**CoquiGames**

**Jesmarie Hernandez, Heriberto Bourdon, Abisai Ramos**

**{jesmarie.hernandez, heriberton.bourdon1, abisai.ramos}upr.edu**

**1.     Introduction**

There is both a lack of competition in the video game store market, and in the options for customers to purchase their favorite video games and accessories, particularly in the island of Puerto Rico. In the island, there is an absence of stores whose primary selling point is video games. Therefore, the gaming community in Puerto Rico are forced to choose between buying their video games online or in the island’s chain stores, which don’t offer a variety in games and prices. To solve this problem, we propose a video game web store, **CoquiGames**. **CoquiGames** is a web store that offers a viable solution to the gaming community in Puerto Rico by offering weekly deals, variety in titles available, optional payment methods etc.

**CoquiGames** is implemented as a responsive web application; it is designed to be viewed appropriately as desktop and as a mobile website. As a responsive web application, clients who either own a phone or a computer will be able to access it without the need of downloading and installing an application.

**CoquiGames** will be implemented using the following technologies

Client Side Tools: AngularJS, HTML/CSS.

Server Side Tools: Flask, PostgreSQL.

In this phase of the project, the connection to the database was established. As stated, the database used was PostgreSQL. Additionally, views for an administrator type of user were established.

**2.     Client App Description**

**CoquiGames** was implemented using HTML5, CSS, JavaScript, Bootstrap, and AngularJS. This application has two kinds of users the regular users and the administrator user. The regular users are persons that can browse the website for merchandise, view a specific product’s details,buy products, add them to a wishlist, store their personal information, etc. Whereas an administrator users are the IT personnel in charge of supporting and maintaining the database.

A regular user can perform the following tasks: register, login, view, and modify their account and personal information, shipping and billing addresses, and the payment methods. The account information consists of: a unique username, and a password. Whereas personal information will have the customer’s full name, email, date of birth and phone number. A customer without an account (guest user) can only browse and view items, and register to become an account user and be granted buying privileges.

Users will be able to browse the following categories of products within their specific platforms: video games, consoles, and accessories. Additionally, products will be able to be browsed through genre tags, a means to search for a product with keywords, and an option to allow for products to be filtered to appear in increasing or decreasing order of price.

Users can view the details of a product. Details will consist of the product’s name, genre, release date, product code, availability, instant price, description, photo of the product, and ratings. Additionally, there will be means to place, view and update the shopping cart. Once an order has been placed, the customer will get an invoice of the order that has been placed by email. Also, the web app will keep the invoice in the records for future reference. Order status information will be shown in the following format: open, closed, in transit, cancelled, pre-ordered. Finally, to review the products we will use a 5-star rating system with an option to leave a comment.

**3.     Server side description**

The store data will be obtained from a Flask server using REST calls and encoding data with JSON. The data will be stored in PostgreSQL (object-relational database system). The data objects are maintained and access in terms of four operations, often described with the acronym CRUD (Create, Read, Update, Delete). For the security and integrity of the store data the functionality will be limited to the user role.

A regular user will be able to perform the following tasks in the server:

* Authenticate a customer account.
* Create\Read\Update a customer account.
* Read products by category.
* Read the details of a product.
* Read\Update a shopping cart.
* Create an order for items in the shopping cart.
* Create\Read\Email an invoice of an order that has been placed.
* Read a Review of a product, only logged user will Create\Update\Delete a review.

An administrator of the **CoquiGames** will be able to:

* Authenticate an administrator account.
* Create\Read\Update\Delete customer and administrator accounts.
* Read sales data.

