Play and Win

Introduction

The product is an interactive website, featuring an array of games that reward the player with points based on their performance. These points are then converted into virtual coins that can then be used in an onsite webstore which features a variety of digital (or physical) goodies, such as player badges, virtual stickers and gift cards. The purpose of these offerings is to keep players hooked on the site and in competition with their peers. To promote these social interactions as well as grow the site and gain more active users, players will also receive points for recruiting others.

In this report, first the key terms used in the project are defined. Then the functionality of the product is explained and finally, the architecture of the product is introduced.

Glossary

Account: A set of player's attributes. An account includes the user profile, personal information and the amount of coins available.

Admin: An admin is a user that can manage webstore and players.

Advertisement: An advertisement is a banner on the site that advertises a company or a product.

Buying: The process of acquiring products from the webstore with coins.

Coin: A coin is virtual currency that can be earned by playing and can be used in the webstore to buy products. The game score will be converted into coins.

Friend: A friend is a player that is linked to another player. Both players need to accept friendship. A player can see a list of his/her friends on the site.

Friend request: A friend request is a request that a user has received from another user but has not yet approved it.

Game: A game is a short game that can be played on the site.

High Score: The highest score personally achieved in a specific game by the user.

Logging in: Logging in is a process where a user gives his/her username and password to access his/her account and the website.

Pending friend request: A pending friend request is a request that a user has sent to another user, but has not been approved yet.

Player: A player is a user who can play games and use the webstore.

Point: A point is an achievement from a game event. Each game has its own logic how points are given. One hundred points corresponds to one coin.

Product: A product is a physical or a virtual good that can be bought in the webstore with coins. A product has a value in coins.

Profile: A profile is a personal page for a player to let other players know about themselves.

Ranking: A ranking is a list of the top 10 scores per game.

Registration: Registration is a process where a user gives his/her username, password and email address to create an account.

Score: The score is the amount of points that the player gets by playing a game.

Sponsors: Sponsors are people or companies who buy advertising space by providing products to the webstore.

User: A user is a person who has registered to the site, and has an account.

Website: A website is a collection of webpages including the games, the webstore and the user profiles.

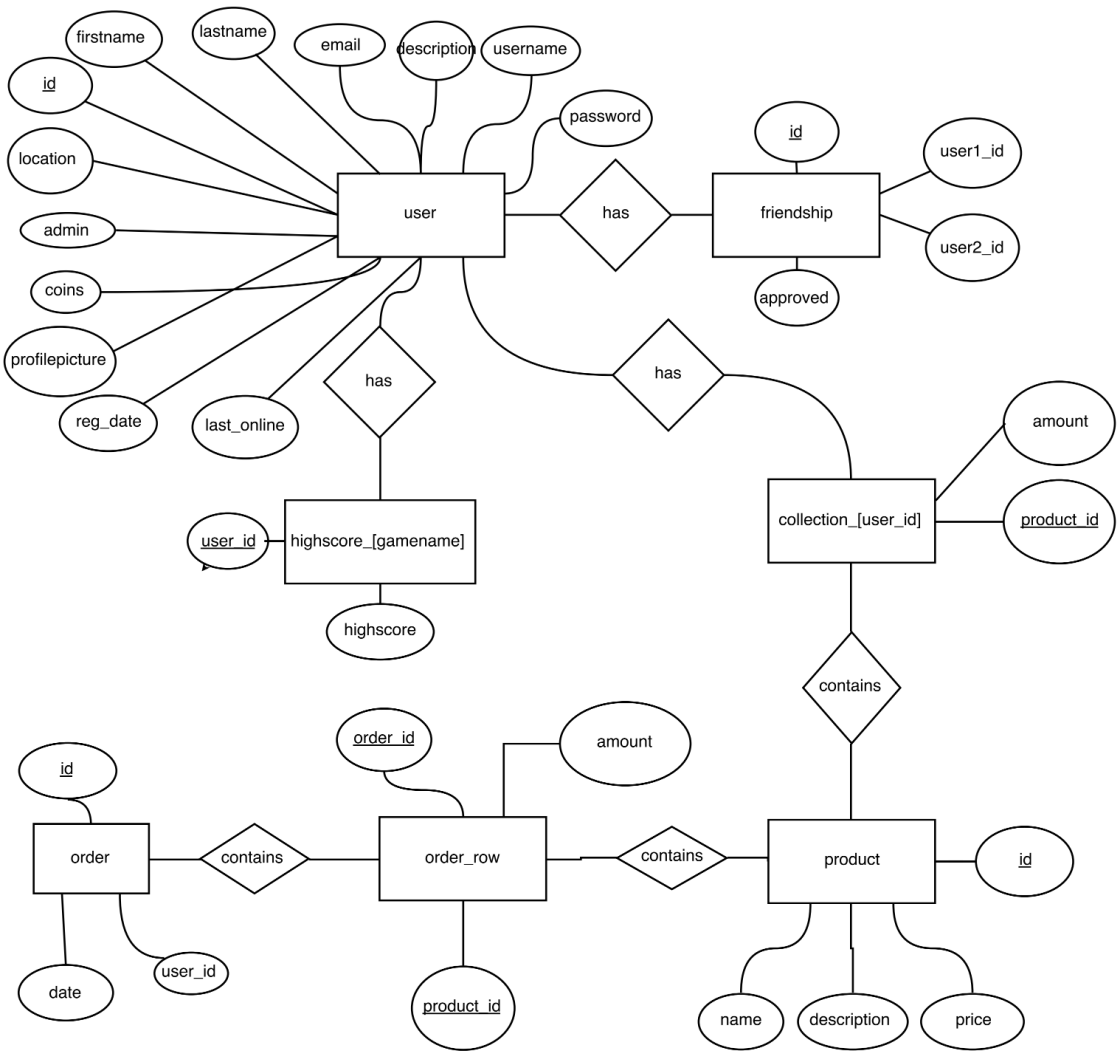
Webstore: The webstore is a place on the site, where products can be bought with coins.

