CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Project Management

Database Design SOFT7022 project

Multiplayer Tic-Tac-Toe Game

Michael Beno

R00155443

Database Design

CO.SD2-A

Cork 2018

Declaration

I hereby certify that this material which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent, that such work has been cited and acknowledged within the text of my work.

In Cork April 27, 2018

……………………………………

Signed

Content:

1 Project description

The purpose of this project is to create the database connection with the minimum of 4 tables which can be used for multiplayer turn games e.g. Chess, Ludo or Tic-Tac-Toe for two players. The database should provide registration and leader-board for registered user. It should provide quick connect without registration. This project will be tested at Tic-Tac-Toe 3x3 game developed in JavaFX.

2 Technical issues dealt with

2.1 Remote access to the database

The first issue was with remote access to the database as there was firewall issue and also SSL connection is required for communication.
For functionality, the port has been forwarded to the server. Connection to the Linux server was made by SSH by Putty.

2.2 Setting grants

Another problem was raised when I try to connect to the database as I didn't have permission to do that.
The grant to the table and new user with password were set in the MySQL shell command line to provide remote access.

2.3 Thread safe JavaFX

The JavaFX doesn't support any Thread or Runnable. Any external thread cannot be created or run outside the main Thread.
For that reason the Timeline was used, which is recommended for games.

2.4 Debugging Thread

The games with running threads are difficult to debug as methods being executed periodically and results are changing. All test was run twice and observed if results are changing correctly in both players applications and in the database.

2.5 JDBC driver error

An occasional unaccepted error occurred when an integer was passed to query without quotes. e.g. *SELECT * FROM Players WHERE pid = 111;*
Caused exception error near ‘ ‘ at line 1…
To avoid these acceptions ware added quotations into the Strings.

3 Database Design

3.1 1NF (First Normal Form)

- Each table cell should contain a single value.
- Each record needs to be unique.

By following rules we create the table.

P1Name	P1Pass	p1status	p1Symbol	P1win	P1loss	P1draw	P1Left	P2Name	P2Pass	P2status	P2Symbol	P2win	P2loss	P2draw	P2Left	GameNo	gBoard	Plturn	timestamp
Joe	jPass	0	x	1	0	0	0	Bob	bPass	0	o	0	1	0	0	1	x,x,x,o,x,x	Joe	2018.1.1
Sam	sPass	0	o	0	0	1	0	Jim	bPass	0	x	0	0	1	0	2	x,x,x,o,x,x	Sam	2018.1.2
Jim	jPass	0	x	2	0	0	0	Sam	bPass	0	o	0	2	0	0	3	x,x,x,o,x,x	Jim	2018.1.3
Bob	jPass	0	o	0	0	1	0	Joe	bPass	0	x	0	0	1	0	4	x,x,x,o,x,x	Bob	2018.1.4

3.2 2NF (Second Normal Form)

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key

In second normal form, we have Single column PK by partitioning table above.

Table 1

id	name	pass	status	type	lastActivity
111	AAA	a	offline	registered	07.04.18 19:31
112	BBB	b	offline	registered	07.04.18 19:31
113	CCC	Cpass	offline	registered	07.04.18 19:31
114	DDD	Dpass	offline	registered	07.04.18 19:31
278	BEDRUNKA	b	offline	registered	07.04.18 19:31
503	GAMMER	g	offline	registered	05.04.18 11:19
600	KKK	Kpass	offline	registered	05.04.18 11:19
885	GAMER	g	offline	registered	07.04.18 19:33
1001	Guest	ifiujifjgi	online	guest	19.04.18 19:33
1002	Guest	kjhkjgtuf	online	guest	19.04.18 19:33

Comment: This table counting with possible future upgrade. The *type* attribute may take variables like: admin, offline-PC…
The variable *lastActivity* may be used for turning user offline when inactive for long time.

Table 2

gameId	player1id	player2id	Symbol1	Symbol2	Gameboard	lastPlayed	gameCreated	lastPlayedTime	gameStat	idWin	idLost
407	1207	1208	O	X	O--------	1210	06.04.18 22:00	06.04.18 22:05	running	0	0
408	1209	1210	O	X	---------	1209	06.04.18 22:05	06.04.18 22:14	finished	0	1209
409	1211	1209	O	X	OXOXOOXXO	1211	06.04.18 22:08	06.04.18 22:14	finished	1209	1211
411	1233	278	O	X	---------	1233	07.04.18 18:27	07.04.18 18:41	finished	0	1233
412	278	885	O	X	OX-OX-O--	885	07.04.18 18:36	07.04.18 19:00	finished	885	278
413	278	278	O	X	OX-OX-O--	885	07.04.18 18:39	07.04.18 19:00	finished	885	278

Comment: Future feature of this table is hidden under *lastPlayedTime* attribute, it may be used for finish the game "lost" result, when player is inactive for long time. We can also calculate length of the game from *gameCreated* and above attribute.

Table 3

pid	called	timeout	status
1211	1209	06.04.18 22:00	pending
1233	278	06.04.18 22:05	refused

Comment: Above table is used for creating the connection only and it doesn't keep records for future. It holds timeout attribute which could be contributed when the user is not responding.

3.3 3NF (Third Normal Form)

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

To move our 2NF table into 3NF, we again need to again divide our table.