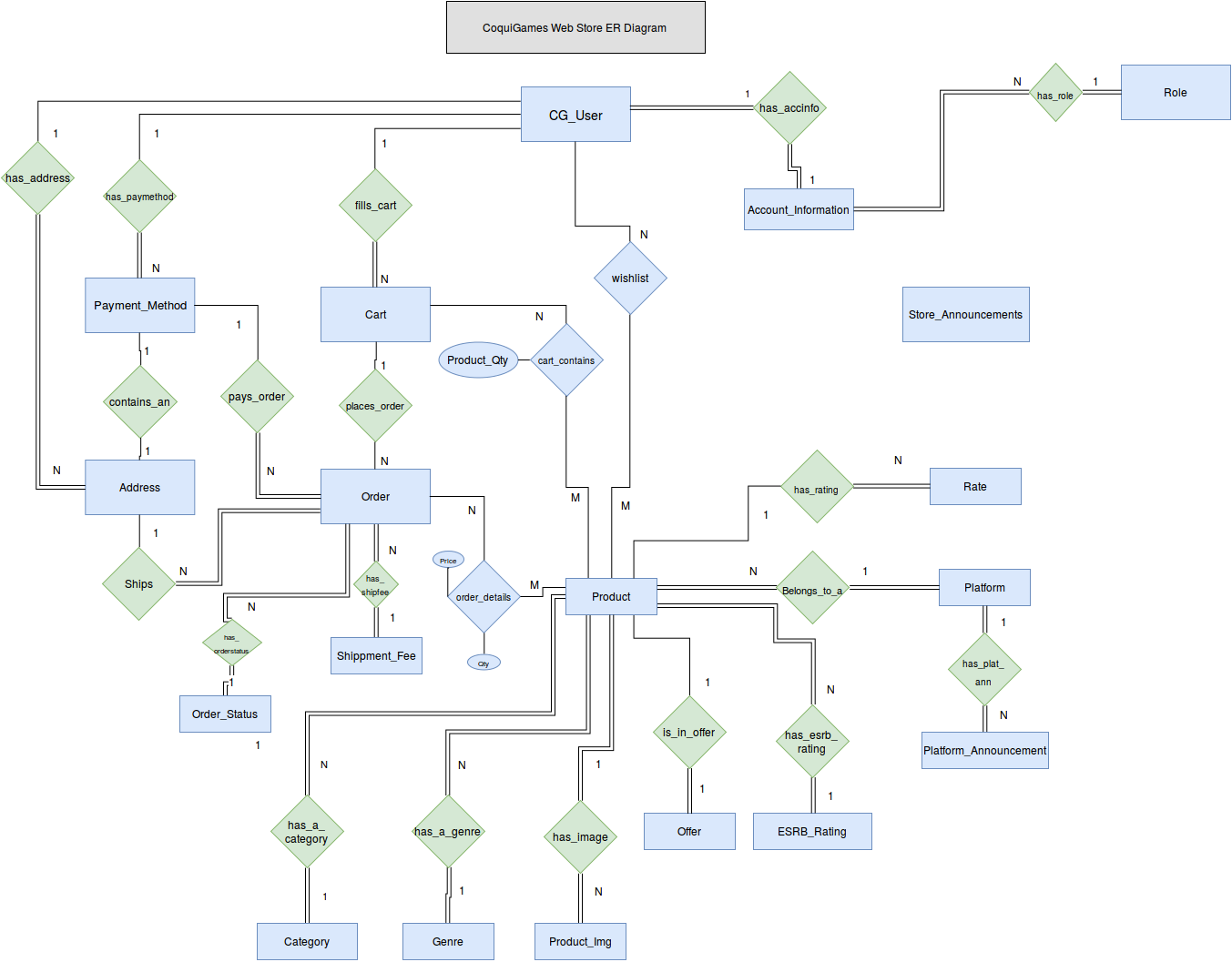
The signup process will be handled by way of a script configured for the application, which connects to the application server, handles the verification process and creates a session in the application server (Flask Server). User sensitive information such as credit numbers must be encrypted, for safety and protection of all the clients. Our front-end (which will be AngularJS) is going to request the back-end (Flask) for an encryption key. The front end will then encrypt the credit card information and then return it to the back end. Finally, the back end will decrypt information and the it will be stored in the database, which will once again encrypt the information.

**Tables:**

The tables created in our database are explained in detail in the ER Table Diagram in part five of this report.

|  |  |
| --- | --- |
| ***Table*** | ***Description*** |
| account\_info | Holds the information to access and account. |
| address | Keeps all the corresponding fields for an address. |
| card\_type | Holds what kind of credit cards the website accepts |
| cart | Keeps the selected products by a customer. |
| cart\_contains | Keeps information on what kind of products the cart holds |
| category | Holds the what kind of category are the products, when they were added, and what quantity. |
| cg\_user: | Stores the user personal information. |
| esrb\_rating | Holds the seven types of ratings by ESRB. |
| genre | Holds all the possible genres that can be applied to products |
| offer | Keeps information of what offer is available, to which product, and if its active. |
| order\_details | Stores the products in an order their quantity and price, what address it was shipped and billed to, what payment was used, etc. |
| order\_status | Holds order status information such as open, cancelled, in transit etc. |
| orders | Retains the orders made on the website. |
| payment\_method | Stores the information of what payment was used by each user/ |
| plaform | Stores additional information regarding a platform |
| platform\_announcements | Keeps the |
| product | Holds additional information regarding a game |
| product\_img | Stores the photo name and location in the server. |
| rate | Retains user reviews. |
| role | Has the account role and if its active. |
| shipment\_fee | Keeps all the possible shipment fees to offer and if they are active. |
| store\_announcement | Keeps the different announcements to be displayed in the homepage. |
| user\_preferences | Holds a user’s preference for shipping and billing address and also payment method. |
| wishlist\_contains | Has the products a user wishes to buy in the near future. |

