CS-C3170 - Web Software Development

Course Project - Online JavaScript Games Store

wsdProjectEIT group:

Relja Paunovic, 605078, relja.2.paunovic@aalto.fi
Sakshyam Panda, 605052, sakshyam.panda@aalto.fi
Sharbel Dahlan , 604626, sharbel.dahlan@aalto.fi

2017-03-05

Contents

1 | Overview

The objective of this project is to create an online JavaScript Games Store. Not only does it provide a chance to work on building a web software using the Django framework, but also it enables experiencing making design decisions, ranging from the high business-level to the low implementation-level.

Named "Awsome[1] JS Games Store", our web software is live at:

https://dry-george-eit.herokuapp.com/

Chapter 2 describes the features implemented in the project in line with the requirements given in the project description. It also includes non/functional details about each feature. Then, Chapter 3 shows how the tasks were divided among the team members. Finally, Chapter 4 provides a tutorial on how to use the application.

2 | Implemented Features

2.1 Minimum functional requirements

Users can register either as game developers or as players. This separation is done during the registration phase, and based on the user type the flow of the website is determined. A user can only play a game after purchasing it. As a player, users can buy games, see purchased games high scores, save their state of game. Along with these features, developers can add new games to their account, view sales for each game, edit/update their games. Developers who upload a game have an access to play it by default.

2.2 Authentication (*mandatory*, 100-200 points)

The authentication for a new registration is done through email. When a user registers, an activation link is sent to the email provided during the registration. The user can login to his/her account only after activating the account through the link. We have used googles gmail SMTP server to send emails. Email used is wsdawsomeproject@gmail.com. We expected to be challenged in the email verification part but it turned out to be straightforward.

[1] With the unintentionally misspelled "awesome" which became a trademark.

Expected Score: 200. Since we implemented registration with email validation, we expect to get the maximum credits for this feature.

2.3 Basic player functionalities (*mandatory*, 100-300 points)

Users can only play the games which they have bought. They can find the games by browsing through the available games, with the ability to browse games by different categories on the "Browse Games" page.

Expected Score:300

2.4 Basic developer functionalities (*mandatory*, 100-200 points)

All the functionalities are included, adding the game (with a price, description, image, url and name), modifying the game details, deleting the game and sales statistics (who bought the game and when). We paid attention to security as always, and tried to cover as much as possible. Only developers who uploaded the game can monitor it and they can only upload to their own inventory. Players have no access to this.

We used Cloudinary services for handling the images. We struggled to effectively use this services provided by Cloudinary. We might have spent a few days in understanding how the connection works and how can we access the images on their servers. To retrieve efficiently we are storing the links of the images on our database and using it to display on the templates.

Expected Score: 200

2.5 Game/service interaction (*mandatory*, 100-200 points)

Interaction with the game for highscores has been implemented using Ajax calls, type of message is recognised using messageType attribute of the message and then call was made to a view that returned a result for the event handler to execute.

Messages from service to the game have been implemented as well.

We think that this was badly designed for saving high scores because it does not give information on how these high scores should be ordered (for some games higher high score is actually worse than lower one, for example, when score is time), but it is not our fault so our expected score is still at maximum.

Expected Score: 200

2.6 Quality of Work (*mandatory*, 0-100 points)

We have used Django MVT purposefully, commented code just the right amount (only things that deviate from standard use of python or Django). We have paid a lot of attention on user experience, by following Nielsen's principles as much as we could (especially the feedback part) and we asked people not involved in the project for what deviates from intuition so that we could adapt it (including our mothers who are technologically impaired). We have written some automated tests in order to show our skills but not very extensively. Manual tests were mainly used.

Expected Score: 90 We expect less than maximum because some re-factoring is needed that we did not do.

2.7 Non-functional requirements (*mandatory*, 0-200 points)

Project plan (part of final grading, max. 50 points)

We worked on a holistic project plan, also including things that are good to have in case we had time to implement them. Since we covered all the required part, we deserve the full grade for that. For the extra features, we only focused on what was encouraged by the instructors, hence we disregarded the trials for games for example.

For the final documentation, we tried to do the most straightforward structure, similar to the project description, while following what the instructor suggested.

Expected Score: 200.

2.8 Save/load and resolution feature (*optional*, 0-100 points)

The service supports saving and loading for games with the simple message protocol described in Game Developer Information.

Users can save their current state while playing a game and can also resume from their saved state.

Expected Score: 100

2.9 3rd party login (*optional*, 0-100 points)

Third/party login was implemented with both GitHub and Facebook. This was done using Django OAuth. Since the system is programmed to redirect auomatically to the index page, which we were not using in the same way as intended, we faced some issues in the beginning, until we fixed it with creating a specific view. Then, implementing 3rd party login on the local host was straightforward, while upon deployment to Heroku, both did not work initially, raising "server error 500" for when trying to login with Facebook. The problem was that the Facebook app needs to be set to be used publicly.

In the end, the process works well, but there is still room for improvement (such as letting the user delete their 3rd party account from our system, which is not implemented now), hence that might affect our grade.

Expected Score: 90.

2.10 RESTful API (*optional*, 0-100 points)

We have a RESTful API through which user can access available games, high score and number of purchases. The API is only accessible by the developers, this was done in order to reduce unnecessary complexity, since most of the users only want to play games. The authentication is provided through a unique key to each developer. Versioning of the api has also been included, although, only one version currently exists. Information about game sales are open to every developer, not only the games they developed.