Table 1

id	name	pass	status	type	lastActivity
111	AAA	a	0	1	07.04.18 19:31
112	BBB	b	0	1	07.04.18 19:31
113	CCC	Cpass	0	1	07.04.18 19:31
114	DDD	Dpass	0	1	07.04.18 19:31
278	BEDRUNKA	b	0	1	07.04.18 19:31
503	GAMMER	g	0	1	05.04.18 11:19
600	KKK	Kpass	0	1	05.04.18 11:19
885	GAMER	g	0	1	07.04.18 19:33
1001	Guest	ifiujifjgi	1	0	19.04.18 19:33
1002	Guest	kjhkjgtuf	1	0	19.04.18 19:33

Table 2

status	command	description
0	offline	user is offline
1	busy	user is logged in
2	online	user is ready to play

Table 3

type	command	description
0	guest	temporrary made player wirh GENERATED NAME and WITHOUT leaderboard
1	registered	registered user with SELECTED NAME, WITH leaderboard

Table 4

gameId	player1id	player2id	Symbol1	Symbol2	Gameboard	lastPlayed	gameCreated	lastPlayedTime	gameStat	idWin	idLost
407	1207	1208	O	X	O--------	1210	06.04.18 22:00	06.04.18 22:05	31	0	0
408	1209	1210	O	X	---------	1209	06.04.18 22:05	06.04.18 22:14	32	0	1209
409	1211	1209	O	X	OXOXOOXXO	1211	06.04.18 22:08	06.04.18 22:14	32	1209	1211
411	1233	278	O	X	---------	1233	07.04.18 18:27	07.04.18 18:41	32	0	1233
412	278	885	O	X	OX-OX-O--	885	07.04.18 18:36	07.04.18 19:00	32	885	278
413	278	278	O	X	OX-OX-O--	885	07.04.18 18:39	07.04.18 19:00	32	885	278

Table 5

sid	command	description
31	running	game is running
32	finished	game id finished

Table 6

pid	called	timeout	status
1211	1209	06.04.18 22:00	24
1233	278	06.04.18 22:05	26

Table 7

sid	called	description
24	pending	Waiting for answer
25	accepted	Player accepted the game
26	refused	Player refused the game

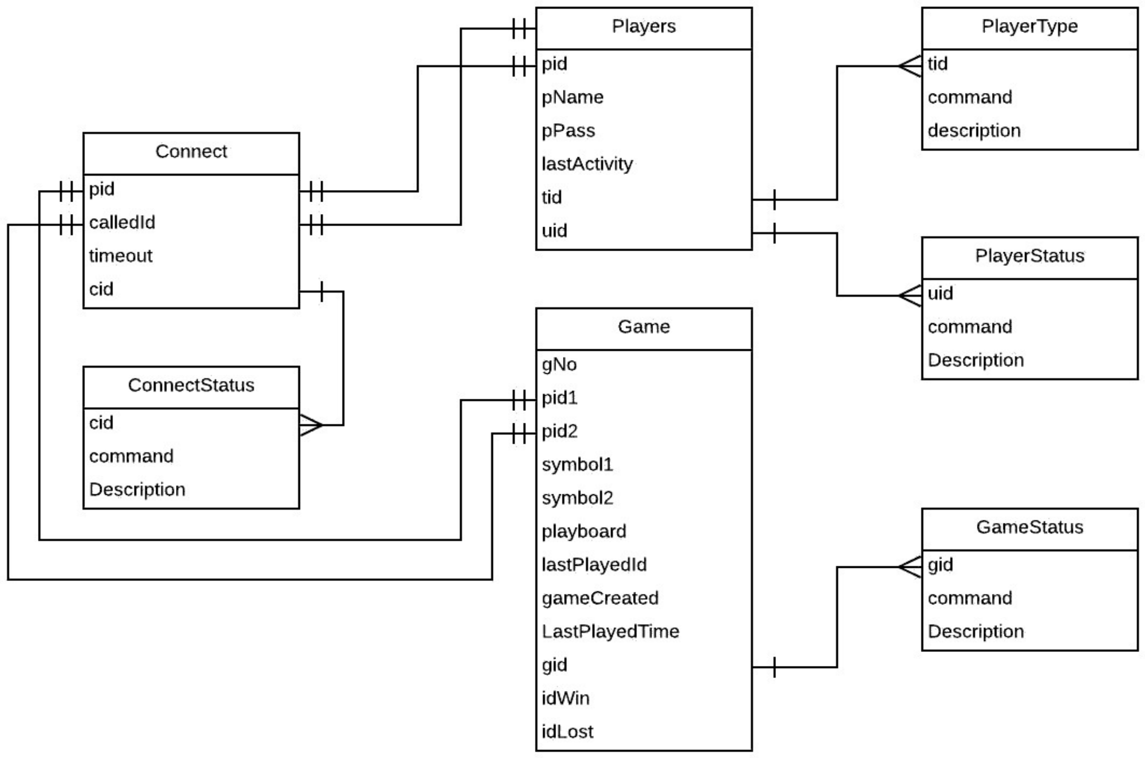
We have again divided our tables and created a new tables which holds status for tables where needed and commands are primary key in new table. There are no transitive functional dependencies, and hence our table is in 3NF.