**4.     E-R Diagram**



This E-R Diagram represents the **CoquiGames** Webstore. It has been updated to be more detailed since the last phase of the project. Our **CoquiGames** system contains eighteen entities and twenty-two relationships. As some of the cardinalities of the relationships are M x N, these results in more tables being created in our database. At the head of our **CoquiGames** ER diagram, it’s the *cg\_user* entity. This entity contains the user’s personal information, and it is related to other entities by different relationships. We will list them and their explanation as follows:

* *Cg\_user* relates to the *Address* entity by means of the *has\_address* relationship with a cardinality of 1 x N. This means one usercan have N number of addresses to his account. Whether billing or shipping addresses.
* *Address* entity is also related to *Order* by means of the *ships* relationship with a cardinality of 1 x N. This means one address can be tied to N orders. One order can only be shipped to one address, but various orders can have different addresses to be shipped to.
* *Cg\_user* relates to the *Payment\_Method* entity by means of the *has\_paymethod* relationship with cardinality of 1 x N. his means one usercan have N number of payment methods (credit cards) to his account.
* *Payment\_Method* is also related to the Order entity by means of the *pays\_order* relationship with a cardinality of 1 x N. This means one of the user’s payment method can be tied to many N orders. An order can only be paid with one payment method, but N orders can have different payment methods.
* *Cg\_user* relates to the *Cart* entity by means of the *fills\_cart* relationship with a cardinality of 1 x N. This means one usercan have N number of carts to his account.
* *Cart* entity is also related to *Order* entity by means of the places\_order relationship which has a cardinality of 1 x N. This means that the cart can place N number of orders.
* *Cart* entity is also related to *Product* entity by means of the *cart\_contains* relationship. With a cardinality of N x M. This means that N of the user’s cart can contain M number of products.
* *Cg\_user* relates to the *Product* entity by means of the *wishlist* relationship with a cardinality of N x M. This means N users can have M number of products in their wishlist. Additionally, the *cart\_contains* relationship has a parameter called *Product\_Qty*.
* *Cg\_user* relates to the *Account\_Information* by means of the *has\_accinfo* relationship with a cardinality of 1 x 1. This means one user can be tied to just one set of account information. In other words, only one name, one email address, one date of birth, etc.

Other entities that have a relationship with more than two entities are Order and Product. For Order, they are related to the following:

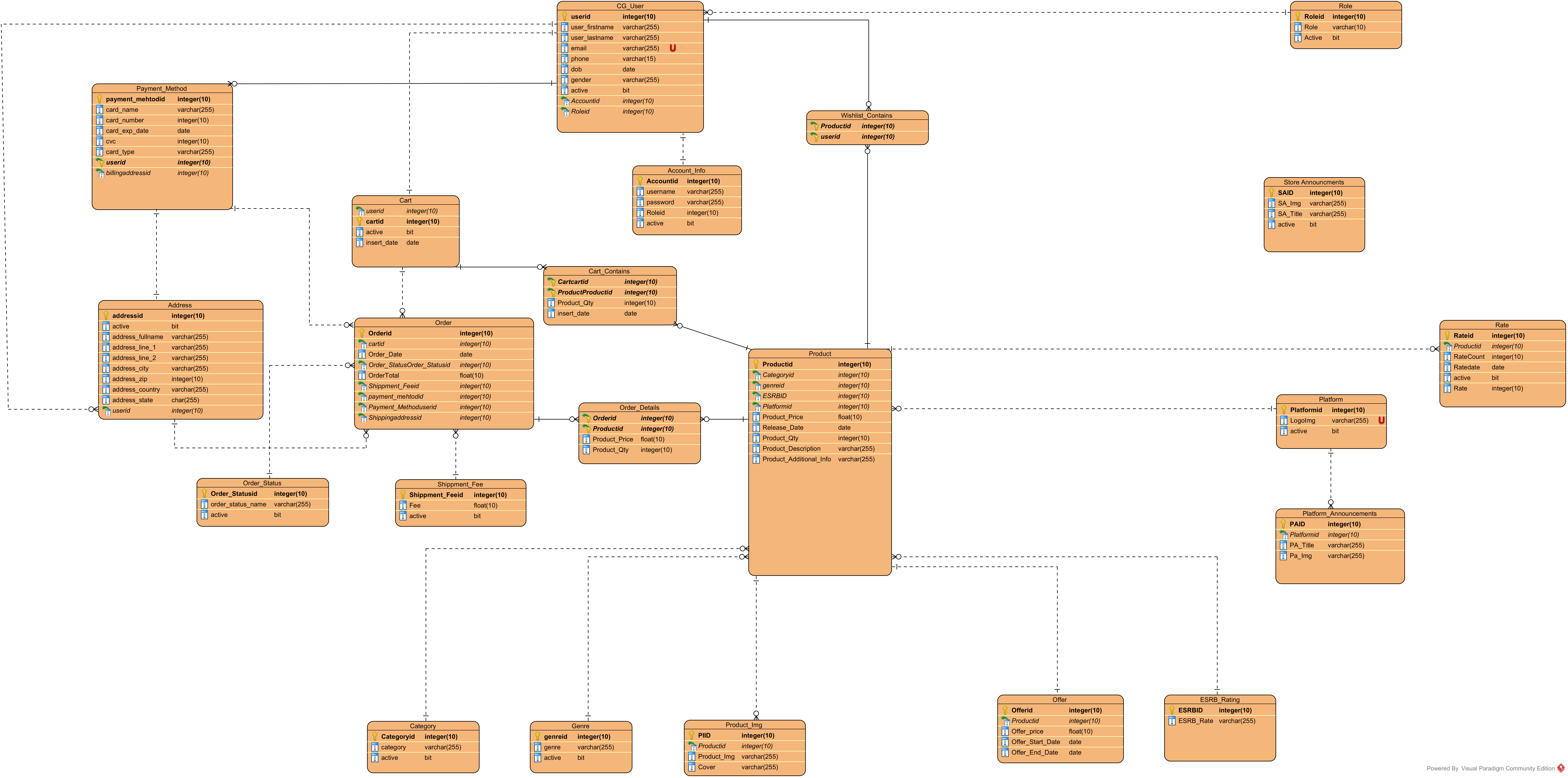
* *Order* relates to the *Order\_Status* entity by means of the *has\_orderstatus* relationship with a cardinality of N x 1. This means that there can be N orders, but each order will have a unique status. One order cannot be shipped and ready to ship at the same time.
* *Order* relates to the *Shippment\_Fee* entity by means of the *has\_shipfee* relationship with a cardinality of N x 1. This means N orders can have 1 type of shipment fee. Each order can be only adjudicated one shipment fee, so each N orders can only be charged one type of shipment fee.
* *Order* relates to the *Product* entity by means of the *order\_details* relationship with a cardinality N x M. This means N number of orders can have M number of products. Additionally, the *order\_details* relationship contains two parameters, *price* and *qty* (quantity of product).
* Order also relates to the *Address* and *Cart* entities, which we have specified above.

