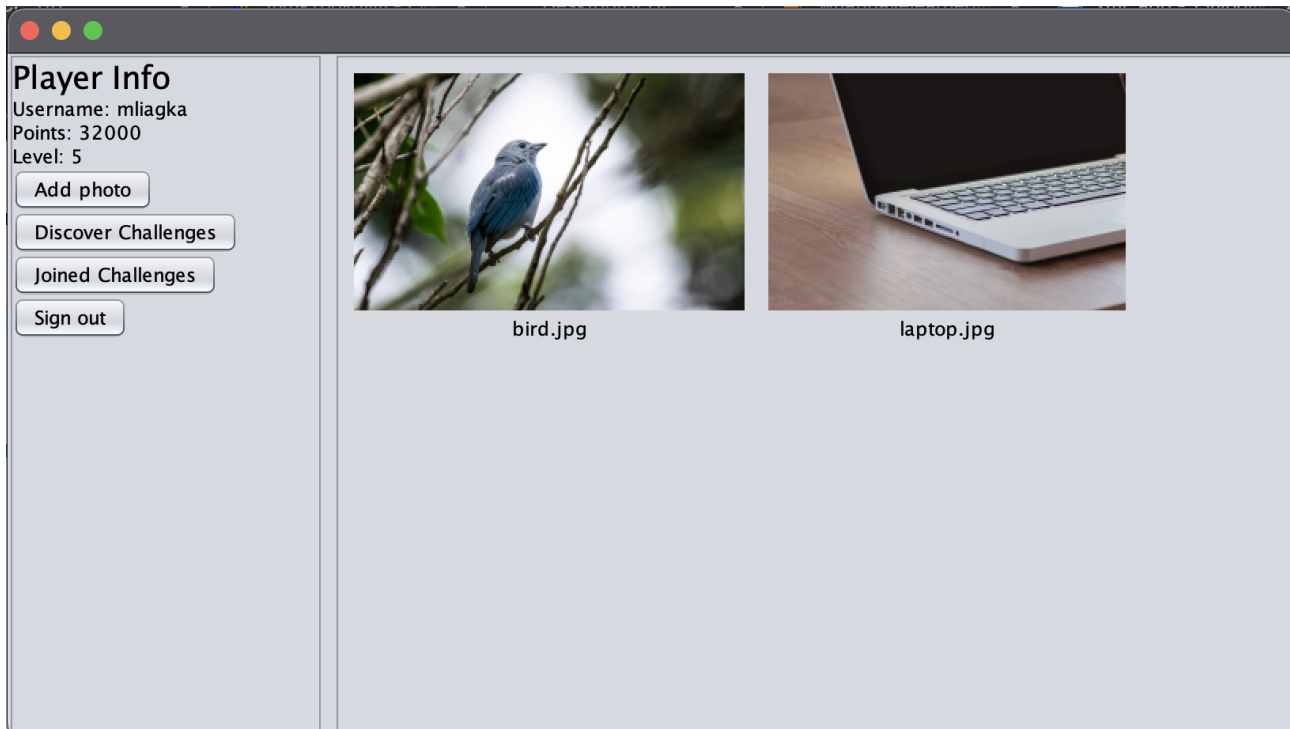
A screenshot of a Java application window titled "Player login". The window has a dark gray title bar with standard macOS window controls (red, yellow, green buttons). The main content area is light gray and features a vertical sidebar on the left. The login form is centered and includes the title "Player login", a "Username:" label with a text input field, a "Password:" label with a text input field, and two buttons labeled "Sign in" and "Sign up".

1. Home screen - Player login

If you are a first-time user, you need to sign up. Fill out the "Username" and the "Password" you want and press "Sign up". Then, sign in.

If you have been already registered, just type your "Username" and your "Password" and sign in.

After signing in, you will see the following window:

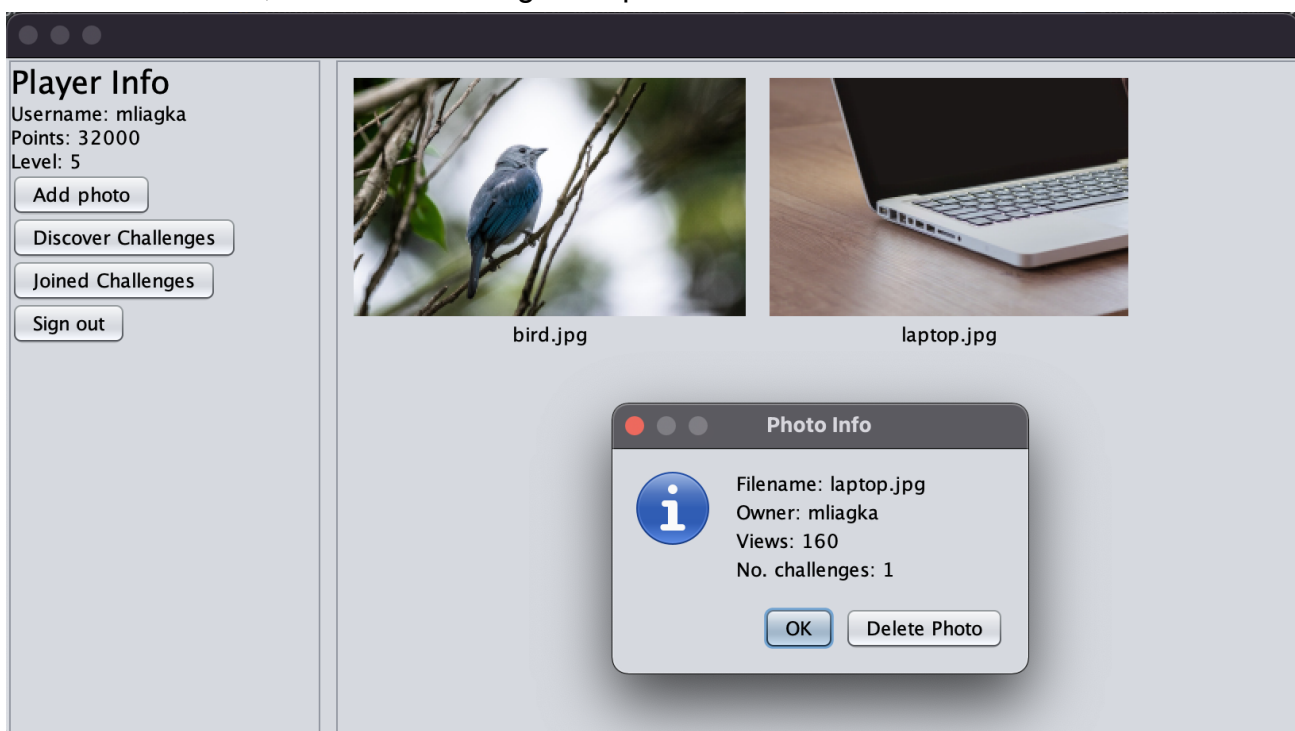


2. Player profile

This is your profile. On the upper left corner you can find your infos: Username, Points and Level. On the main screen you can see your photos you have already uploaded and their titles.

From the menu (down from Player Info) you have four (4) choices: "Add photo", "Discover Challenges", "Joined Challenges" and "Sign Out".

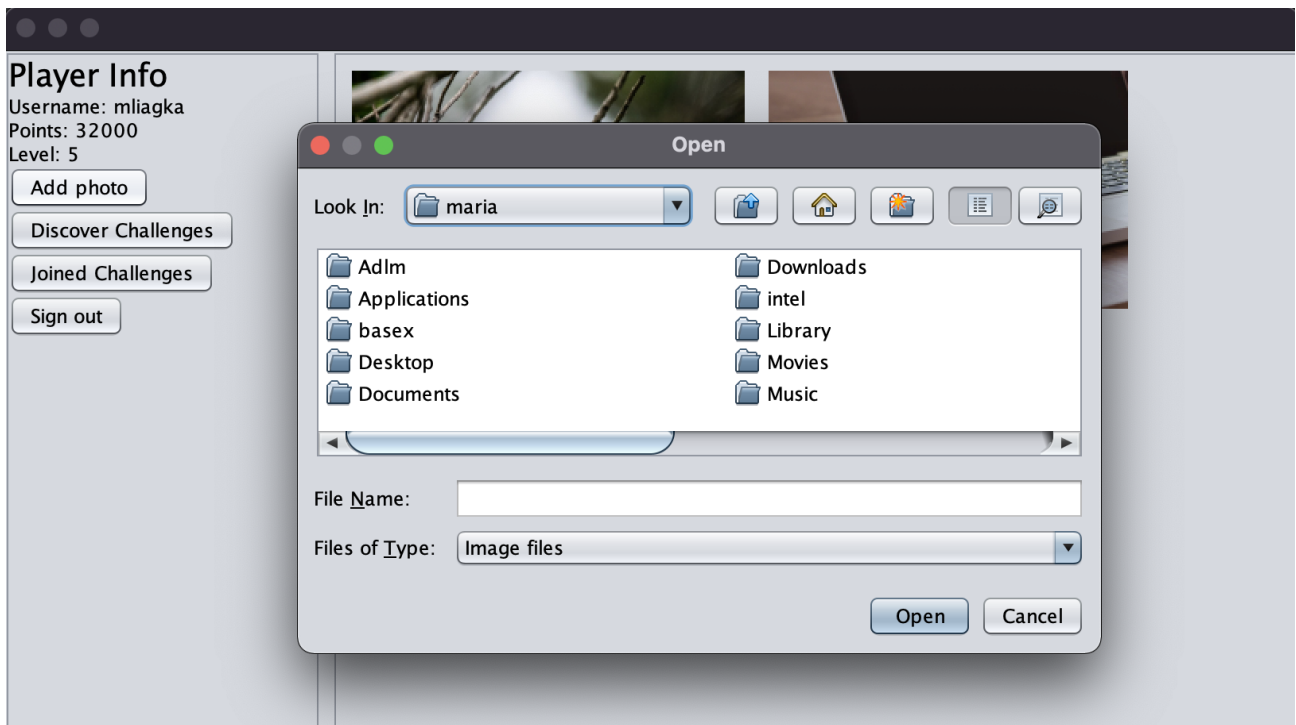
If you click on a photo you will see all the available information of the photo and you will also be able to delete it, as in the following example:



3. Photo Info or Delete Photo

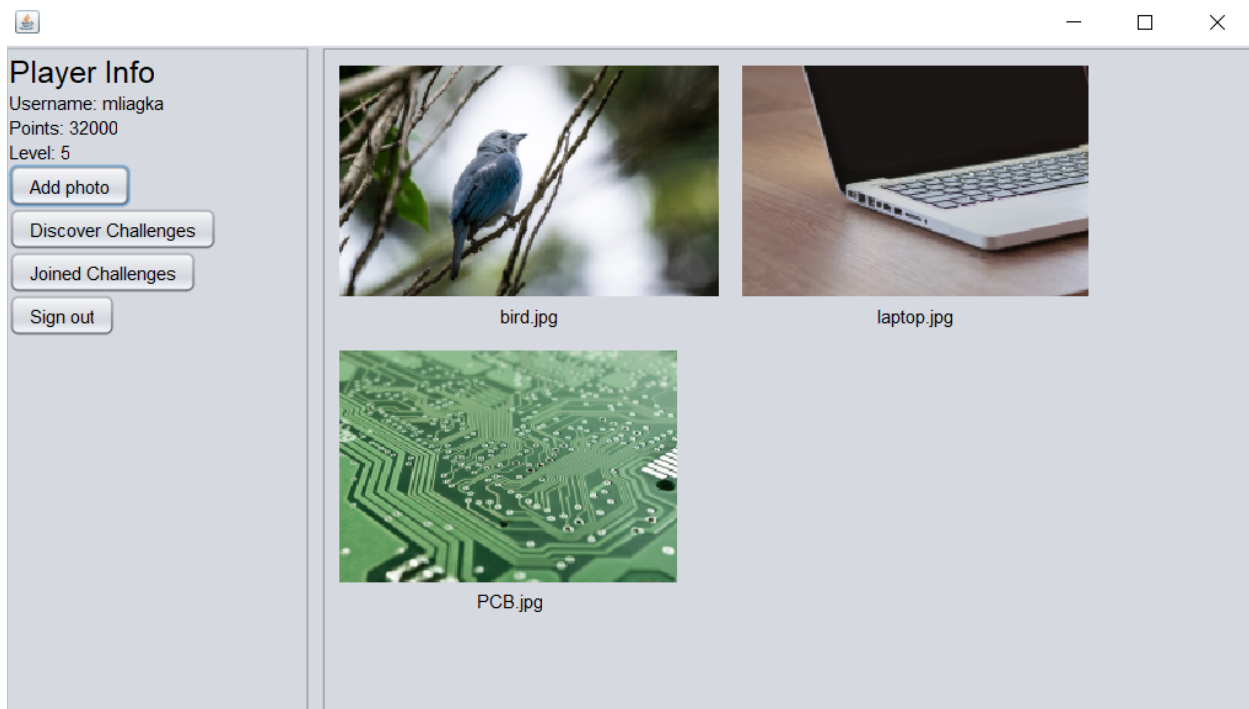
First option - **Add Photo**

By clicking “Add Photo” it will open a new window in order to search in your disk the photo you want to upload as the following:



4. Upload photo

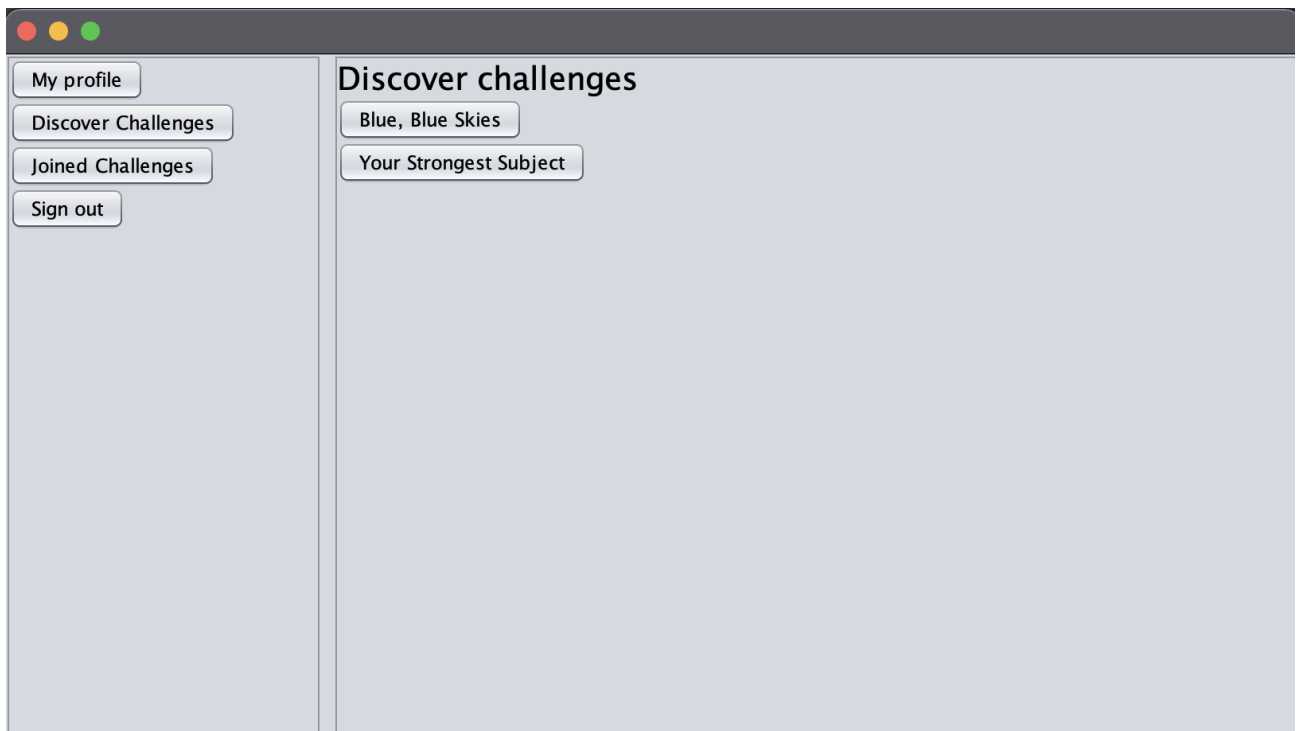
After finding your photo press “Open” in order to upload the photo. In case you regret the uploading just press “Cancel”. When the uploading has been completed, you can see your photo in your profile.



