Aalto University
School of Science
and Technology

Ye Olde Game Shoppe:

Submission Document

CSE-C3210 - Web Software Development

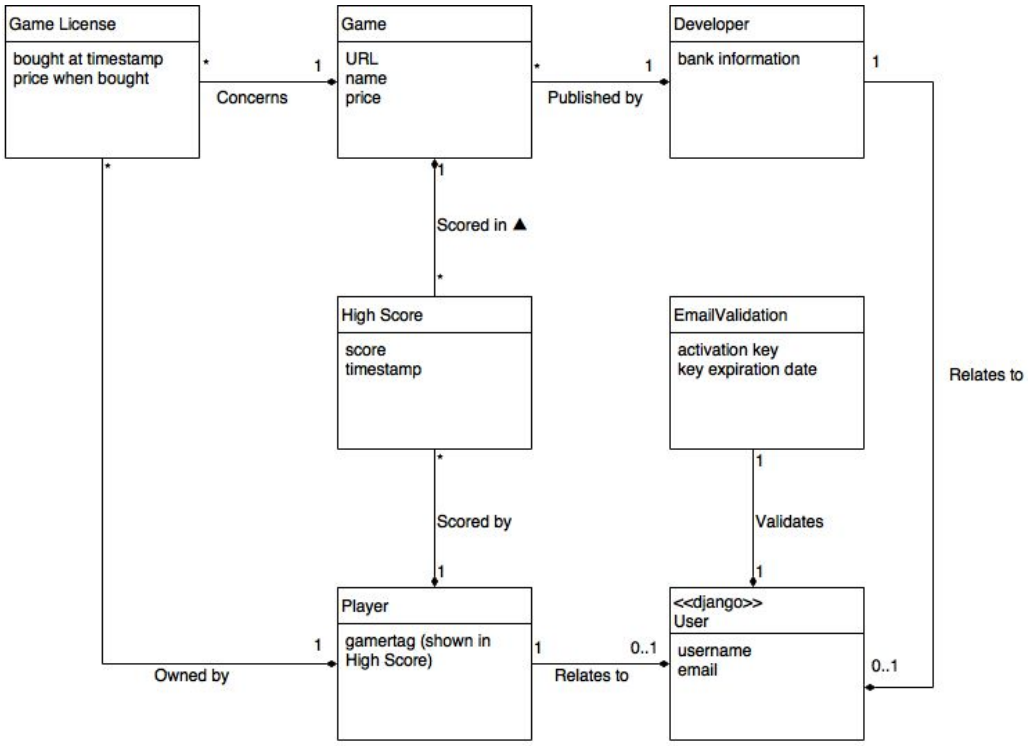
Topic: Online game store for JavaScript games
Group:

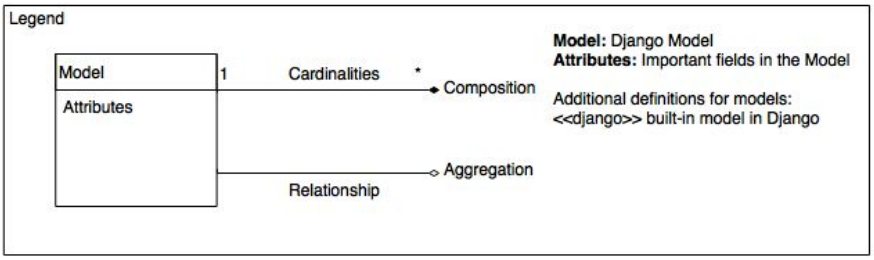
- Martin Yrjölä, 84086N
- Lakshika Perera, 547411
- Sai Manoj Katta Rokkaiah, 465111

Usage Instructions

1. Browse to https://murmuring-bayou-2830.herokuapp.com
2. Log in with the usernames player or developer. The user accounts have the same password as the username itself, so the ‘player’ username has the password ‘player’.
3. You can also register a new user by clicking on the corresponding link or by browsing to https://murmuring-bayou-2830.herokuapp.com/register

Django Models





Time schedule realization

- Week 0 (Until December 20th):
    - Meeting regarding discussion of project,plans and basic idea.
    - Project Report

Went as planned.

- Week 1 (December 21-27):
    - Authentication Module

Didn't do anything this week as it was Christmas Week. It was done in the next week.