Documentation for the api is following:

1. searchGamesAPI/v1 with following arguments

   (a) key - Mandatory argument

   (b) q - Represents query and expects game name, if not provided all games will be listed

2. searchHighScoresAPI/v1 with following arguments

   (a) key - Mandatory argument

   (b) q - Represents query and expects game name, if not provided highscores for all games will be listed

3. searchGameSalesAPI/v1 with following arguments

   (a) key - Mandatory argument

   (b) q - Represents query and expects game name, if not provided sales for all games will be listed

Expected Score: 100 Even though we didn't make it available for players we expect maximum points because it is a design decision and we are convinced that business wise it is a better decision to allow it only for developers.

2.11 Own game (*optional*, 0-100 points)

We have developed a simple memory game.This type of game is chosen because it is simple, it demonstrates communication with the platform perfectly and most of the people are already acquainted with the rules of the game.

Rules of the game are following. User chooses two cards that are facing down, those two cards flip to show the face, if they are the same they remain face up, if not, they are turned face down again. Game is over when all the cards are facing up and the score is calculated as a time that user took to find all the duplicates. Game features eight different cards (with pictures of technologies used, except Node and Panda (which is not a technology but a cute animal)), thus, game is a grid with sixteen cards in total. Cards are randomized every time the game starts.

If user saves game, only the cards he/she found will be saved (and not their position), this is chosen to prevent cheating (user can save game at the beginning, take as much time as he needs to finish the game and then load to return to the beginning with the knowledge of the cards position), thus, loading game will turn cards he found face up but they will be on different positions (most probably).

This game implements all necessary features, thus, we expect the maximum score.

Expected Score: 100

2.12 Mobile Friendly (*optional*, 0-50 points)

Attention is paid to usability on both traditional computers and mobile devices (smart phones/tablets).We are using bootstrap to achieve responsive design. Our layout and design is consistent across devices of varying screen width and also is usable with touch devices.

Expected Score: 50

2.13 Social media sharing (*optional*, 0-50 points)

Unlike other features that seemed difficult to implement but turned out simple, social media sharing surprisingly did not work as it is expected. We managed to share on Facebook and on Twitter the URLs of the specific games that we want to share, so when the link is clicked, the user is redirected to the game page or to the login page if they are not logged in. Despite using meta tags to allow Facebook and Twitter to get the image of the game and a description, only the page name and the app name appeared on Facebook. We thought first that this is due to the caching that Facebook and Twitter crawlers use, which saves the link for a long time. However, upon trying with different paths, the problem still persisted.

Expected Score: 30. Sharing with Twitter is still more personalized than that of Facebook because we managed to pass the game name and the URL to the content of the tweet. For that, it can be argued that we did more than just sharing a simple URL. However, we could not get the Cards to work (with the image and the description), so we expect that it might cause deduction.

3 | Task Division

During our project, we followed the agile software development method roughly by using a Trello board as a scrum board tool, wherein we divide features into cards assigned to team members. Cards were picked by members depending on what interests them and what would maximize their learning. In other words, the members picked tasks that also challenged them, since this project provides a good learning opportunity to expand their knowledge.

A skeleton of a project was made by Relja using a template which was then stripped down to remove everything that was unnecessary. This was done because we were warned that deploying to Heroku could cause problems if you do not deploy early and test if everything was set up. The skeleton included main templates (with some bootstrap, jQuery and all the stylesheets and other libraries used) and views (and corresponding URLs) that sent mockup data in order to test if everything was connected properly. While most of this first part was carried out by Relja, Sakshyam was helped with finding bugs and some decision making.

Meanwhile, Sharbel was designing the models since designing a database was a crucial first step in software design. This task took a long time because it required a lot of insight into the structure of the project, especially that it depends on a lot of business logic that we have to agree on from the beginning. This logic includes whether developers have access to their own game, or whether or not they are able to change the URL of an uploaded game.

Table 3.1 shows how the tasks were divided. It is noteworthy that everybody was present in every part when debugging was needed, logic was not clear, and ideas were discussed.

TABLE 3.1: Summary of Task Division

Member	**Task**
Relja	Authentication, basic player and developer functionalities, game service interaction, developing own game, designing the API, setting up Heroku.
Sakshyam	User Interface Design, creating and designing the templates and corresponding views, cloudinary integration, setting up Heroku
Sharbel	Database design, writing the models, implementing external payment service, writing the APIs, social media login and sharing, view game sales

4 | How To Use The Application

Named "Awsome[1] JS Games Store", our web software is live at:

https://dry-george-eit.herokuapp.com/

One can register as developer or as regular player. This option can be selected during the registration process. To explore the potentiality of this platform, we recommend to create a developer account through registration page.

[1]With the unintentionally misspelled "awesome" which became a trademark.

After successful registration, an email containing the activation link will be sent to the registered email provided during registration. The account can be activated only through this link. Follow the link and explore the platform.

While logged in as a developer, you can also try the API by following the documentation in section 2.10.

5 | Conclusion

This project allowed us to get challenged and realize that things could get complex even in the simplest systems. We decided to treat this project as if we were in a startup and working on delivering version 1 of the system. This affected how we took some of the business decisions, which in turn affected our technical implementation. While we are aware that there might be flaws in the current version, such as submitting URLs that do not actually contain games, we made sure that the main required functionalities are met, at least with one way of implementing them. Since this is a learning process for the three of us, we highly value criticism and suggestions that we get for any aspect of this project.