5. Profile after upload

Second option - **Discover Challenges**

By clicking “Discover Challenges”, you will be redirected to a new page where you can see all the available challenges to participate in.



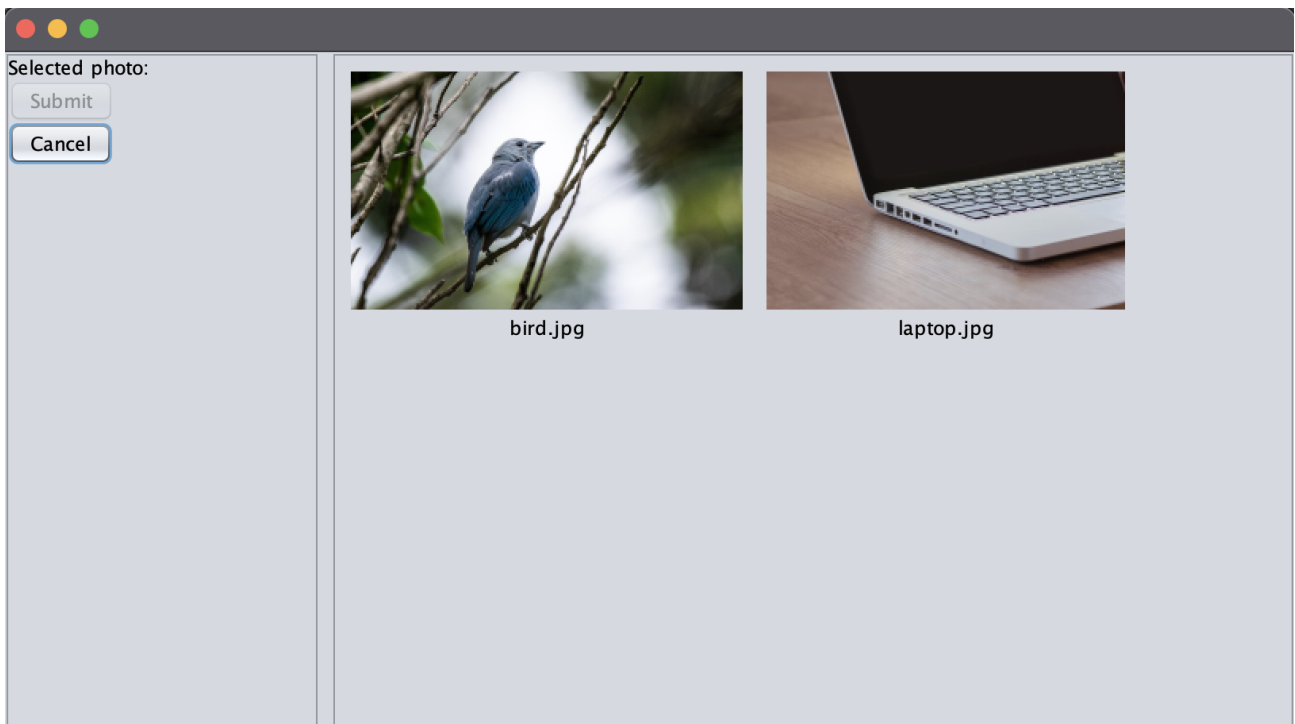
6. Discover Challenges

Choose between the available challenges the one you want to discover. You can either “Join!” or “See leaderboard”.



7. Join! or See leaderboard in a challenge

If you press “Join!”, you are directed in your collection of uploaded photos and you need to choose the one to participate in the challenge, as the following example:



8. Join a challenge with an uploaded photo of your profile

After choosing the photo press “Submit” and you are an official participant of the challenge. In case you don’t want finally to join the challenge, press “Cancel”.

