Design Document for “exgames stock”

Riccardo Graziosi | rg505@exeter.ac.uk | 690011148

Website available at: www.rg505.altervista.org

To login use username: rg505, password: ECM1417pass!

Source code available at: www.rg505.altervista.org/source.zip

Introduction and high-level requirements

The system must provide the functionalities needed to manage the stock of a videogame company. The application must be accessed only by administrators or employees of the company, so an authentication system is required. Since the staff will use the web app on different devices with different screen sizes, it must be able to work properly on every device and screen size. Additionally, as the staff usually works in areas with limited internet connection, the application should make use of client-side processing and AJAX calls to reduce the number of times the application needs to reload the entire page.

The system must use the following technologies:

- HTML and CSS for the frontend.
- JavaScript for client-side processing and AJAX calls.
- PHP for the backend.
- Database. In particular, the database should use a single table to store the stock information.

Low-level requirements

Given the high-level requirements of the company, this section tries to complement them to achieve an unambiguous requirements specification. Some design choices are also made in this section.

Functional requirements

Authentication

Two different types of users will use the application: administrators and employees. Each type will have different privileges inside the application:

- Administrators must be able to manage products as well as users. They are the ones that create accounts for the employees.
- Employees must only be able to manage products. They should not be able to manage other users.

Since there are different types of users, the database table that contains the users should have a column to distinguish between them. This field should be a number, allowing for an arbitrary number of roles and not just two. In this way it will be easy to add new roles when it is needed.

Access to the system must be secured through authentication: no page, except the login page, must be shown to a non-logged user. Moreover, there is no registration page because no external user should be able to register to the system. Instead, administrators must create accounts for their employees. Then, they should notify the employees that the account has been created, along with the username and password to access it.

Product management

The products are the videogames the company sells. The system must provide functionalities to:

- Add a product
- View a product
- Edit a product
- Delete a product

Both employees and administrators should be able to manage products.

The system must provide validation for the data of products being submitted, so no invalid data is saved to the database.

A product should contain the following data:

- Code: a unique identifier used to distinguish a product from all other products.
- Name: the unique name of the product.
- Platform: the platform for which the product is available.
- Price: the price of the product.
- Quantity: the quantity of the product in stock.
- Description: a description of the product. This field is optional.
- Developer(s): the list of developers that created the product. This field is optional.
- Publisher(s): the list of publishers that sponsored the product. This field is optional.
- Image: an image of the cover of the product. This field is optional.

It should be easy to filter and sort products based on their attributes. Additionally, most of the product management tasks should be carried out with the minimum data usage possible, to enable the staff to manage the stock even when Internet connectivity is poor.

User management

The system must provide functionalities to:

- Add a user
- View a user
- Edit a user
- Delete a user

Only administrators should be able to manage users.

The system must provide validation for the data of users being submitted, so no invalid data is saved to the database.

A user should contain the following data:

- Code: a unique identifier used to distinguish a user from all other users.
- Username: the unique username of the user.
- First name: the first name of the user.
- Last name: the last name of the user.
- Password: the password of the user.

Non-functional requirements

Mobile-friendliness

The web application should work properly on different screen sizes and devices.

High performance

The web application should offer an acceptable speed given low Internet connectivity. This goal must be achieved, as stated in the high-level requirements, through AJAX requests to the backend service. To improve performance, the PHP backend service must process requests as fast as possible.

Security

The database contains important information for the business, so no unauthorized people should be able to access the web application. Additionally, the people authorized to access the system have different roles and the system must be able to distinguish between them.

Technologies used

Given the technological constraints imposed by the high-level requirements and the functionalities of the low-level requirements, the following section describes which technologies the system will use.

HTML

Since the system is a web application that works in a browser, HTML is the obvious choice. It was decided not to use variations of HTML such as XHTML because HTML is more future-proof and widely used.

CSS

To improve the user experience the system makes use of CSS. In particular, CSS is used to achieve mobile-friendliness and good looks.

