| | | | | |
|---|---|---|---|---|
| Computer Engineering | 3rd Year | 1st Semester | 2015-16 | Periodic Evaluation |
| **Final Project** | Limit date for results divulgation: 26 January 2016 | | | |
| Date: 12 Nov. 2015 | **Delivery Date: 4 January 2016** | | | |

# Project 4 – Multiplayer Memory Game

## 1 OBJECTIVE

This project's objective is to implement a multiplayer memory game, using REST architecture and making use of frameworks and technologies such as Laravel, AngularJS, Node.js, Server-Sent Events (SSE) or WebSockets.

Despite the fact that this project implements the same game that was developed in Project 3, there are substantial differences that are pointed out in the Scenario section.

## 2 SCENARIO

The memory game objective is to find pairs of similar tiles, where tiles are represented by images (students can use the provided images or create their own set of images). In the multiplayer version, two or more players can play the game against each other.

Tiles are displayed randomly on the game board. In each game there are always 2 tiles (one pair) with the same face (image). At the beginning of the game, every tile has its face hidden. On each game move, the player must choose two tiles and flip them (the images on the tiles become visible). If a user selects two tiles that have the same face (a pair), those tiles are removed from the game board and the user plays again. If the two tiles don't have the same face, after 1 second they return to their hidden state and the next player plays another move. The game ends when there are no more tiles in the game board.

The goal of the game is to find the maximum number of pairs, in the shortest time possible and with the minimum number of moves. The winner of the game is the player who discovers more pairs. If two or more players discover the same number of pairs, the game winner is the player with the shortest time playing. A player is considered to be playing from the time the tiles from the previous move returned to the hidden state until the moment that the two selected tiles return

to the hidden state or the game is finished. When it's the first move of the game, for the first player, the play time starts only when the player choses the first tile.

Have in mind that in the Final Project, the board size is dynamic. This means that the user can choose the size of the game board. Nevertheless, consider that:

- Each row/column can display 2 to 10 tiles;
- The maximum number of tiles is 80 (40 pairs);

The game board should always have an even number of tiles. During the game, it should be displayed information concerning the game as a whole, namely the elapsed time, the number of moves, the number of matched pairs, the number of remaining tiles on the board and the current player name (nickname). Also, it should display information specific for each player, including the number of matched pairs, the number of moves, and the total elapsed playing time. The information relative to each player must be highlighted to himself.

## 2.1 IDENTITY AND AUTHENTICATION

In order to play the game, the user has two options: login to his game account, or play as a guest. It will be necessary to develop a sign-up mechanism where the user must provide the following information:

- Nickname – A unique nickname – it will be used to perform the login;
- Password – The password must have at least 3 characters;
- Re-type Password – It must be equal to the password;
- E-mail – the user must provide a valid e-mail.

If the user opts to play as a guest, a temporary nickname must be created that should be valid through his session.

## 2.2 GAME LOBBY

The game lobby allows a user to create, a (pending) game, to join a pending game or watch games created by someone else. The list of games on the lobby should include the pending games (still waiting for players) and the active games (games that are currently being played). Users can join only pending games and watch (without interfering) any (pending or active) public game.

When a pending game starts, it becomes an active game. The moment when a game begins

depends on the project implementation. For instance, it might start when a specified number of human players join the game, after a certain amount of time has elapsed, or by decision of the user that has created the game. Project groups are encouraged to decide this logic.

When a user creates a game it is necessary to specify the game board size and whether the game is public or private. Public games can be viewed by any user and private games are only accessible to its players (other users cannot view/watch private games).

Groups have to decide what options to implement, such as if there is a mandatory or maximum number of players and/or bots; whether it's possible to replace human players for bots; how the game starts; the time that two different tiles are visible (1 second by default), etc.

## 2.3 TOP TEN TABLE

The application should display a table with the top ten players of all time, ordered by the number of victories of each player. In case of a tie, the first player to achieve the number of victories appears first on the table.

## 2.4 MULTIPLE GAMES

This feature should allow users to play or watch multiple games simultaneously. This means that on the same web page, several game boards are displayed and several games take place concurrently and independently.

## 2.5 OTHER FEATURES

Students are encouraged to implement additional features to improve the game experience. Some suggestions are described in this section (they are not mandatory), but students should consider other features that are not explicit here and that add value to the project.

### 1. PLAYING AGAINST BOTS

Allow users to play against bots, whenever there are no other users to play, or a user does not want to play against other human players.

It is not important for the bot to be very smart, so, you can opt for taking a simple approach on the bot artificial intelligence algorithm (e.g. each time that the bot plays, it picks two random tiles, regardless of the game's previous moves).

### 2. LOGIN WITH EXTERNAL AUTHENTICATION SERVICES

Allow users to login using the Facebook Login for the Web, Google Identity Platform, Twitter Sign In, etc. You can get more information about this at:

- https://developers.facebook.com/docs/facebook-login/web;
- https://developers.google.com/identity/
- https://developers.google.com/identity/sign-in/web/sign-in
- https://dev.twitter.com/web/sign-in

### 3. CHAT ROOMS

Allow users to communicate with each other through chat rooms. Chat rooms may be accessible to all users and/or associated to games. In this case, they are accessible only to the players of each game.