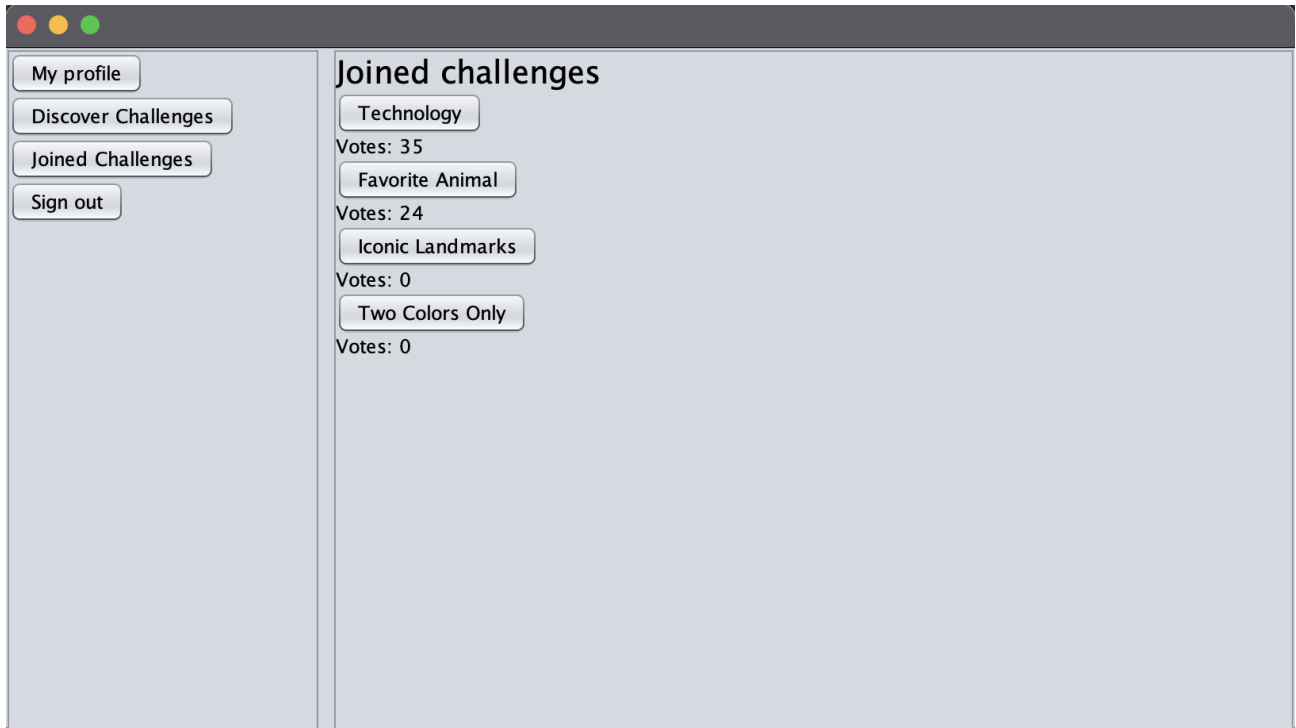
If you press “Show leaderboard”, you will see the leaderboard of the challenge:



9. Show leaderboard of a challenge

Third option - **Joined Challenges**

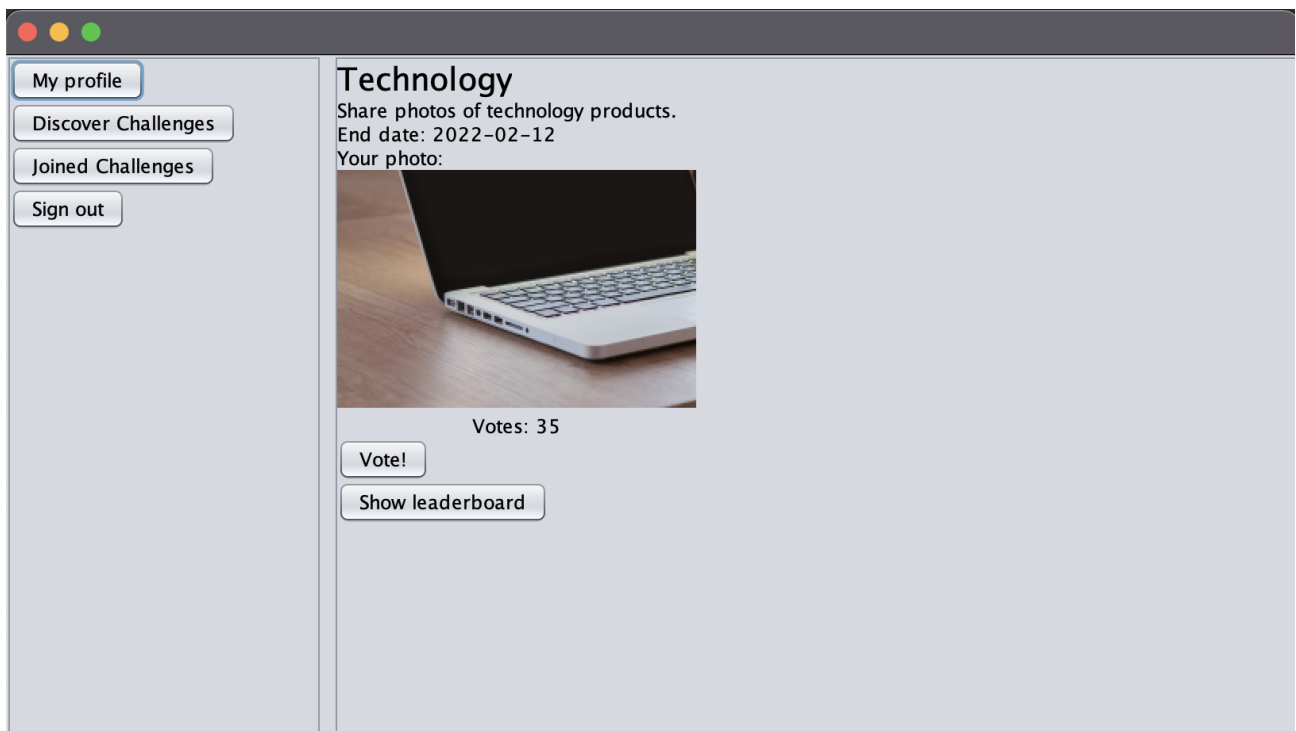
By clicking “Joined Challenges” you see the challenges you have already taken part in, the photo with which you have participated in and the votes you have collected for every photo.