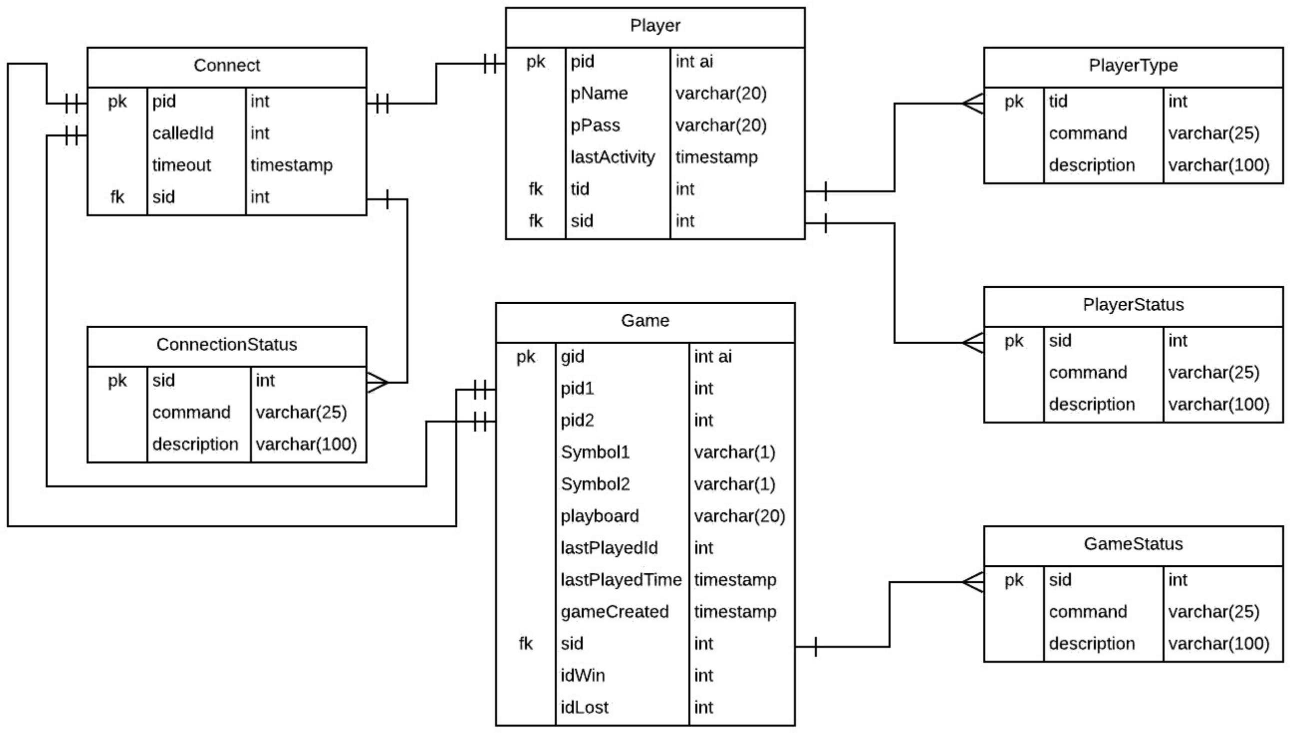
Boyce-Codd Normal Form (BCNF)

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.

We can use extra table for Symbols, but it is not done for simplicity and performance reasons.

3.4 ER modeling





3.5 SQL Queries

3.5.1 SQL Creating tables

```
CREATE TABLE `PlayerStatus` (
  `sid` int(11) NOT NULL,
  `command` varchar(25) DEFAULT NULL,
  `description` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `PlayerType` (
  `tid` int(11) NOT NULL,
  `command` varchar(25) DEFAULT NULL,
  `description` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`tid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Players` (
  `pid` int(11) NOT NULL AUTO_INCREMENT,
  `pName` varchar(20) DEFAULT NULL,
  `pPass` varchar(18) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT
NULL,
  `sid` int(11) DEFAULT '0',
  `tid` int(11) DEFAULT '0',
 `lastActvity` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`pid`)
) ENGINE=InnoDB AUTO_INCREMENT=111 DEFAULT CHARSET=utf8;

CREATE TABLE `GameStatus` (
  `sid` int(11) NOT NULL,
  `command` varchar(25),
  `description` varchar(100),
  KEY `pk` (`sid`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Game` (
  `gid` int(11) NOT NULL AUTO_INCREMENT,
  `pid1` int(11) DEFAULT NULL,
  `pid2` int(11) DEFAULT NULL,
  `Symbol1` char(1) DEFAULT NULL,
  `Symbol2` char(1) DEFAULT NULL,
  `playboard` char(9) DEFAULT NULL,
  `lastPlayedId` int(11) DEFAULT NULL,
  `idWin` int(11) DEFAULT 0,
  `idLost` int(11) DEFAULT 0,
  `lastPlayedTime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  `gameCreated` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `sid` int(11) DEFAULT NULL,
  KEY `pk` (`gid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `Connect` (
  `pid` int(11) NOT NULL,
  `calledId` int(11) NOT NULL,
  `timeout` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `sid` int(11) DEFAULT NULL,
  PRIMARY KEY (`pid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `ConnectStatus` (
  `sid` int(11) NOT NULL,
  `command` varchar(25) DEFAULT NULL,
  `description` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3.5.2 Initiating tables

```
INSERT INTO `PlayerStatus` (`sid`, `command`, `description`)
VALUES ('0', 'offline', 'user is offline');
INSERT INTO `PlayerStatus` (`sid`, `command`, `description`)
VALUES ('1', 'online', 'user is logged in');
INSERT INTO `PlayerStatus` (`sid`, `command`, `description`)
VALUES ('2', 'ready', 'user is ready to play');

INSERT INTO `PlayerType` (`tid`, `command`, `description`) VALUES
('0', 'guest', 'temporrary made player wirh GENERATED NAME and
WITHOUT leaderboard');
INSERT INTO `PlayerType` (`tid`, `command`, `description`) VALUES
('1', 'registered', 'registered user with SELECTED NAME, WITH
leaderboard');

INSERT INTO `Players` (`pName`, `pPass`, `tid`) VALUES ('AAA',
'a', '1');
INSERT INTO `Players` (`pName`, `pPass`, `tid`) VALUES ('BBB',
'b', '1');
INSERT INTO `Players` (`pName`, `pPass`, `tid`) VALUES ('CCC',
'c', '1');
INSERT INTO `Players` (`pName`, `pPass`, `tid`) VALUES ('DDD',
'd', '1');