### 4. LOGS AND HISTORY

This feature implies that a log of every game played is stored in the system. The log of each game may include only information related to the final result of the game, or it may also include every move played in the game. Users should be able to access all/his game logs, extract and analyze historical information about all/his played games. Also, it may be able to review each game by reproducing all moves of the game.

### 5. MOBILE APP

This feature should allow users to play the game using a mobile client specific application. This feature implies the usage of a multi-platform framework, such as "React Native", "PhoneGap", "Ionic", among others.

Please note that applying responsive design to the web application to allow access through a mobile browser, is not considered a mobile client specific application. Responsive design should be applied by default to the *"normal"* Web Application.

# 3 PROJECT

In addition to the functional requirements specified on the scenario, the project must follow some technological constraints and some non-functional requirements related to the application and code structure and to the interaction between the Web client and Web server.

### 3.1 TECHNOLOGICAL CONSTRAINTS

It is recommended that the client side of the application uses AngularJS framework. However, other client technologies, libraries or frameworks may be incorporated on the project.

The server side of the application must be implemented using only technologies and frameworks available on the Homestead vagrant box. Project must use Laravel 5.1 MVC framework and, if required, Node.js server. If the project requires the usage of a relational database, MySQL server must be used (instead of PostgreSQL or SQLLite).

### 3.2 NON-FUNCTIONAL REQUIREMENTS

Several non-functional requirements should be considered, namely:

- The application's architecture should follow good design practices and the technologies used for each of its components should be chosen appropriately, according to the architecture and usage context;
- Application's structure and code should follow the separation of concerns principle, where different software units (files, modules, objects, functions, etc.) must be as independent and loosely coupled as possible;
- The speed of the interaction of the application should be optimized. For example, game moves or chat messages should be propagated almost instantly to all application's clients;
- The application should be as secure as possible, ensuring not only that resources and features are only available to the proper users, but also that every game move associated to a human player is really executed by that human - external elements should not be able to interfere in the game;
- The application must ensure that players, or any other entities, may not "cheat" the game by analyzing its internal state on the client or by any other mean possible.

## 4 PROJECT CONFIGURATION AND DELIVERY

Project configuration and delivery must follow some rules.

### 4.1 HOMESTEAD CONFIGURATION

To ensure that professors can install and evaluate multiple projects simultaneously on their own testing environment (Homestead vagrant box), it is **mandatory** that each group has an

independent Homestead configuration. For that purpose, each group is assigned a unique number that is available on the Curricular Unit Moodle's page.

**Note**: contact the professor if your group has no number assigned, or if there is any mismatch in the compositions of groups.

Considering XX as the group number with 2 digits (e.g. 01, 02, …, 10, 11), apply the following rules to configure your project:

- All project's code (Laravel application, Node.js code, etc.) must be located on the Homestead virtual machine folder: **`/home/vagrant/grpXX`**
- The Laravel application code must be located on the Homestead virtual machine folder: **`/home/vagrant/grpXX/laravel`**
- The project site must be mapped to **`grpXX.dad`**. It should be possible to access the application website through this address: **`http://grpXX.dad`**
    - Note: Don't forget to map this address using the hosts file ("`/etc/hosts`" on OSX or Linux, "`c:\windows\system32\drivers\etc\`" on Windows)
- The project site folder on the Homestead virtual machine must be mapped to: **`/home/vagrant/grpXX/laravel/public`**.
- MySQL database (if used) name must be: **`bd-grpXX`**
- Default user names and passwords for the Homestead virtual machine and MySQL server should not be modified.

Applying the previous rules imply, among other things, that `Homestead.yaml` configuration file (usually located on `~./homestead` folder) includes the following segments of code:

```
. . .

folders:
    - map: … any local folder on the student machine … (e.g. c:\prjDAD)
      to: /home/vagrant/grpXX
. . .

sites:
    - map: grpXX.dad
      to: /home/vagrant/grpXX/laravel/public
. . .

databases:
    - bd-grpXX
```

### 4.2 DELIVERY

The project must be delivered on a **single zip file**, using the provided link in the Curricular Unit Moodle's page. The zip file should contain:

- **"info.txt"**: file with info on the project's group, namely, the number, name, laboratorial class of the students;
- **"Homestead.yaml"**: file with the configuration of the Homestead vagrant box (a copy of the Homestead.yaml file of the student's machine, usually located on `~/.homestead` folder);
- folder named **"grpXX"**: folder containing all the files of the project (exact copy of the group folder "grpXX" in the Homestead vagrant box);
- **"bd-grpXX.sql"**: file with the backup script for the MySQL database. Script should include the instructions required to create the database ("create database . . . "). If a relational database was not used, this file is not required.

# 5 EVALUATION

The following table summarizes the grading criteria for evaluating the project.

| Nº | Weight | Criteria |
|---|---|---|
| 1 | **45%** | Game and Core Application |
| 2 | **10%** | Identity and authentication |
| 3 | **10%** | Game Lobby |
| 4 | **5%** | Top Ten |
| 5 | **10%** | Multiple games |
| 6 | **20%** | Other Features<br>Classification depends on the quantity, originality and usefulness of the features implemented, as well as the complexity and quality of the implementation |

All of the criteria consider the resulting application functionality, usability, reliability and maintainability, as well as the non-functional requirements.