For the Product entity, it is related to the following entities:

* *Product* relates to the *Category* entity by means of the *has\_a\_category* relationship with a cardinality of N x 1. This means that N numbers of products can only be placed under one category. For instance, the Nintendo 3DS system can only be a platform. It cannot be a game or accessory.
* *Product* relates to the *Genre* entity by means of the *has\_a\_genre* relationship with a cardinality of N x 1. This means that N number of products can only have one genre. A video game such as Pokemon, for example, can only be classified as RPG. Not RPG and Action, or any other additional genre.
* *Product* relates to the *Produc\_Img* entity by means of the *has\_image* relationship with a cardinality of 1 x N. This means that one specific product can have many images attached to it.
* *Product* relates to the *Offer* entity by means of the *is\_in\_offer* relationship with a cardinality of 1 x 1. This means that one product can be in one offer. There cannot be two specials at the same time for one product.
* *Product* relates to the *ESRB\_Rating* entity by means of the *has\_esrb\_rating* relationship with a cardinality of N x 1. This means that N number of products can have one type of ESRB rating. No game can be E (everyone) and M (mature) at the same time.
* *Product* relates to the *Platform* entity by means of the *belongs\_to\_a relationship* with a cardinality of N x 1. This means that N number of products can be related to one type of platform. This is not to say, that games can be released under different platforms, for example. It means that a game for the WiiU, as an example, can only be played in that specific platform.
* *Platform* also relates to the *Platform\_Announcement* entity by means of the *has\_plat\_ann* relationship with a cardinality of 1 x N. This means that one platform can have N number of announcements in their particular platform homepage.
* *Product* relates to the *Rate* entity by means of the *has\_rating* relationship with a cardinality of 1 x N. This means that each product can have N number of ratings.
* Product also relates to the *CG\_User* and *Order* entities, which were mentioned previously.

An additional entity which is not related to any entity in our ER Diagram is the Store\_Announcements entity.

**5.     E-R Table Diagram**

****

This ER table diagram shows the parameters and the type of the parameter for each entity. It also portrays how each entity relates to the other entities of our CoquiGames system.