INSERT INTO `ConnectStatus` (`sid`, `command`, `description`)
VALUES ('0', 'accept', 'Player accepted the game');
INSERT INTO `ConnectStatus` (`sid`, `command`, `description`)
VALUES ('1', 'refuse', 'Player refused the game');

INSERT INTO `GameStatus` (`sid`, `command`, `description`) VALUES
('31', 'running', 'game running');
INSERT INTO `GameStatus` (`sid`, `command`, `description`) VALUES
('32', 'finished', 'game id finished');
```

3.5.3 SQL selects, inserts, updates used in the game.

Getting list of all player which are ready to play excluding me (myID):
```
SELECT * FROM Players
WHERE sid = ‘online’ AND pid != ‘myID’
ORDER BY tid DESC;";
```

Getting list of all registered player excluding me (myID):
```
SELECT * FROM Players
WHERE pPass = 'password' AND pName = 'name';"
```

Getting getting my opponent which want to play with me (myID):
```
SELECT * FROM Connect c ,Players p
WHERE p.pid = calledId AND c.pid = ‘myID’;
```

Getting getting my opponent which want to play with me (myID) and opponent ID is known:
```
SELECT * FROM Connect c, Players p
WHERE calledId = ‘myID’ AND p.pid = ‘opponentID’;
```

Getting getting my opponent from opponent ID if know:
```
SELECT * FROM Players p
WHERE pid = ‘opponentID’;
```

Getting player by given name:
```
SELECT * FROM Players
WHERE pName = 'name';
```

Getting game which is running and contains myID;
```
SELECT * FROM Game g, Players p
WHERE (pid1 = pid or pid2 = pid) and g.sid = 'runnig' and pid =
'myID';
```

Inserting new player, setting name password and setting user offline:
```
INSERT INTO `Players` (`pName`, `sid` , `pPass`)
VALUES ('name', 'offline' , 'password');";
```

Challenging opponent, inserting event that I (myID) want to play:
```
INSERT INTO Connect (pid, calledId, sid)
VALUES (‘myID’, ‘opponentID’, ‘call’);
```

Creating new game, both players, both symbols and setting status to running:
```
INSERT  INTO  `Game`  (`pid1`,  `pid2`,  `Symbol1`,  `Symbol2`,
`playboard`, `lastPlayedId`, gameCreated, sid)
VALUES   ('myID',   'opponentID',   'mySymbol',   'oppSymbol',
'---------', 'myID', now(), ‘runnig’);
```

Updating players status by ID to offline:
```
UPDATE Players set sid = ‘offline’ WHERE pid = 'ID';
```

Updating game;

```
UPDATE Game SET sid = ‘finish’, lastPlayedId = ‘myID’ , playboard
= 'board', idWin = ‘idWin’, idLost = ‘idLost’
WHERE pid1 = ‘pid1’ OR pid2 = ‘pid2’ AND sid = ‘running’;
```

Deleting event from connect by given ID:

```
DELETE from Connect where pid = ‘ID’;
```

Deleting my ID if I am playing as a ‘Guest’:

```
Delete from Players where pid ='myID' and pName = 'Guest';
```

Leader-Board, games wins, lost, draw and played :

```
select pName , SUM(idWin = pid) as wins, SUM(idLost = pid) as lost
, SUM(idLost = 0 AND idWin = 0) as draw ,count(gid) AS
Games_Played
FROM Players, Game g where (pid1 = pid or pid2 = pid) and g.sid =
‘finished’
group by pid
ORDER by wins DESC, draw DESC, lost LIMIT 10;
```

Comment: In the queries are used variables, marked red.

3.5.4 Special queries