Product functionality

The website is divided into an index page, webstore, profile, community and login options. The functionality achieved on each section is described in the following subsections.

Entity-relational model

From the ER-model seen below, the entities with their attributes and their relationships are described.



ER-model of the product

Games

At the moment the website features four games. These games reside on their own pages containing their respective html and javascript. The games are displayed to the user through an iframe on the main game page of the website in order to maintain a consistent user interface and reduce repeating code. The game which is loaded into the iframe is determined through a hash link parsed from the URL. The games are a flappy bird clone, a vertical jumper game, a snake game and a reaction test clicker game. The flappy bird and jumper games take advantage of the javascript game engine library Phaser. Phaser simplifies writing games as it has built-in functions for things like loading sprites, collision handling, user inputs and physics calculations. The snake and reaction games are written in native javascript with some JQuery used to handle the menus and ajax requests.

All of the games have a shared css file that is used to keep the menus uniform and appealing throughout the games. There is also a shared script file with functions for changing the start button of the games to a random color, as well as a function to send the highscore to the backend controller and display the received response.

The backend of the games consists of functions to save the highscore of a game and retrieve the highscores. Those functions are in the highscore handler with is called through the controller. After a game is played, the highscore and name of the game are sent to the highscore handler, which loads the user's previous highscore from the database and compares the two values, replacing the old highscore if the new one is a greater. Regardless, the greater of the two values is sent back to the frontend and displayed to the user. The highscore handler is also used to generate an html table, or JSON object, of all the users highscores, based on POST parameters. The html table is displayed on the user's profile page.

Coins

All users on the website can earn coins to be used in the webstore. These coins are obtained by playing any of the games on the site and the amount received is based on the score achieved on the game. The highscore handler, which is responsible for updating highscores, is also used to handle coins as it receives data of the score after each game. Each game has a unique multiplier which is applied to the score. The resulting value is rounded up and added to the total coins of the user in the database.

On the frontend side, the amount of coins a user has is displayed in the navigation bar on the top of the page. This display is updated through an ajax request, after every game and every page load. It also features an animation if news coins were received.

Login

Logging in and registering to the website is handled by two buttons that appear in the navbar if the user is not logged in. Clicking on either of the buttons opens a popup modal with a form with input fields that depend on the actual button pushed.

Registering requires a unique username, the user's full name, email address and a password that is at least 6 characters long and must contain at least one lowercase letter, a capital letter and a number in order to be valid. The password then needs to be re-typed identically onto an additional field. The password is validated both on the client and the server side. All errors are shown on the login modal to inform the user on what went wrong. Once successfully registered the user is logged in which also changes the layout of the navbar the user sees. The profile and community tabs become visible, a coin counter appears and the "login"/"sign up" buttons are replaced by a "log out" button. Also the user's login is stored in a php $_SESSION variable on the server side. If the user doesn't specifically log out of the website the variable remains set and the user stays logged in even in the case of the connection severing between client and server. The session variable is used to identify the user in many situations as it contains the user's id number which can be used to retrieve the user's information from the server side database.

Webstore

The webstore consists of products than can be bought with coins. Without logging in, the user can browse through the products and see the product information, including their costs in products' pop-up windows. The user needs to be logged in to the website in order to buy the products. A product is bought by clicking the "buy" button under the product. When a product is bought, it is added to the user's collection of prizes and the cost is subtracted from the user's coins. If the user does not have enough coins to buy the product, the product cannot be bought. Buying also adds an order and an order row, including a date and time of the purchase, to the database.

The webstore also contains a button for adding products that is only visible for a user who is specified as an admin. The "add products" button opens a modal on which the product information can be filled in. Javascript function checks that all the required information is given and the information does not contain only whitespace. Finally, the product is added to the database table of products.

Profile

Each user has their own profile. A profile consists of public information such as username, personal high score, friends and optional description and location. Profiles are public to other logged in users. Users can edit their information by clicking "Edit profile" button on their own profile. This button opens a new modal with a prefilled form. The form is filled with the current user information. The user can change these values and save the new information. Clicking the save button, the new information will be sent to the server side with PHP and saved to database. Changing profile picture functions in local host environment, but in the Metropolia server it does not work at the moment.