**6. Coqui Games Database Schema**

**CREATE TABLE account\_info**

(

**accountid INTEGER PRIMARY KEY NOT NULL**,

**username VARCHAR**(25) **NOT NULL**,

**upassword VARCHAR**(255) **NOT NULL**,

**roleid INTEGER NOT NULL**,

**active BOOLEAN NOT NULL**,

**CONSTRAINT** account\_info\_role\_roleid\_fk **FOREIGN KEY** (roleid) **REFERENCES role** (roleid)

);

**CREATE UNIQUE INDEX** "Account\_Info\_username\_uindex" **ON** account\_info (username);

**CREATE INDEX** account\_info\_active\_index **ON** account\_info (active);

**CREATE TABLE** address

(

addressid **INTEGER PRIMARY KEY NOT NULL**,

active **BOOLEAN NOT NULL**,

address\_fullname **VARCHAR**(255) **NOT NULL**,

address\_line\_1 **VARCHAR**(255) **NOT NULL**,

address\_line\_2 **VARCHAR**(255) **NOT NULL**,

address\_city **VARCHAR**(255) **NOT NULL**,

address\_zip **VARCHAR**(6) **NOT NULL**,

address\_country **CHAR**(3) **NOT NULL**,

address\_state **CHAR**(2) **NOT NULL**,

userid **INTEGER NOT NULL**,

**CONSTRAINT** userid **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid)

);

**CREATE UNIQUE INDEX** "Address\_addressid\_uindex" **ON** address (addressid);

**CREATE INDEX** address\_active\_index **ON** address (active);

**CREATE TABLE** card\_type

(

card\_type **VARCHAR**(16) **PRIMARY KEY NOT NULL**,

active **BOOLEAN NOT NULL**

);

**CREATE TABLE** cart

(

cartid **INTEGER PRIMARY KEY NOT NULL**,

userid **INTEGER NOT NULL**,

active **BOOLEAN NOT NULL**,

insert\_date **DATE NOT NULL**,

**CONSTRAINT** cart\_cg\_user\_userid\_fk **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid)

);

**CREATE INDEX** cart\_active\_index **ON** cart (active);

**CREATE TABLE** cart\_contains

(

cartid **INTEGER NOT NULL**,

productid **INTEGER NOT NULL**,

cart\_product\_qty **INTEGER NOT NULL**,

insert\_date **DATE NOT NULL**,

**CONSTRAINT** cartcartid **FOREIGN KEY** (cartid) **REFERENCES** cart (cartid),

**CONSTRAINT** productid **FOREIGN KEY** (productid) **REFERENCES** product (productid)

);

**CREATE UNIQUE INDEX** cart\_contains\_cartid\_productid\_pk **ON** cart\_contains (cartid, productid);

**CREATE TABLE category**

(

categoryid **INTEGER PRIMARY KEY NOT NULL**,

**category VARCHAR**(255) **NOT NULL**,

active **BOOLEAN NOT NULL**

);

**CREATE INDEX** category\_active\_index **ON category** (active);

**CREATE TABLE** cg\_user

(

userid **INTEGER PRIMARY KEY NOT NULL**,

user\_firstname **VARCHAR**(255) **NOT NULL**,

user\_lastname **VARCHAR**(255) **NOT NULL**,

email **VARCHAR**(255) **NOT NULL**,

phone **VARCHAR**(15) **NOT NULL**,

dob **DATE NOT NULL**,

active **BOOLEAN NOT NULL**,

accountid **INTEGER NOT NULL**,

**CONSTRAINT** cg\_user\_account\_info\_accountid\_accountid\_fk **FOREIGN KEY** (accountid) **REFERENCES** account\_info (accountid)

);

**CREATE UNIQUE INDEX** "CG\_USER\_Email\_uindex" **ON** cg\_user (email);

**CREATE UNIQUE INDEX** cg\_user\_accountid\_uindex **ON** cg\_user (accountid);

**CREATE INDEX** "CG\_USER\_active\_index" **ON** cg\_user (active);

**CREATE TABLE** esrb\_rating

(

esrbid **INTEGER PRIMARY KEY NOT NULL**,

esrb\_rate **VARCHAR**(255) **NOT NULL**

);

**CREATE UNIQUE INDEX** "ESRB\_Rating\_ESRBID\_uindex" **ON** esrb\_rating (esrbid);

**CREATE TABLE** genre

(

genreid **INTEGER PRIMARY KEY NOT NULL**,

genre **VARCHAR**(255) **NOT NULL**,

active **BOOLEAN NOT NULL**

);

**CREATE UNIQUE INDEX** "Genre\_GenreId\_uindex" **ON** genre (genreid);

**CREATE INDEX** genre\_active\_index **ON** genre (active);

**CREATE TABLE** offer

(

offerid **INTEGER PRIMARY KEY NOT NULL**,

productid **INTEGER NOT NULL**,

offer\_price **NUMERIC**(100,2) **NOT NULL**,

offer\_start\_date **DATE NOT NULL**,

offer\_end\_date **DATE NOT NULL**,

**CONSTRAINT** productid **FOREIGN KEY** (productid) **REFERENCES** product (productid)

);