However, writing CSS is an iterative process that can take a lot of time. Since this application does not require special style features, it was decided to use Bootstrap to speed up the process. Bootstrap brings several advantages, such as:

- Responsive and mobile-friendly design.
- A clean and traditional style.
- Speed of development.

Every effort has been made to make the website mobile-friendly: each component of the pages resizes and changes accordingly to screen size and every frontend feature has been proven to work on the most widely used devices. However, as the application needs to show large amounts of data, it is best suited to be used on a computer, tablet or a mobile phone in landscape mode. It is not recommended to use it on mobile phones in portrait mode because tabular data inevitably takes up a lot of horizontal space, even though everything works as expected.

JavaScript

JavaScript is used to give interactivity to the frontend and to enable AJAX requests. These last are used as much as possible throughout the application to make the web application more responsive and to make it possible to use it even when Internet connectivity is poor. In particular, AJAX requests are used to add, edit and remove a user or product to/from the database, which are the most commonly used functionalities of the application.

JavaScript is also the technology that makes searching, filtering and sorting possible.

Other secondary features are powered by JavaScript, such as the drop-down menu shown on small devices.

PHP

The server-side language in which the application is written is PHP. Since it can be difficult to develop a PHP application from scratch, it was decided to employ a PHP web framework. A framework is useful because it provides a standard structure for the system and common functionalities are already implemented.

There are a lot of options available when it comes to PHP frameworks, but for this particular project CodeIgniter seemed the best choice.

CodeIgniter

CodeIgniter suits this project superbly for a number of reasons:

- It is fast. Since the system will be used for work, the faster it processes employee requests the more productive they will be.
- It is light weight at its core, but nevertheless extensible. As stated below, two third-party libraries are used to implement important features of the web application.
- It is well documented and has a vast community of developers, which means it is easier to find solutions for problems when they occur.

For this particular web application, two CodeIgniter third-party libraries are particularly useful:

- Community Auth. Community Auth is an authentication library for CodeIgniter. It offers all the common authentication features and much more. It was decided to rely on an external library to handle authentication because it is a tricky part to implement. Moreover, a single error could mean a security flaw in the application. On the other hand, Community Auth is an open-source, long-lived and popular authentication library, which means that it is less likely to have security bugs and if one is found it is fixed quickly.
- Grocery CRUD. It is an open-source library for CodeIgniter that speeds up the development of management systems with CRUD (Create Read Update Delete) features. It is very convenient for this application because it automatically creates the interface and logic to create, view, update and delete products and users. Additionally, Grocery CRUD makes heavy usage of AJAX requests. Examples are creation, modification and deletion of a resource, upload of a file and more. Grocery CRUD provides other useful built-in features such as searching, sorting and filtering resources in real-time through JavaScript.

MySQL

It was decided to use MySQL as the DBMS for various reasons, the more important being its popularity and reliability.

It is important to note that the database schema is not normalized because the high-level requirements stated that product information must be stored on a single database table. For this reason, there will be some level of redundancy in the data stored in the database.

Security concerns

Password strength

The system forces the password to be at least eight characters long and to include a lower-case letter, an upper-case letter and a non-alphanumeric character.

Password storage

Community Auth hashes and salts passwords before they are stored:

- Hashing assures that even if a hacker gains access to our database, it cannot steal user passwords.
- Salting mitigates rainbow table attacks.

Man-in-the-middle-attacks

Communication between clients and server occurs over HTTPS (HTTP over SSL). This contrivance mitigates man-in-the-middle-attacks.

SQL injections

CodeIgniter communicates to the database using a library called Active Record, which escapes queries automatically to avoid SQL injections.

Cross-site scripting (XSS)

Every possible input has to pass through Grocery CRUD, which prevents XSS by default. However, as the web application is supposed to be accessed only by the company staff, switching the website to non XSS-protected has been made easy (for example, it can be useful to have HTML formatting in some fields such as the description of a product). This change can be done, by the developer, by editing just one line of code.

Error logs

Error logs are displayed only in the development environment and not in production. In this way, a malicious user cannot get any information about the internals of the system.

Backups

Since the application makes heavy use of the database, a backup system that runs often is essential.