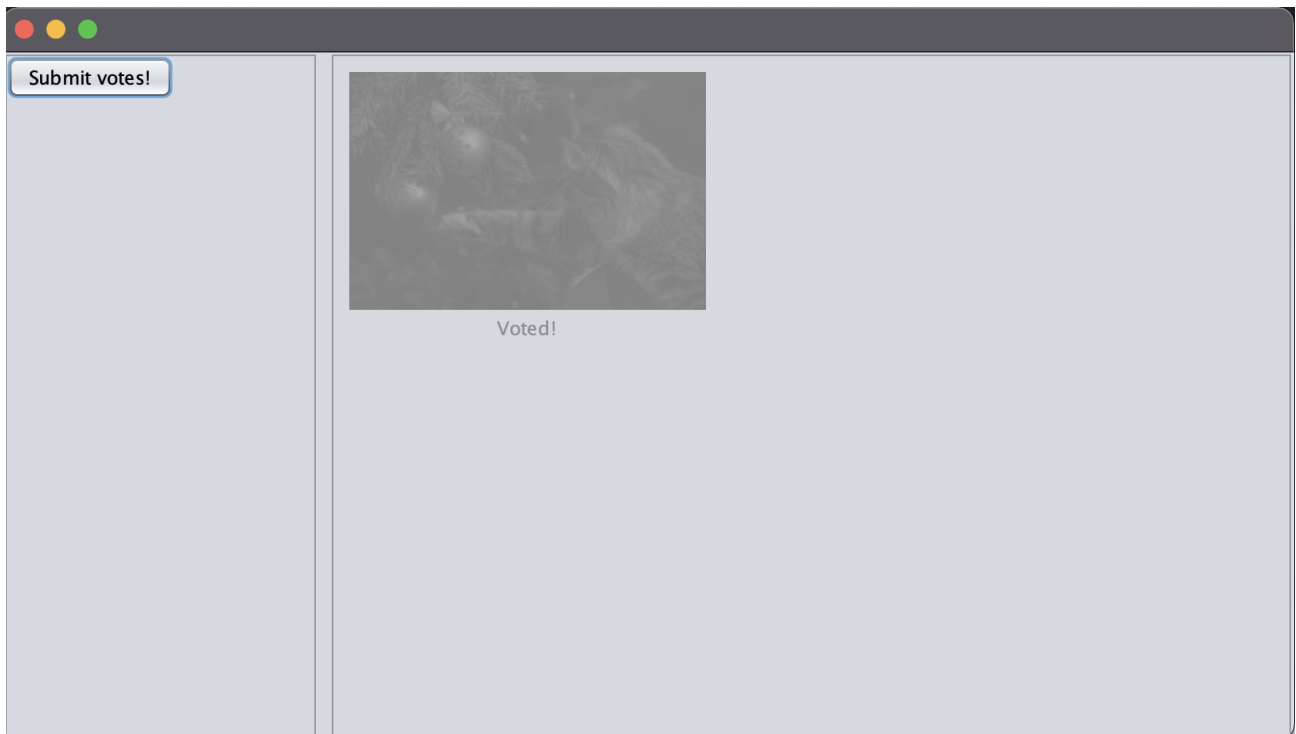
10. Joined challenges

It also provides some information on every challenge you have joined such as the title of the challenge, the description of the challenge, the end date and the leaderboard.