```
#SET SQL_SAFE_UPDATES = 0;
#SET SQL_SAFE_DELETE = 0;
```

Comment: These two queries was used to allow multiple update or delete.

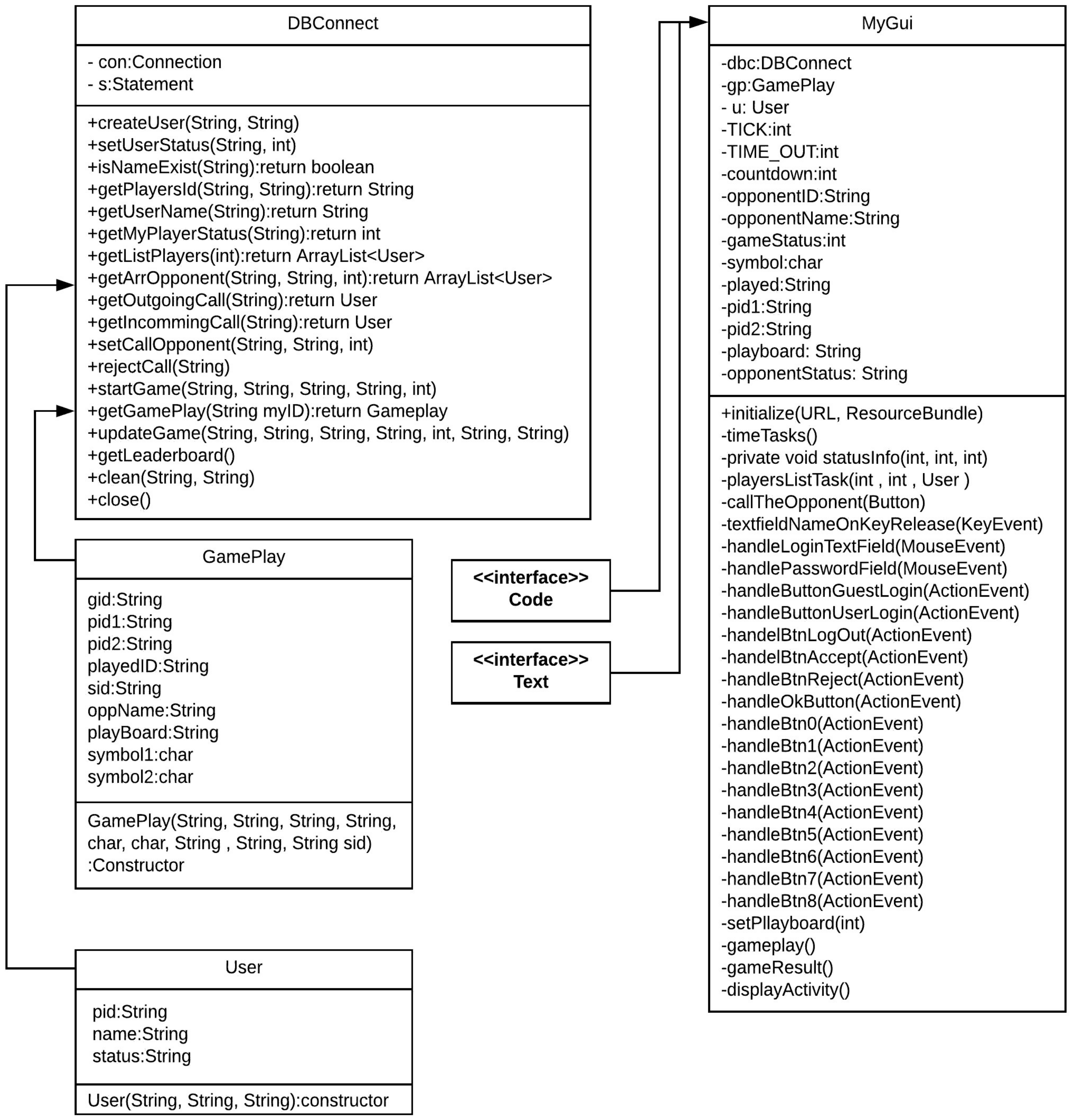
```
update Players set sid = 0 where now() > adddate(
lastActvity,INTERVAL 5 MINUTE) and sid > 0 and tid > 0; #change
all unactive players to offline