Below the user information the profile page also provides links to the 8 last registered and 8 last logged in users. Every time a user registers a timestamp is

saved to the database and also within every log in another row in the database is timestamped.

Friends

A user profile also shows also how many friends a user has and who these friends are. The number of friends is retrieved with other public user information. When the avatar symbol with a count next to it is clicked on profile, all the friend's information including id, username and profile picture are retrieved and shown in a modal. Users can view their friend requests also in their own profiles. These are shown in the same friend modal in different tabs.

A registered in user can send a friend request to another registered user. When visiting the profile of another user, a button depending on the friend status will appear. It will either be "Add Friend", "Accept Request", "Cancel Request" or "Delete Friend".

A friendship status is mutual friendship when both have added each other. In this case there are two rows in the friendship table, one for each add. When only one has added, it's a pending friend request to the adder and a friend request to the receiver. In this case there is only one row in the database. The one who added is user1_id in the table and the receiver is user2_id.

Chat

The website features a community page, which is essentially a chatroom for users of the site and unregistered guests alike. The chat room itself is a separate page, which is loaded into the community page through an iframe, in order to maintain consistency of the website's appearance using minimal code. Each chat reply is displayed separately with the date and time of posting as well as the posters name. The page also has a textbox and submit button in order to post a reply to the chat.

When the page is loaded, a javascript function sends an ajax request to the backend controller, which forwards the request to the chat handler. The chat handler loads the chat history from the database and generates the html of the chat. The html is sent back to the frontend and displayed to the user. The html is generated on the backend because it was found to load much faster. When a user posts a new chat response, the html for that chat response is generated on the frontend and immediately displayed. Also, another ajax request is sent to the chat handler, which saves the response to the database along with a timestamp and the poster's username. Both the backend and frontend html generation scripts feature a function that changes the color of every other chat reply through CSS, to enhance the overall readability of the chat.

Product architecture

The product uses client-server architecture and is designed using the MVC model as a basis. Quite simply a central controller processes all requests from the client

side and calls on the appropriate server side models for functionality and then relays all returns back to the client. A MySQL database is used to store data on the server side.

Git

The project utilizes Git for version control and the project's source code is located in a GitLab (https://gitlab.com/sainip/PlayAndWin.git) remote repository.

Web page

The website is build with HTML. The site uses a ready Bootstrap template and its HTML and CSS has been customized for the site's needs. The functionality on the browser side is done with Javascript and JQuery. Some parts of the site are static but also many things like the webstore products and user profiles are mostly created dynamically from database using DOM.

Ajax is used to communicate with the server side. Javascript or JQuery sends Ajax to the PHP controller or directly to some PHP files. Ajax calls that come to the controller are forwarded to the right PHP files and functions and responses are echoed back to client side. The controller is also designed so that REST can be easily implemented later on. REST calls would be directed to the controller and from there forwarded to the right PHP file.

All the files are separated into different folders depending on their file type. A few exceptions are chat and games folders that include all the html, css and javascript files related to these functionalities. All the other files are structures so that HTML files right under the main PlayAndWin folder. Css is in the css folder and Javascript files are in the js folder. Images are under the images folder. Uploaded user profile pictures go to images/user.

Backend folder is separated in two subfolders, database and php. All the php files are under the php folder. The script for the mySQL database is under the database folder. PHPUnitTests folder has all the test files made for PHP and files related to tests.

MySQL and ORM

As mentioned a MySQL relational database is used to store the product's data. The database consists of 7 common tables in addition to a unique table for each game on the website and for each user a collection table which stores the specific user's purchase history. The common tables hold all the data that is needed to run the website. For example, the user table holds all user information like id, password, email address and the user's coins while the product table holds all the information, for instance the costs, of the webstore's products. On the server side a library called RedBeanPHP is used to provide ORM (Object Relational Mapping) and ease the use of CRUD (Create, Retrieve, Update and Delete) operations when manipulating data on the database. RedBeanPHP is an easy to use ORM solution

which requires zero configuring and automatically senses the database in use, populating it with the required tables if it does not find them.

Jenkins, PHPUnit

This project utilizes the testing framework PHPUnit and the Jenkins automation server, in order to automate the building and testing of the code used in the project. At the moment the project includes 22 PHPUnit tests, which test functions in the hshandler, chathandler, addProduct, buyProduct, connection and user php scripts. These assertion tests are performed calling the function, with parameters if they are required, and comparing the value returned to a given expected value. The test fails if there is an error is thrown or the return value differs from the expected value.

The Jenkins server monitors the Git repository of the project and automatically makes a new build and runs the tests if changes are pushed to the repository. Jenkins shows the test results as well as generates a Clover PHP Coverage report, which shows the amount of code that is being tested in relation to the total amount of php code present in the project, as well as the coverage of individual php files. At the moment of writing the total test coverage of this project is about fifty percent.

AgileFant

In this project, the product development is done using Scrum. A cloud service called Agilefant is used as a tool to organize the project tasks and sprint meetings. Agilefant is also used to keep track of the effort spent with this project. The project vision is divided into stories, that are further on divided into tasks. For each sprint, a set of stories/tasks are selected in the order of their priority. The project, at this point, consists of five sprints.