After joining a challenge you have the chance to “Vote!” photos you like. For voting, press “Vote!” and then “Submit Vote” as the following example:



11. Vote a photo

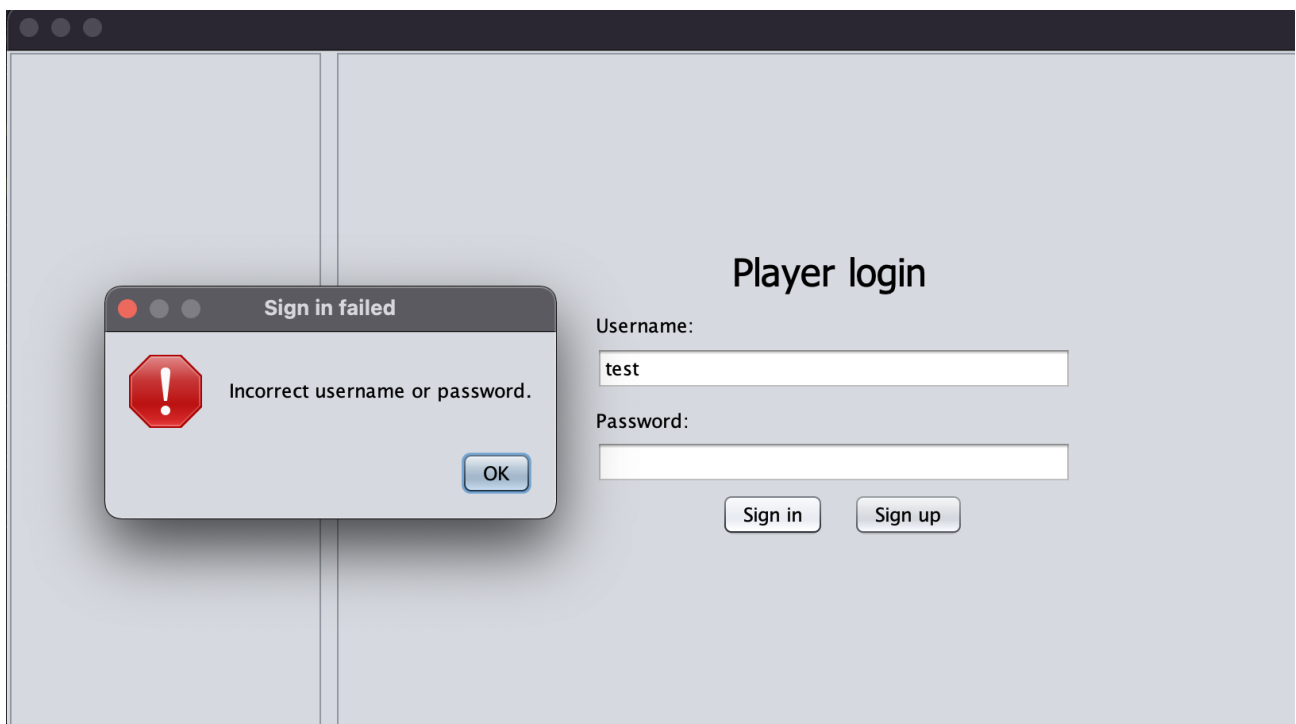


12. Submit voting

Fourth option - **Sign Out**

By clicking "Sing Out" you are directed to the home screen page where you should either sign in or sign up to the application.

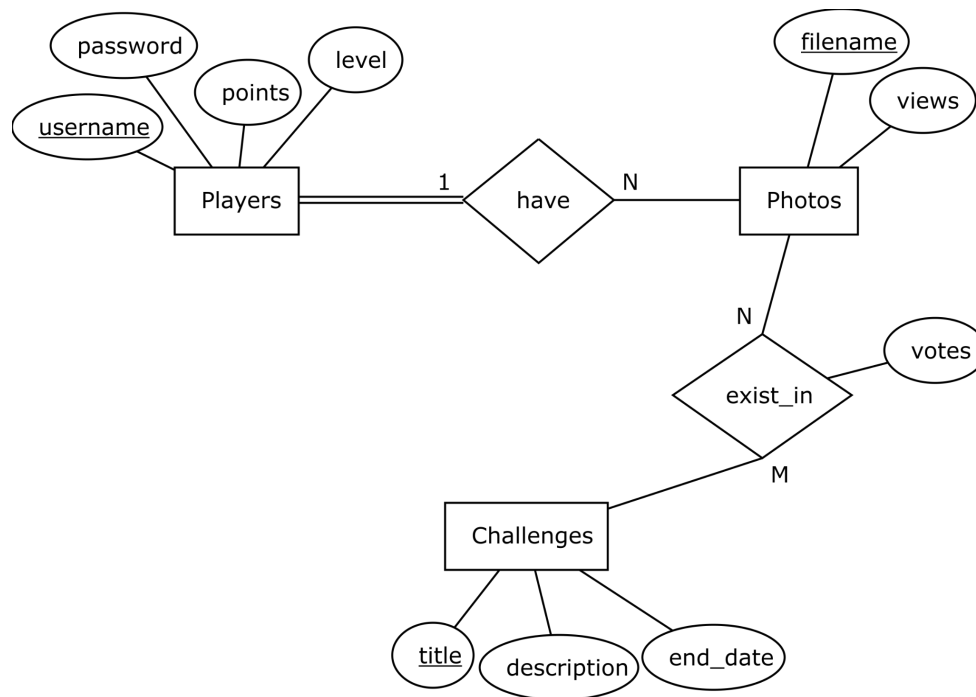
In case you give a wrong username or password on the login page the application returns an error message.