delete from Players where now() > adddate( lastActvity,INTERVAL 5
MINUTE) and tid = 0 ;#and sid >= 0 ; #delete all guests which
arent active after 5minutes
```

Comment: Above two special queries were never been used. First one turns every user to offline status if is not active for longer than 5 minutes. It will require changing the checking state method to the update state method.
The second query deletes all temporary (not registered) users which have status ‘offline’ longer than 5 minutes.

4 Application design

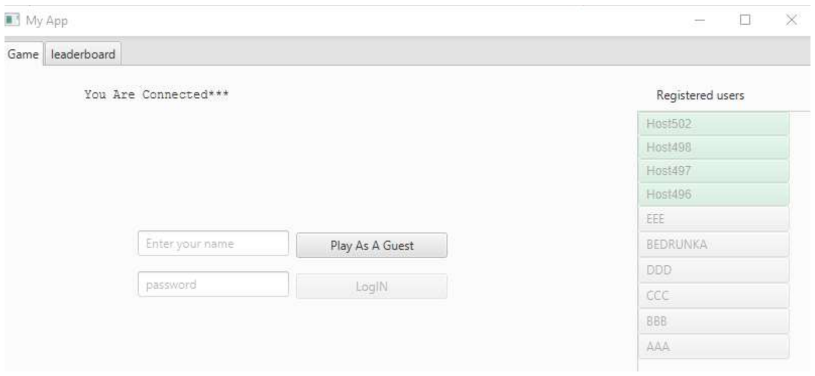
4.1 UML Diagram


DBConnect
- con:Connection
- s:Statement
+createUser(String, String)
+setUserStatus(String, int)
+isNameExist(String):return boolean
+getPlayersId(String, String):return String
+getUserName(String):return String
+getMyPlayerStatus(String):return int
+getListPlayers(int):return ArrayList<User>
+getArrOpponent(String, String, int):return ArrayList<User>
+getOutgoingCall(String):return User
+getIncommingCall(String):return User
+setCallOpponent(String, String, int)
+rejectCall(String)
+startGame(String, String, String, String, int)
+getGamePlay(String myID):return Gameplay
+updateGame(String, String, String, String, int, String, String)
+getLeaderboard()
+clean(String, String)
+close()
GamePlay
gid:String
pid1:String
pid2:String
playedID:String
sid:String
oppName:String
playBoard:String
symbol1:char
symbol2:char
GamePlay(String, String, String, String, char, char, String , String, String sid) :Constructor
User
pid:String
name:String
status:String
User(String, String, String):constructor
<<interface>> Code
<<interface>> Text
MyGui
-dbc:DBConnect
-gp:GamePlay
- u: User
-TICK:int
-TIME_OUT:int
-countdown:int
-opponentID:String
-opponentName:String
-gameStatus:int
-symbol:char
-played:String
-pid1:String
-pid2:String
-playboard: String
-opponentStatus: String
+initialize(URL, ResourceBundle)
-timeTasks()
-private void statusInfo(int, int, int)
-playersListTask(int , int , User)
-callTheOpponent(Button)
-textfieldNameOnKeyRelease(KeyEvent)
-handleLoginTextField(MouseEvent)
-handlePasswordField(MouseEvent)
-handleButtonGuestLogin(ActionEvent)
-handleButtonUserLogin(ActionEvent)
-handelBtnLogOut(ActionEvent)
-handelBtnAccept(ActionEvent)
-handleBtnReject(ActionEvent)
-handleOkButton(ActionEvent)
-handleBtn0(ActionEvent)
-handleBtn1(ActionEvent)
-handleBtn2(ActionEvent)
-handleBtn3(ActionEvent)
-handleBtn4(ActionEvent)
-handleBtn5(ActionEvent)
-handleBtn6(ActionEvent)
-handleBtn7(ActionEvent)
-handleBtn8(ActionEvent)
-setPllayboard(int)
-gameplay()
-gameResult()
-displayActivity()

4.2 GUI design

4.2.1 Connection Info

The connection is established on startup and all players names are listed on the right. Status is color coded. eg. Green – ready to play. Grey – registered but offline.

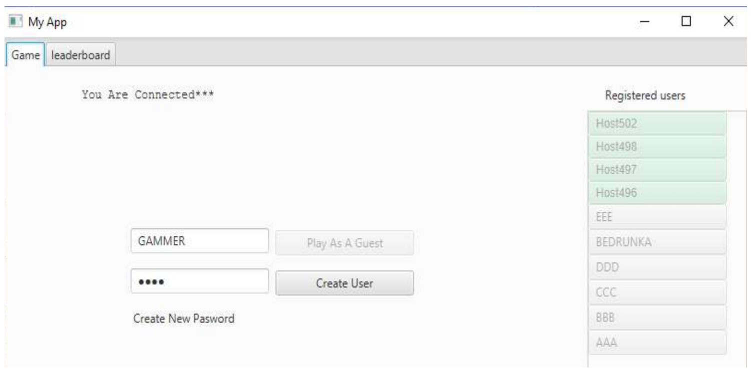


You Are Connected***

Host502
Host498
Host497
Host496
EEE
BEDRUNKA
DDD
CCC
BBB
AAA

4.2.2 Creating new user

When we fill up the name in the text field and click into the password field, the application will check the name in the database and respond „Create New Password“. The button is also turned to „Create User“ text.

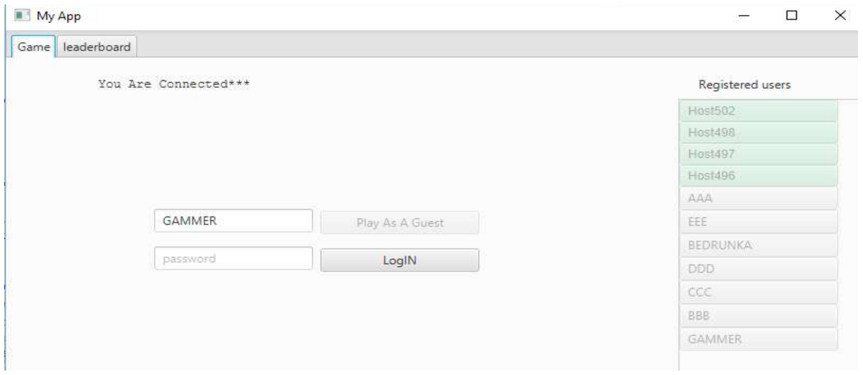


You Are Connected***

GAMMER
••••
Create New Pasword

Host502
Host498
Host497
Host496
EEE
BEDRUNKA
DDD
CCC
BBB
AAA

By clicking button is new user created in the database and name is listed with offline status (grey color). The user can now enter his password to log-in. The button is changed to „Login“ state as this name is in the database.



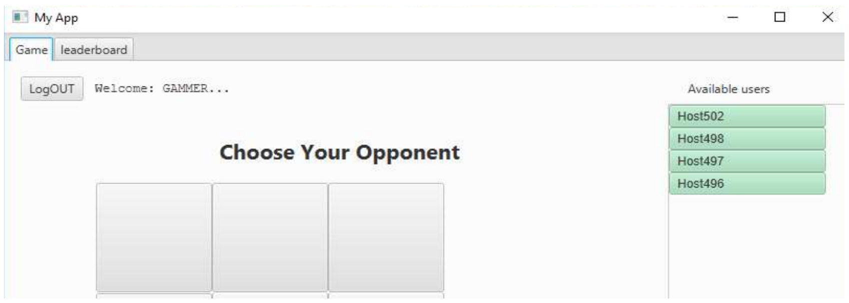
4.2.3 Log-in as User

By entering the name and correct password which corresponding to the database, the application let you in. Otherwise, inform you of „wrong password“ message.

The „LogOUT“ button and welcome text with your name indicates that you are logged in and application displaying only ready to play users.

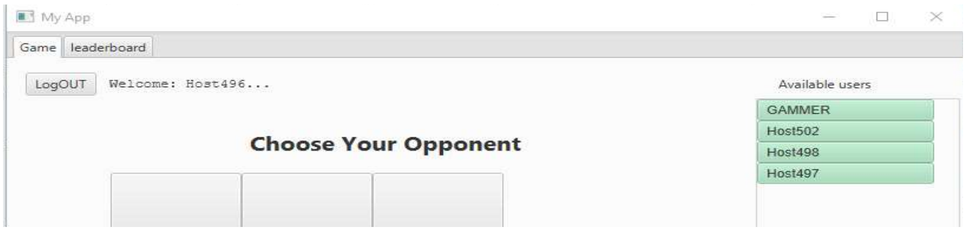
When a user starts typing name the Guest button is disabled.

This user has information in the database that it is registered user and the result of the game will be stored in the leaderboard.



4.2.4 Login as Guest

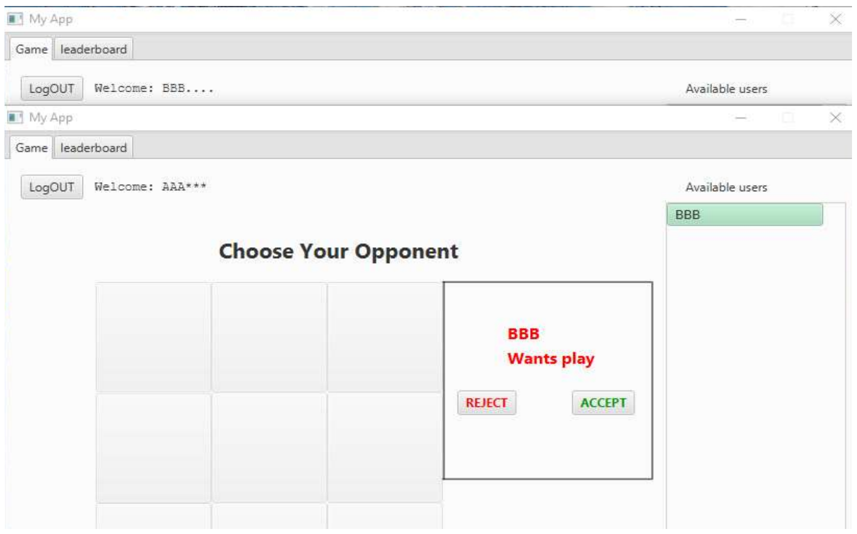
A guest can play same as the registered user and can play with any other available players, but the nickname is generated by the application and it will be removed when log-out. Autogenerated nicknames deosn't appear on the leaderboard.



4.2.5 Creating Game

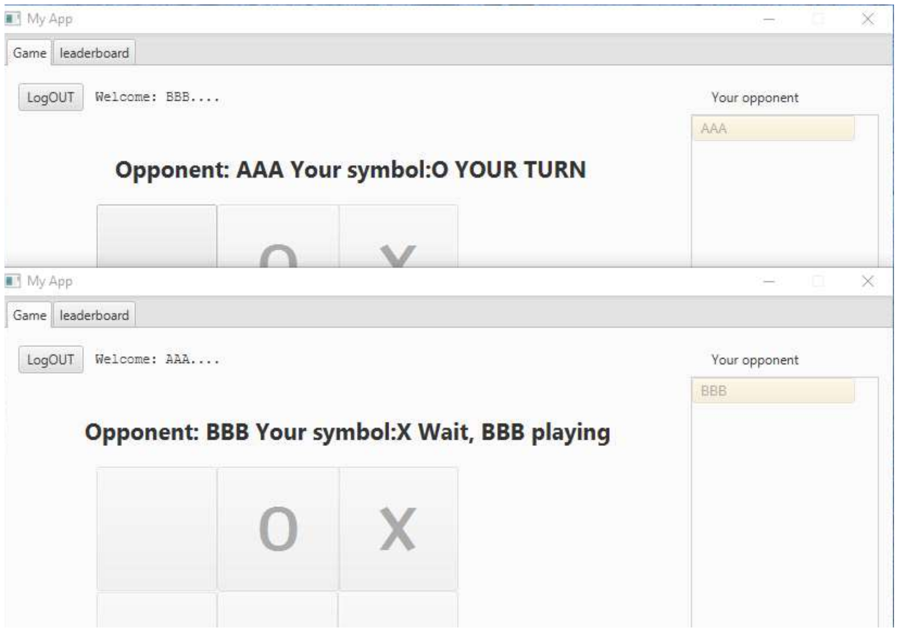
On the below picture the player “BBB” want to play with player “AAA”.

When the game is rejected, The player can choose from other players or try to ask the same player for play again.



4.2.6 Gameplay

When the game is accepted, players have to finish the game with the win, lost or draw result. However if one of the players leave the game before finish it will make the result as lost and win for the opponent.



