CS-C3170 Web Software Development 2016

Project report

502841 Teo Mertanen
428022 Elis Viitanen
605311 Bingjing Xu

What features you implemented and how much points you would like to give to yourself from those?

Authentication - 200/200: Using Django auth and a linked custom model, we made an extensible and very robust authentication system with two kinds of accounts. Most templates dynamically reflect which kind of account is logged in. Pages are appropriately restricted. There is an email server that sends an email upon registration, though no action is required to confirm the account.

Basic player functionalities - 300/300: Buying is functional and payment errors are handled well. Playing games works, and is properly restricted to only users who have bought the game, or the creator of the game.

Basic developer functionalities - 175/200: Developers can add games, including descriptions and images which are shown on the front page and game page. They can also modify their own games and delete them.

Game/service interaction 200/200: Highscores and messaging work as described.

Quality of Work - 100/100: Code avoids repetition and aims for conciseness. Missing objects, etc. are handled appropriately instead of littering try-blocks. Concerns are separated well and all functionality has been tested as a player, developer and while not logged in.

Non-functional requirements 200/200: We still have not received any feedback about our Project Plan so that's a bit tough to estimate. Following the plan could have gone better, we did not meet quite as often as intended and most of the work got delayed. The git project was set up with feature branches, and commit messages are mostly informative.

Optional features

Save/load and resolution feature 20/100: The resolution of the game is dictated by the SETTINGS message from the game as described. Saving/loading is not implemented.

3rd party login 100/100: Google and Facebook login is supported.

Mobile Friendly 50/50: Implemented the Responsive Design

How did you divide the work between the team members - who did what?

Bingjing Xu:

1. Environment Setting: configure the django framework, set the initial heroku environment;
2. Authentication: login, logout, 3rd-party login and registration;
3. Payment Service: connect the payment service to the fake banking system provided in the class;
4. Front-End Design: All the front-end design including homepage and all the sub-pages, implement responsive design in every page;
5. Part of Heroku test work: upload the code from gitlab to heroku system and assist Teo to complete some testing work.

Teo Mertanen:

Git setup & advice, merging etc. Most database models, forms and views. Adding, editing and deleting games functionality. Saving and viewing highscores. Testing and readme documentation. Access restrictions and navigation between pages. Heroku deployment.

Elis Viitanen:
I definitely did not do as much as the others as I was really busy with dealing with new exchange students and Guild Board stuff early on in the project. I also had some big issues with using Windows, eventually giving up and switching to Mac which also took some time to get to work properly.

1. Setting up the URLs and playing the games
2. Messaging to and from the games

Where do you feel that you were successful and where you had most problems?

Creating a partially custom authentication system was quite challenging to start with. It is very hard to start with the Django built-in authentication system for a Django rookie. We couldn't

even know where to start. After a long-time search and the assistance through tutors, we finally find a clue how to solve this problem. And when customizing the user model, we really put a large amount of time on this. And in the end it worked very smoothly.

Also, the 3rd-party login isn't easy for us. We read a lot documentation trying to figure out how to configure Facebook and Google+ oauth2 plugin. Furthermore, we are so satisfied with the styling of the home page. It looks very modern and concise. The testing work is also very time-consuming. We meet different errors when uploading the code to Heroku, and it is very helpful to solve the problem with team members together.

However, there still some work unperfect. First of all, we couldn't meet a lot in case of individual timetable. Therefore, this leads to insufficient time management. Then, we couldn't finish most of the extra function.

The model-view-template flow of Django is quite simple to develop, and fast once it is learned. However, most of the difficulties during development were related to database migrations, problems with the git repository and using git, settings conflicts, the huge amount of trash files generated by Python/Django/virtualenv, and Heroku deployment difficulties.

Getting Javascript working initially without any real experience with it was quite a job with lots of trial and error but once it got going it went quite smooth. Messaging was the trickiest with that part. Other gameplay related parts went relatively smoothly.

Instructions on how to use the application

https://git.niksula.hut.fi/mertant1/wsd-project

https://wsd2016.herokuapp.com/

You are free to create accounts as either a developer (can play and create games) or player (can only play). Alternatively, use the ready-made developer account: dev/dev. To test as a player, you can use the account player/player.

The front page lists all games in the shop (and a non-functional search bar). Games are added by clicking on the "Developer. Click here to see your games" bar on the top of the front page. This page also shows the developer's created games. Links lead to each game's page with details, and if the logged in user is the creator of that game, they also see edit and delete forms. By clicking on the creator's name on a game page a public developer profile can be accessed. Players also have their own private page that lists the games they own. A link on each game's page allows buying or playing the game, depending on whether it is already owned.