**CREATE UNIQUE INDEX** "Offer\_Offerid\_uindex" **ON** offer (offerid);

**CREATE TABLE** order\_details

(

orderid **INTEGER NOT NULL**,

productid **INTEGER NOT NULL**,

product\_price **NUMERIC**(100,2) **NOT NULL**,

product\_qty **INTEGER NOT NULL**,

**CONSTRAINT** orderid **FOREIGN KEY** (orderid) **REFERENCES** orders (orderid),

**CONSTRAINT** productid **FOREIGN KEY** (productid) **REFERENCES** product (productid)

);

**CREATE UNIQUE INDEX** "Order\_Details\_Productid\_uindex" **ON** order\_details (productid);

**CREATE INDEX** order\_details\_orderid\_productid\_index **ON** order\_details (orderid, productid);

**CREATE TABLE** order\_status

(

order\_statusid **INTEGER PRIMARY KEY NOT NULL**,

order\_status\_name **VARCHAR**(20) **NOT NULL**,

active **BOOLEAN NOT NULL**

);

**CREATE INDEX** order\_status\_active\_index **ON** order\_status (active);

**CREATE TABLE** orders

(

orderid **INTEGER PRIMARY KEY NOT NULL**,

cartid **INTEGER NOT NULL**,

shipment\_feeid **INTEGER NOT NULL**,

order\_statusid **INTEGER NOT NULL**,

shipping\_addressid **INTEGER NOT NULL**,

userid **INTEGER NOT NULL**,

payment\_methodid **INTEGER NOT NULL**,

order\_date **DATE NOT NULL**,

order\_total **NUMERIC**(100,2) **NOT NULL**,

**CONSTRAINT** order\_cart\_cartid\_fk **FOREIGN KEY** (cartid) **REFERENCES** cart (cartid),

**CONSTRAINT** order\_shippment\_fee\_shippment\_feeid\_fk **FOREIGN KEY** (shipment\_feeid) **REFERENCES** shipment\_fee (shipment\_feeid),

**CONSTRAINT** order\_order\_status\_order\_statusid\_fk **FOREIGN KEY** (order\_statusid) **REFERENCES** order\_status (order\_statusid),

**CONSTRAINT** order\_address\_addressid\_fk **FOREIGN KEY** (shipping\_addressid) **REFERENCES** address (addressid),

**CONSTRAINT** order\_cg\_user\_userid\_fk **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid),

**CONSTRAINT** order\_payment\_method\_payment\_methodid\_fk **FOREIGN KEY** (payment\_methodid) **REFERENCES** payment\_method (payment\_methodid)

);

**CREATE UNIQUE INDEX** "Order\_Cartid\_uindex" **ON** orders (cartid);

**CREATE INDEX** "Order\_Shipping\_AddressId\_index" **ON** orders (shipping\_addressid);

**CREATE INDEX** "Order\_Userid\_index" **ON** orders (userid);

**CREATE INDEX** "Order\_Payment\_MethodId\_index" **ON** orders (payment\_methodid);

**CREATE TABLE** payment\_method

(

payment\_methodid **INTEGER PRIMARY KEY NOT NULL**,

card\_name **VARCHAR**(255) **NOT NULL**,

card\_last\_four\_digits **CHAR**(4) **NOT NULL**,

card\_number **VARCHAR**(255) **NOT NULL**,

card\_exp\_date **DATE NOT NULL**,

cvc **VARCHAR**(4) **NOT NULL**,

card\_type **VARCHAR**(25) **NOT NULL**,

userid **INTEGER NOT NULL**,

billing\_addressid **INTEGER NOT NULL**,

active **BOOLEAN NOT NULL**,

**CONSTRAINT** payment\_method\_card\_type\_card\_type\_fk **FOREIGN KEY** (card\_type) **REFERENCES** card\_type (card\_type),

**CONSTRAINT** payment\_method\_cg\_user\_userid\_fk **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid),

**CONSTRAINT** payment\_method\_address\_addressid\_fk **FOREIGN KEY** (billing\_addressid) **REFERENCES** address (addressid)

);

**CREATE UNIQUE INDEX** "Payment\_Method\_Card\_Number\_uindex" **ON** payment\_method (card\_number);

**CREATE INDEX** payment\_method\_active\_index **ON** payment\_method (active);

**CREATE TABLE** platform

(

platformid **INTEGER PRIMARY KEY NOT NULL**,

logoimg **VARCHAR**(255) **NOT NULL**,

active **BOOLEAN NOT NULL**,

platform **VARCHAR**(50) **NOT NULL**

);