```
Welcome: BBB....
```

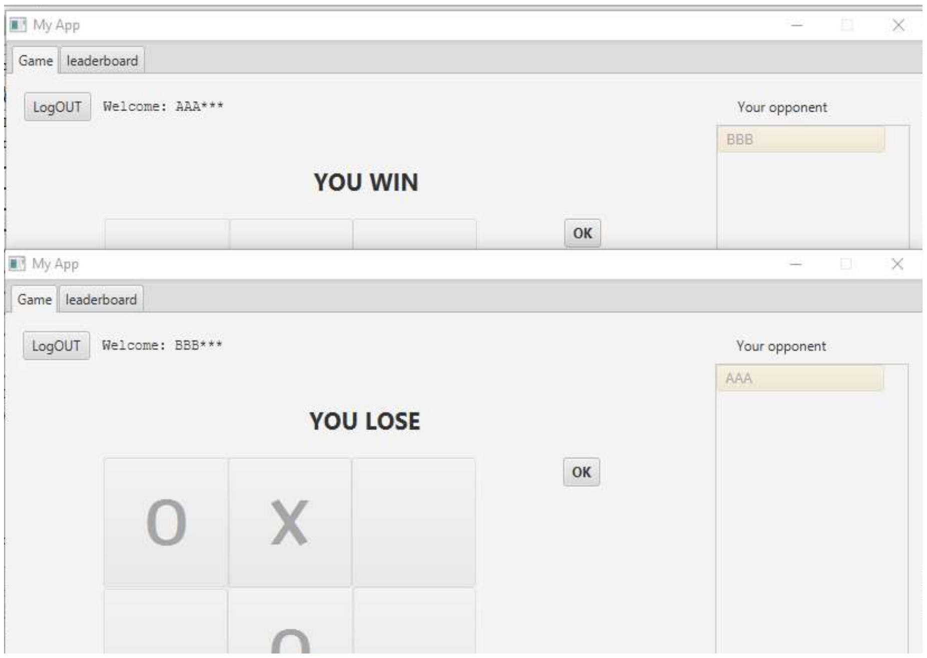
Opponent: AAA Your symbol:O YOUR TURN

```
Welcome: AAA....
```

Opponent: BBB Your symbol:X Wait, BBB playing

4.2.7 Game Over

This example showing a player who loose and winner. Both players have “OK” button available to confirm his result and play again.



