HazzardHulen.io

Anders Bundgaard
abund14@student.sdu.dk

Casper Beese Nielsen
casni14@student.sdu.dk

Oliver Vestergaard
olves13@student.sdu.dk

Simon Hjortshøj Larsen
sila114@student.sdu.dk

Steen Schütt
stesc14@student.sdu.dk

Thomas Lemqvist
thlem14@student.sdu.dk

November 21, 2016

Contents

1 The Software

HazardHulen.io is a multiplayer blackjack table, where players can bet virtual currency to participate in a round of blackjack. The game uses the same ruleset as regular blackjack played in casinos, with a few details left out for simplicity. Players play against an artificial intelligence which acts as the dealer, and win double their bet if they beat the AI. If a player loses, his bet is lost. All players get a set amount of virtual money when they join the table, and have to place a bet of at least a minimum amount to play. If a player does not place a bet within a certain time period in between rounds, he is labeled as inactive, and cannot take part in the next round.

2 Technology

This section describes the technologies used to implement this system.
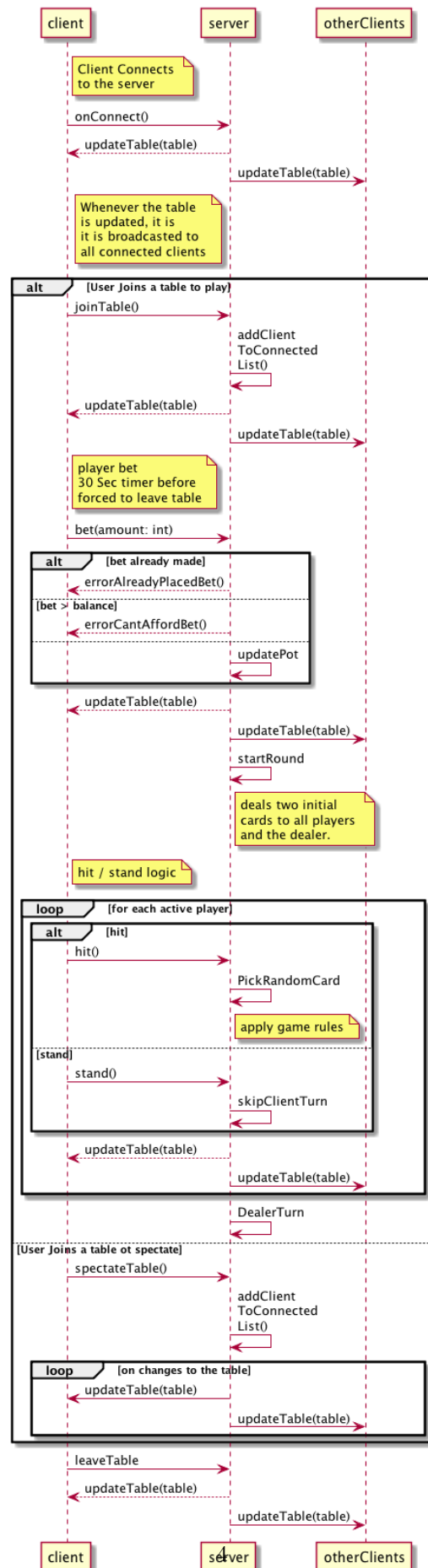
2.1 Client Side

The client is written in *HTML* and *jQuery*, which uses *socket.io*, which is a web socket library, to connect to the server.

2.2 Server Side

The server is written in *JavaScript* and uses *socket.io* to communicate with the clients.

The server utilizes the duplex connection, provided by *socket.io* to synchronize the state of the game across all clients

3   UML Sequence Diagram

client → server → otherClients

Note: Client Connects to the server

client → server: onConnect()
server → client: updateTable(table)
server → otherClients: updateTable(table)

Note: Whenever the table is updated, it is it is broadcasted to all connected clients

alt [User Joins a table to play]

client → server: joinTable()
server → server: addClientToConnectedList()
server → client: updateTable(table)
server → otherClients: updateTable(table)

Note: player bet 30 Sec timer before forced to leave table

client → server: bet(amount: int)

  alt [bet already made]
  server → client: errorAlreadyPlacedBet()
  [bet > balance]
  server → client: errorCantAffordBet()
  server → server: updatePot

server → client: updateTable(table)
server → otherClients: updateTable(table)
server → server: startRound

Note: deals two initial cards to all players and the dealer.

Note: hit / stand logic

loop [for each active player]

  alt [hit]
  client → server: hit()
  server → server: PickRandomCard
  Note: apply game rules
  [stand]
  client → server: stand()
  server → server: skipClientTurn

  server → client: updateTable(table)
  server → otherClients: updateTable(table)

server → server: DealerTurn

[User Joins a table ot spectate]

client → server: spectateTable()
server → server: addClientToConnectedList()

  loop [on changes to the table]
  server → client: updateTable(table)
  server → otherClients: updateTable(table)

client → server: leaveTable
server → client: updateTable(table)
server → otherClients: updateTable(table)

The sequence diagram, shown above, shows all calls made between the server and the client.

The server is responsible for synchronizing the table across all clients. It uses the table object *(as seen in the object diagram below)* for this, in the updateTable method.

4   UML Object Diagram
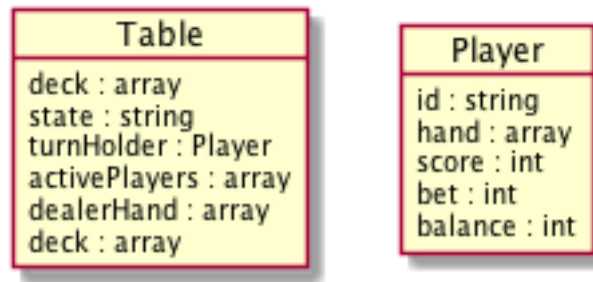
The object diagram contains the two key objects used in this project, *Table* and *Player*.

* The types set in the fields in the objects, are not according to UML standards.

5   Walkthrough of key Code-snippets

5.1   The Connection between the client and server

```
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io')(server);

// ... omitted for brevity

server.listen(13337);
```

The code snippet above is responsible for making the serve serve the clients the last line, sets the port, on which is listened on for requests to the server.

```
var port = 13337;

// ... omitted for brevity,

        var socket = io.connect("http://localhost:" + port + "/");
```

The code snippet above is taken from the client, and makes it connect to the server, in this case it connects to the localhost, on the port 13337.

5.2 The calls from the Client to the Server

```javascript
io.on('connection', function (client) {

  // ... omitted for brevity

    client.emit('setId', id);
    client.emit('updateTableState', table);

    client.on('joinTable', function () {

      // ... omitted for brevity

    });

  // ... omitted for brevity

});
```

The code snippet above is taken from the server, and it is responsible for taking commands from the client.

6 How to Run

To run the server, you have to have npm (it can be downloaded along with node here: https://nodejs.org/en/download/).

Run the following commands to start the server:

```
$ npm install
```

Install bower by running this command, in the command line to install bower: (from: https://bower.io)

```
$ npm install -g bower
```

To start the server, now you simply just has to run the following command.

```
$ npm start
```

Now the server should be up and running.

To connect a client to the server, go to http://localhost:13337/

7 References

npm packages

https://nodejs.org/en/download/
https://bower.io
https://lodash.com
http://socket.io
https://www.npmjs.com/package/monolog

Helpful Links

http://www.programwitherik.com/socket-io-tutorial-with-node-js-and-express/

Information about Blackjack

https://en.wikipedia.org/wiki/Blackjack