- Week 2(28-3):
    - Basic player functionalities
    - Basic developer functionalities

Few parts of the above two modules were done.

- Week 3 (4-10) :
    - Game/service interaction

Some parts were covered.

- Week 4(11-17):Additional modules
    - 3rd party login
    - Social media sharing

Nothing much was done.

- Week 5(18-24):
    - Documentation.

We got the initial comments on the project as per our plan. Our deadline was extended for another two weeks. We got a bit lazy.

- Week 6 (25-31):Additional modules.
    - Save/load and resolution feature
    - Own game
    - Mobile Friendly

Continued the work of Week-2,3,4

- Week 7 (1-7): Wrapping up and Finalization. Taking appointment.

Working on the Week-6 target

- Week 8(8-14):Presentation and submission of document.

Finished the basic modules, made a demo-ready project

Further, in the coming weeks, missing parts were added. Documentation is done.

Feature Implementation

Mandatory Requirements

Authentication (mandatory, 100-200 points): 200 points

- Login, logout and register (both as player or developer).
- Email validation
- Use Django auth

We have implemented everything above. We chose to design user accounts so that a registered user can enable both a player account and a developer account in the profile page.

Basic player functionalities (mandatory, 100-300 points): 300 points

- Buy games, payment is handled by a mockup payment service
- Play games
- Security restrictions, e.g. player is only allowed to play the games they've purchased
- Also consider how your players will find games

We have implemented everything above. The players can find games through a search field in the all games view.

Basic developer functionalities (mandatory 100-200 points): 150 points

- Add a game (URL) and set price for that game and manage that game (remove, modify)
- Basic game inventory and sales statistics (how many of the developers' games have been bought and when)
- Security restrictions, e.g. developers are only allowed to modify/add/etc. their own games, developer can only add games to their own inventory, etc.

We have implemented adding of a game to the developer's inventory, we didn't implement modification and removal of the added games. The developer can view the sales statistics in the inventory view.

Game/service interaction (mandatory 100-200 points): 200 points

- When player has finished playing a game (or presses submit score), the game sends a postMessage to the parent window containing the current score. This score must be recorded to the player's scores and to the global high score list for that game
- Messages from service to the game

We have implemented the ERROR, SCORE and SETTING messages. The setting message only alters the size of the iframe. The high score list is shown while playing the game and it is updated dynamically when a SCORE message is sent.

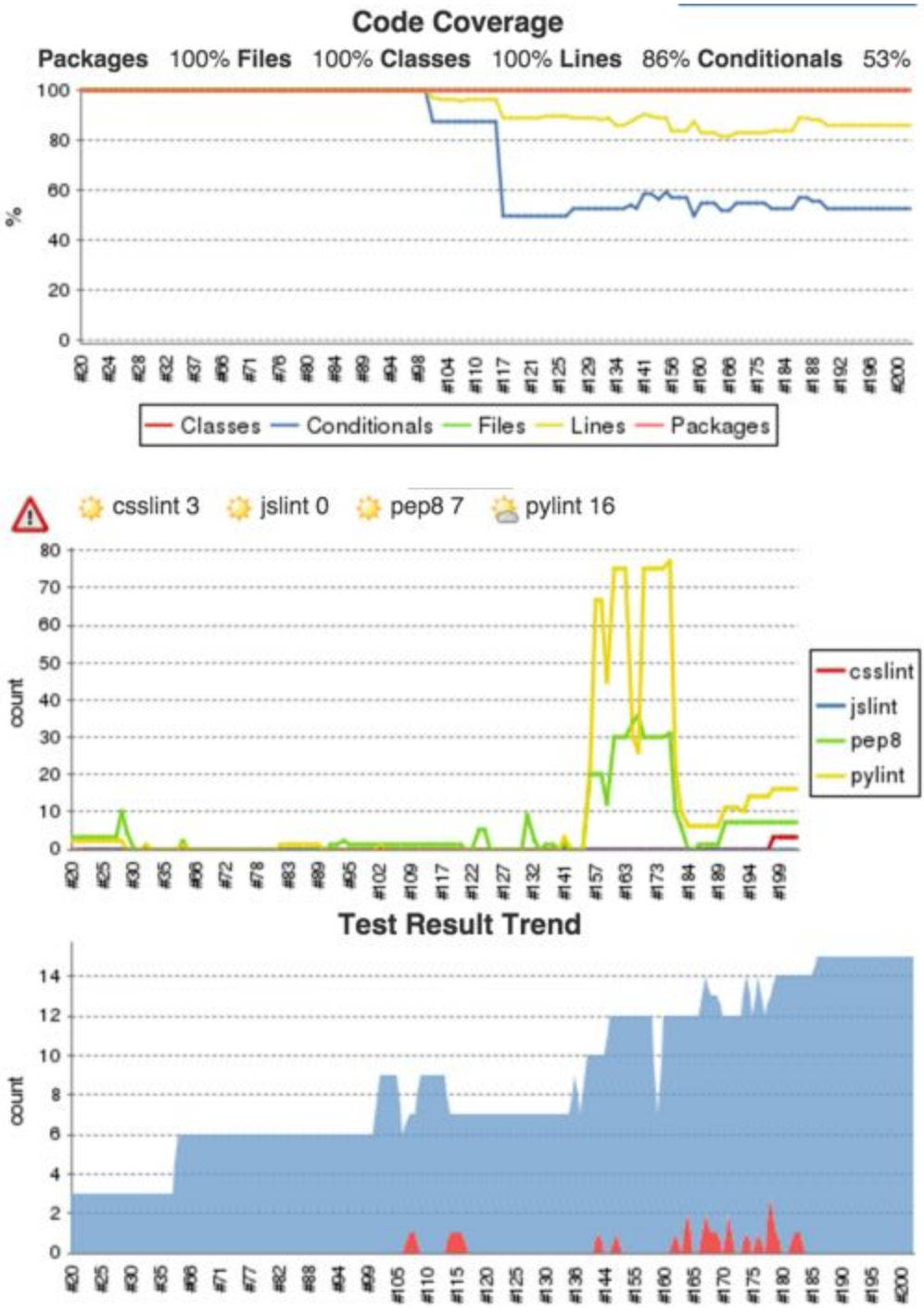
Quality of Work (mandatory 0-100 points) 80 points