```
Welcome: AAA***
```

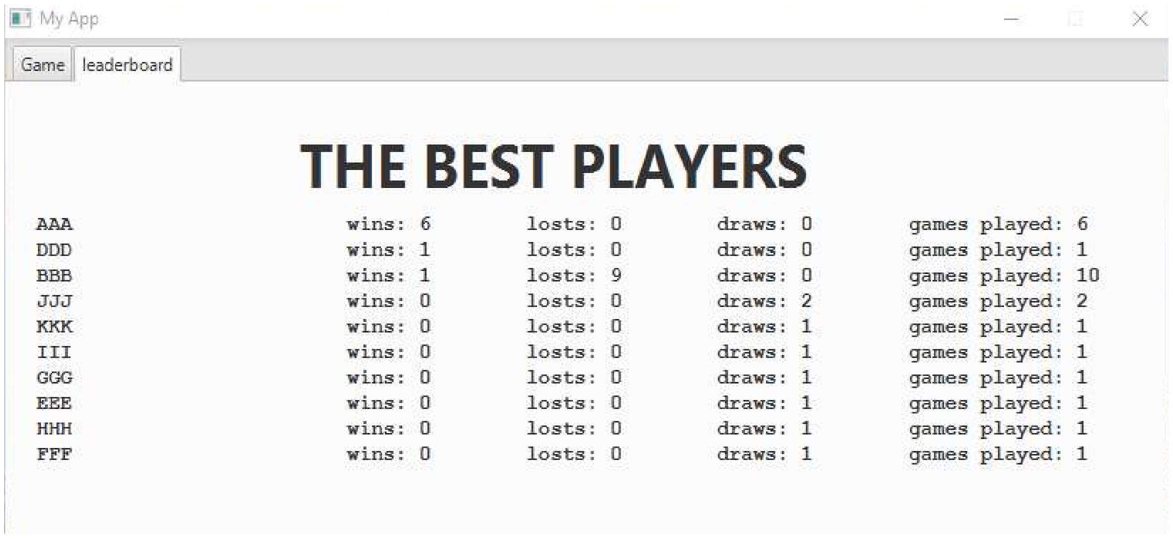
YOU WIN

```
Welcome: BBB***
```

YOU LOSE

4.2.8 Leader Board

Registered users can find their result on Leader Board which holds top ten best players and achievements are sorted by how many winnings and following draws are made.



THE BEST PLAYERS

```
AAA          wins: 6     losts: 0     draws: 0     games played: 6
DDD          wins: 1     losts: 0     draws: 0     games played: 1
BBB          wins: 1     losts: 9     draws: 0     games played: 10
JJJ          wins: 0     losts: 0     draws: 2     games played: 2
KKK          wins: 0     losts: 0     draws: 1     games played: 1
III          wins: 0     losts: 0     draws: 1     games played: 1
GGG          wins: 0     losts: 0     draws: 1     games played: 1
EEE          wins: 0     losts: 0     draws: 1     games played: 1
HHH          wins: 0     losts: 0     draws: 1     games played: 1
FFF          wins: 0     losts: 0     draws: 1     games played: 1
```

5 Plan/log

date	plan	planed finish date	finish date	delay	reason for delay
26/03/2018	Having the working application done.	01/04/2008	07/04/2008	3days	1 day with setting up server and 2 days with Java Threads
26/03/2018	Setting up the server.	26/03/2018	27/03/2018	1 day	The difficulty with SSL, firewalls, and grants for the tables.
27/03/2018	Learn, how to use Scene Builder with JavaFXML	29/03/2018	29/03/2018	0	
29/03/2018	„Hallo Database“ JavaFX Testing Application	30/03/2018	01/04/2018	2 days	Threads didn't work, was looking for alternatives.
01/04/2018	ER Modeling. Making all tables	02/04/2018	02/04/2018	0	
03/04/2018	Console Application design	03/04/2018	03/04/2018	0	
04/04/2018	GUI Application design	04/04/2018	04/04/2018	0	
05/07/2018	Developing application	07/04/2018	07/04/2018	0	
09/04/2018	Testing game	09/04/2018	09/04/2018	0	
13/04/2018	Setting up and testing the game in the lab. Playing with a classmate.	13/04/2018	13/04/2018	0	
23/04/2018	Having all documentation done and submit the project on Friday.	28/04/2018	28/04/2018	0	

6 Conclusion / review

I learned from this project, which didn't go smoothly as I accepted, how to setup the server to make the database for a multiplayer game. How to use Scene Builder with Java FXML and Java Timeline. However, from now I can use this documentation to create any turn multiplayer game for two players and I can benefit from this experience in the future.

If I didn't have a difficulty with Java Threads and setting up my server, my application would be done as my plan was set.

7 References

GRANT Syntax from MySQL

https://dev.mysql.com/doc/refman/5.7/en/grant.html

Using JavaFX Scene Builder with Java IDEs

https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm

Creating Transitions and Timeline Animation in JavaFX

https://docs.oracle.com/javafx/2/animations/basics.htm