**CREATE UNIQUE INDEX** platfrom\_logoimg\_key **ON** platform (logoimg);

**CREATE UNIQUE INDEX** platform\_platform\_uindex **ON** platform (platform);

**CREATE INDEX** platform\_active\_index **ON** platform (active);

**CREATE TABLE** platform\_announcements

(

paid **INTEGER PRIMARY KEY NOT NULL**,

pa\_title **VARCHAR**(255) **NOT NULL**,

pa\_img **VARCHAR**(255) **NOT NULL**,

platformid **INTEGER NOT NULL**,

active **BOOLEAN NOT NULL**,

**CONSTRAINT** platform\_announcements\_platfrom\_platformid\_fk **FOREIGN KEY** (platformid) **REFERENCES** platform (platformid)

);

**CREATE INDEX** platform\_announcements\_platformid\_index **ON** platform\_announcements (platformid);

**CREATE INDEX** platform\_announcements\_active\_index **ON** platform\_announcements (active);

**CREATE TABLE** product

(

productid **INTEGER PRIMARY KEY NOT NULL**,

categoryid **INTEGER NOT NULL**,

genreid **INTEGER NOT NULL**,

esrbid **INTEGER NOT NULL**,

platformid **INTEGER**,

product\_price **NUMERIC**(100,2) **NOT NULL**,

release\_date **DATE NOT NULL**,

product\_qty **INTEGER NOT NULL**,

product\_description **VARCHAR**(2244) **NOT NULL**,

prodcut\_additional\_info **VARCHAR**(2244) **NOT NULL**,

active **BOOLEAN NOT NULL**,

product\_title **VARCHAR**(250) **NOT NULL**,

**CONSTRAINT** product\_category\_categoryid\_fk **FOREIGN KEY** (categoryid) **REFERENCES category** (categoryid),

**CONSTRAINT** product\_genre\_genreid\_fk **FOREIGN KEY** (genreid) **REFERENCES** genre (genreid),

**CONSTRAINT** product\_esrb\_rating\_esrbid\_fk **FOREIGN KEY** (esrbid) **REFERENCES** esrb\_rating (esrbid),

**CONSTRAINT** product\_platfrom\_platformid\_fk **FOREIGN KEY** (platformid) **REFERENCES** platform (platformid)

);

**CREATE INDEX** "Product\_Active\_index" **ON** product (active);

**CREATE TABLE** product\_img

(

piid **INTEGER PRIMARY KEY NOT NULL**,

productid **INTEGER NOT NULL**,

product\_img **VARCHAR**(255) **NOT NULL**,

cover **BOOLEAN NOT NULL**,

**CONSTRAINT** product\_img\_product\_productid\_fk **FOREIGN KEY** (productid) **REFERENCES** product (productid)

);

**CREATE INDEX** product\_img\_cover\_index **ON** product\_img (cover);

**CREATE TABLE** rate

(

rateid **INTEGER PRIMARY KEY NOT NULL**,

productid **INTEGER NOT NULL**,

rate\_date **DATE NOT NULL**,

rate **NUMERIC**(10,2) **NOT NULL**,

**CONSTRAINT** rate\_product\_productid\_fk **FOREIGN KEY** (productid) **REFERENCES** product (productid)

);

**CREATE INDEX** rate\_rate\_index **ON** rate (rate);

**CREATE TABLE role**

(

roleid **INTEGER PRIMARY KEY NOT NULL**,

**role VARCHAR**(10) **NOT NULL**

);

**CREATE TABLE** shipment\_fee

(

shipment\_feeid **INTEGER PRIMARY KEY NOT NULL**,

fee **NUMERIC**(100,2) **NOT NULL**,

active **BOOLEAN NOT NULL**,

fee\_description **VARCHAR**(255) **NOT NULL**

);

**CREATE INDEX** shippment\_fee\_active\_index **ON** shipment\_fee (active);

**CREATE TABLE** store\_announcement

(

said **INTEGER PRIMARY KEY NOT NULL**,

sa\_img **VARCHAR**(255) **NOT NULL**,

sa\_title **VARCHAR**(255) **NOT NULL**,

active **BOOLEAN NOT NULL**

);

**CREATE INDEX** store\_announcement\_active\_index **ON** store\_announcement (active);

**CREATE TABLE** user\_preferences

(

userid **INTEGER PRIMARY KEY NOT NULL**,

shipping\_addressid **INTEGER NOT NULL**,

billing\_addressid **INTEGER NOT NULL**,

payment\_methodid **INTEGER**,

**CONSTRAINT** user\_preferences\_cg\_user\_userid\_fk **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid),

**CONSTRAINT** user\_preferences\_shipping\_address\_addressid\_fk **FOREIGN KEY** (shipping\_addressid) **REFERENCES** address (addressid),

**CONSTRAINT** user\_preferences\_billing\_address\_addressid\_fk **FOREIGN KEY** (billing\_addressid) **REFERENCES** address (addressid),

**CONSTRAINT** user\_preferences\_payment\_method\_payment\_methodid\_fk **FOREIGN KEY** (payment\_methodid) **REFERENCES** payment\_method (payment\_methodid)

);