- Quality of code (structure of the application, comments)
- Purposeful use of framework (Don't-Repeat-Yourself principle, Model-View-Template separation of concerns)
- User experience (styling, interaction)
- Meaningful testing

We were using Jenkins as a build service. You can contact martin.yrjola@aalto.fi if you need an user account to investigate our build further. We were mostly following the test results and violations, but also the code coverage can give insights into the quality of our testing. We have implemented both unit tests and Selenium browser tests. We use the django-jenkins library to integrate our web service with Jenkins.

We make heavy use of Bootstrap to provide a nice user experience. We use the django-bootstrap3 library to more easily use bootstrap in templates.

We strived to minimize code duplication and document the code sensibly. We weren't that familiar with Django from before, but we hope we haven't done any big blunders regarding the usage of the framework. We have reduced our points because all features aren't tested and there are some violations reported by pep8, pylint and csslint.

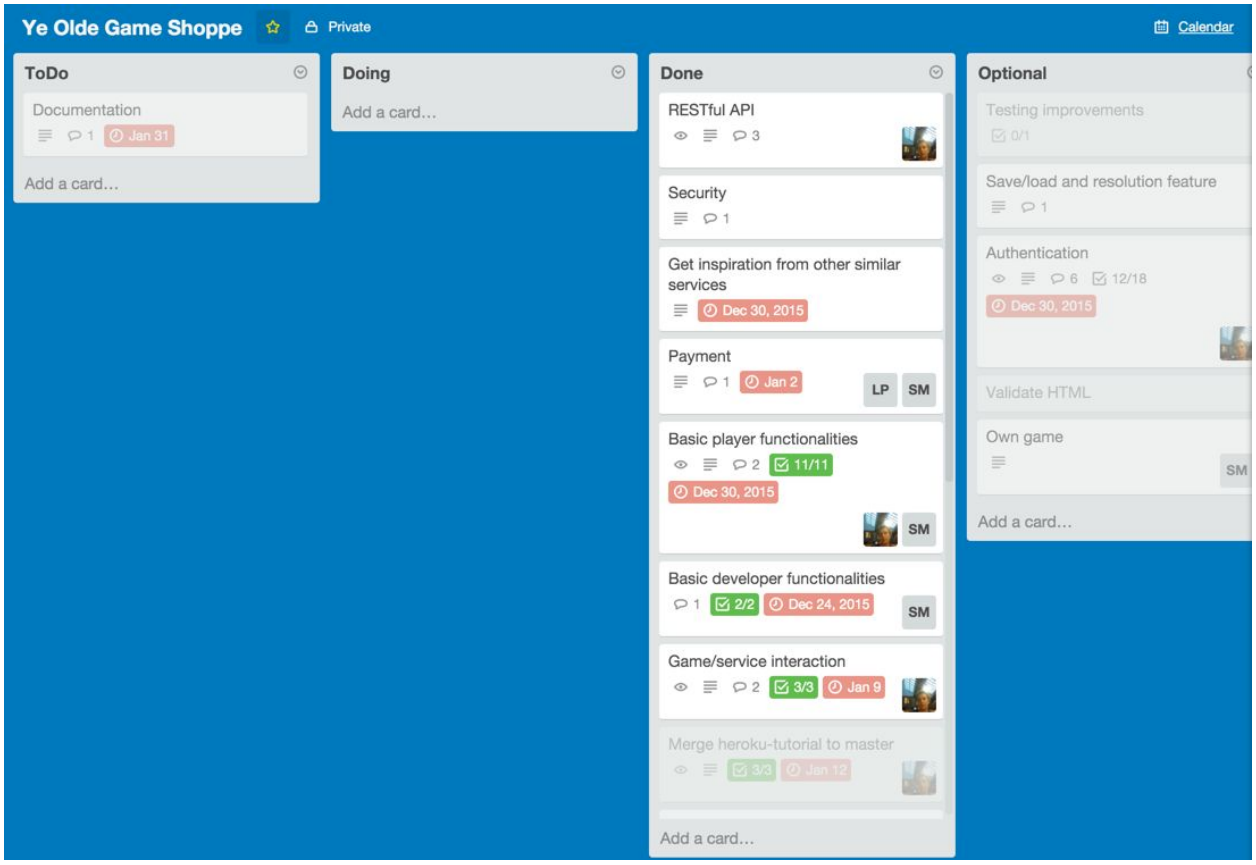


Jenkins graphs as of 2.3.2015. Code coverage, Violations and Test results.

Non-functional requirements (mandatory 0-200 points) 200 points

- Project plan (part of final grading, max. 50 points)
- Overall documentation, demo, teamwork, and project management as seen from the history of your GitLab project (and possible other sources that you submit in your final report)

Otto said that our project plan was very detailed, so we hope for full points from that. The overall documentation should be in good shape and we managed the project effectively using Trello. We used Facebook and Skype for communication. Martin is an experienced developer, so he worked much more effectively than the other group members, but the time usage should be quite balanced. We had two hours of face to face meetings every week and organized one day hackathon during the winter months. We made use of pair programming for the more challenging features like payment and when a group member needed guidance regarding Django.