13. Login Error

Entity–relationship (ER) diagram

The ER model of our photography game is illustrated by the diagram below:



There are three entities (Players, Photos and Challenges), each one stored in a different XML file. For example, here is "players.xml":

```
<players>
  <player username="achatzimichail">
    <password>qwerty</password>
    <points>150000</points>
    <level>7</level>
  </player>
  <player username="mliagka">
    <password>maria</password>
    <points>32000</points>
    <level>5</level>
  </player>
</players>
```

Initially, there are two players in the database, with usernames “achatzimichail” and “mliagka”.

As shown in the diagram, there are two relationships, "have" and "exist_in".

Each player can have many photos in his/her profile, but one photo belongs to only one player. Thus, the "have" relationship does not need to be a separate XML file. Username is used as a "foreign key" in photos entries, as shown in "photos.xml":

```
<photos>
  <photo filename="christmas_cat.jpg" fk_username="achatzimichail">
    <views>100</views>
  </photo>
  <photo filename="bird.jpg" fk_username="mliagka">
    <views>241</views>
  </photo>
  <photo filename="laptop.jpg" fk_username="mliagka">
    <views>160</views>
  </photo>
</photos>
```

The second relationship is stored in its own XML file and looks like this:

```
<exist_in_rel>
  <exist_in fk_filename="christmas_cat.jpg"
fk_username="achatzimichail" fk_title="Favorite Animal">
    <votes>59</votes>
  </exist_in>
  <exist_in fk_filename="bird.jpg" fk_username="mliagka"
fk_title="Favorite Animal">
    <votes>24</votes>
  </exist_in>
  <exist_in fk_filename="laptop.jpg" fk_username="mliagka"
fk_title="Technology">
    <votes>35</votes>
  </exist_in>
</exist_in_rel>
```

At that point, there are three entries, each one having information about the photo that participates in a challenge, along with the votes that it has gathered.

Implementation

The user interface and the functionality of the application is developed in Java. There is a [XQuery Processor for Java](#), which is used by the application to run XQuery commands, both to retrieve information and to modify XML files. In simple words, when the user presses certain buttons, some Xquery commands are executed in the background and the results are then shown inside the application.

Xquery examples

To better explain the above, we will go through some examples of Xquery commands that are executed by the application.

One of the first queries that are executed is when a user tries to sign in. More specifically, that query searches for the player with the same username that the user entered and returns his/her password to the application.

Example 1. “getPassword.xq”

```
let $players := doc("players.xml")
for $pl in $players/players/player
where data($pl/@username) eq "achatzimichail"
return data($pl/password)
```

This query returns `qwerty`.

The next query is executed when a user (for example, `achatzimichail`) presses the “Discover Challenges” button.

Example 2. “getDiscoverChallenges.xq”

```
let $takes_part:= (
let $exist_in := doc("exist_in.xml")
for $ex in $exist_in/exist_in_rel/exist_in
where data($ex/@fk_username) eq "achatzimichail"
return $ex/@fk_title/string()
)
return
doc("challenges.xml")/challenges/challenge/@title/string()[not(= $takes_part)]
```

The variable `takes_part` stores all challenge titles that `achatzimichail` joined. Then, the query returns all the challenge titles that are not in `takes_part` sequence. After that, the application creates buttons for the titles returned, as shown in *Image 6*.

There is also the capability of ordering the resulting sequence by some attribute, as shown in the next example, where photos in a challenge are ordered by their votes to form the challenge’s leaderboard.

Example 3. "getLeaderboard.xq"

```
let $exist_in := doc("exist_in.xml")
for $ex in $exist_in/exist_in_rel/exist_in
where data($ex/@fk_title) eq "Favorite Animal"
order by $ex/votes descending
return (data($ex/@fk_filename), data($ex/@fk_username),
data($ex/votes) )
```

Moreover, some queries modify the database. One such example is when a player adds a photo.

Example 4. "addPhoto.xq"

```
insert nodes
  <photo filename="test.jpg" fk_username="test">
    <views>0</views>
  </photo>
as last into doc("photos.xml")/photos
```

On the contrary, if a photo is deleted, it is also deleted from any challenge that it was participating in.

Example 5. "deletePhoto_part_I.xq"

```
for $ph in doc("photos.xml")/photos/photo
where data($ph/@filename) eq "bird.jpg" and
data($ph/@fk_username) eq "mliagka"
return delete nodes $ph
```

"deletePhoto_part_II.xq"

```
for $ex in doc("exist_in.xml")/exist_in_rel/exist_in
where data($ex/@fk_filename) eq "bird.jpg" and
data($ex/@fk_username) eq "mliagka"
return delete nodes $ex
```

Finally, when a player votes for a photo, a modification query like the following is executed.

Example 6. "updateVotes.xq"

```
(: A photo in a challenge was voted. Its votes are increased by 1 :)
for $ex in doc("exist_in.xml")/exist_in_rel/exist_in
where data($ex/@fk_filename) eq "christmas_cat.jpg" and
data($ex/@fk_username) eq "achatzimichail" and
data($ex/@fk_title) eq "Favorite Animal"
return replace value of node $ex/votes
with {xs:int($ex/votes)+1}
```

Conclusion

We have shown how Xquery can be used to access and modify information on XML files. The Java application that was built, implements a photography game and accesses information from a database represented by XML files. Xquery has wide capabilities in modifying XML files, and with this project, we have highlighted a fraction of them.