Screenshot from Trello at the time of submission.

More Features

Save/load and resolution feature (0-100 points): 33 points

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

We only implemented the resolution feature.

3rd party login (0-100 points): 100 points

- Allow OpenID, Gmail or Facebook login to your system. This is the only feature where you are supposed to use third party Django apps in your service

We have implemented Facebook login, we aren't using email validation when creating an account through Facebook.

RESTful API (0-100 points): 75 points

- Design and Implement some RESTful API to the service

We implemented querying of the top 10 high scores of a game. It is used by the service to implement an AJAX call to update the high score list dynamically.

Own game (0-100 points) 100 points

Develop a simple game in JavaScript that communicates with the service (at least high score, save, load)

A tic-tac-toe game was developed by Martin. The code can be found in the static folder in the yogsgame application.

Mobile Friendly (0-50 points) 50 points

- Attention is paid to usability on both traditional computers and mobile devices (smart phones/tablets)
- It works with devices with varying screen width and is usable with touch based devices (see e.g. http://en.wikipedia.org/wiki/Responsive_web_design)

We deliberately made the service single column to make it easier to use on mobile devices. We make fit the content to the viewport width on mobile devices with the <meta> tag. Bootstrap also makes the Navbar responsive.

Social media sharing (0-50 points) 30 points

- Enable sharing games in some social media site: via Facebook
- Focus on the metadata, so that the shared game is "advertised" well (e.g. instead of just containing a link to the service, the shared items should have a sensible description and an image)

Game pages includes Facebook like and share buttons. No sensible metadata is shared.