**CREATE UNIQUE INDEX** user\_preferences\_shipping\_addressid\_uindex **ON** user\_preferences (shipping\_addressid);

**CREATE TABLE** wishlist\_contains

(

productid **INTEGER NOT NULL**,

userid **INTEGER NOT NULL**,

**CONSTRAINT** wishlist\_contains\_pkey **PRIMARY KEY** (userid, productid),

**CONSTRAINT** productid **FOREIGN KEY** (productid) **REFERENCES** product (productid),

**CONSTRAINT** userid **FOREIGN KEY** (userid) **REFERENCES** cg\_user (userid)

);

**CREATE INDEX** wishlist\_contains\_userid\_index **ON** wishlist\_contains (userid);

**CREATE TABLE** active\_cart

(

userid **INTEGER**,

pid **INTEGER**,

pname **VARCHAR**(250),

pprice **NUMERIC**(100,2),

pquantity **INTEGER**,

cartid **INTEGER**

);

**CREATE TABLE** order\_errors\_check

(

orderid **INTEGER**,

order\_satus **VARCHAR**(20),

order\_total **NUMERIC**(100,2),

shippment\_fee **NUMERIC**(100,2),

order\_total\_error **NUMERIC**,

cart\_active **BOOLEAN**

);

**CREATE TABLE** order\_subtotal

(

orderid **INTEGER**,

osubtotal **NUMERIC**

);

**CREATE TABLE** order\_summary

(

**oid INTEGER**,

odate **TEXT**,

cid **INTEGER**,

ostatus **VARCHAR**(20),

osubtotal **NUMERIC**,

ototal **NUMERIC**(100,2),

shipmment\_fee **NUMERIC**(100,2),

oshippingaddressid **INTEGER**,

obillingaddressid **INTEGER**

);

**CREATE TABLE** product\_details

(

pid **INTEGER**,

photolink **VARCHAR**(255),

title **VARCHAR**(250),

platformid **INTEGER**,

genre **VARCHAR**(255),

**category VARCHAR**(255),

esrb **VARCHAR**(255),

**release TEXT**,

availability **BOOLEAN**,

price **NUMERIC**(100,2),

description **VARCHAR**(2244),

aditionalinfo **VARCHAR**(2244),

inoffer **BOOLEAN**,

offerprice **REAL**,

rating **NUMERIC**

);

**CREATE TABLE** wishlist

(

pid **INTEGER**,

pname **VARCHAR**(250),

pprice **NUMERIC**(100,2),

userid **INTEGER**

);

**CREATE FUNCTION** armor(**BYTEA**) **RETURNS TEXT**;

**CREATE FUNCTION** armor(**BYTEA**, **TEXT**[], **TEXT**[]) **RETURNS TEXT**;

**CREATE FUNCTION** crypt(**TEXT**, **TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** dearmor(**TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** decrypt\_iv(**BYTEA**, **BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** decrypt(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** digest(**BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** digest(**TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** encrypt\_iv(**BYTEA**, **BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** encrypt(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** gen\_random\_bytes(**INTEGER**) **RETURNS BYTEA**;

**CREATE FUNCTION** gen\_random\_uuid() **RETURNS UUID**;

**CREATE FUNCTION** gen\_salt(**TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** gen\_salt(**TEXT**, **INTEGER**) **RETURNS TEXT**;

**CREATE FUNCTION** hmac(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** hmac(**TEXT**, **TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_armor\_headers(**TEXT**, **key OUT TEXT**, **value OUT TEXT**) **RETURNS SETOF RECORD**;

**CREATE FUNCTION** pgp\_key\_id(**BYTEA**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_pub\_decrypt\_bytea(**BYTEA**, **BYTEA**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_decrypt\_bytea(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_decrypt\_bytea(**BYTEA**, **BYTEA**, **TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_decrypt(**BYTEA**, **BYTEA**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_pub\_decrypt(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_pub\_decrypt(**BYTEA**, **BYTEA**, **TEXT**, **TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_pub\_encrypt\_bytea(**BYTEA**, **BYTEA**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_encrypt\_bytea(**BYTEA**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_encrypt(**TEXT**, **BYTEA**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_pub\_encrypt(**TEXT**, **BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_decrypt\_bytea(**BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_decrypt\_bytea(**BYTEA**, **TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_decrypt(**BYTEA**, **TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_sym\_decrypt(**BYTEA**, **TEXT**, **TEXT**) **RETURNS TEXT**;

**CREATE FUNCTION** pgp\_sym\_encrypt\_bytea(**BYTEA**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_encrypt\_bytea(**BYTEA**, **TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_encrypt(**TEXT**, **TEXT**) **RETURNS BYTEA**;

**CREATE FUNCTION** pgp\_sym\_encrypt(**TEXT**, **TEXT**, **TEXT**) **RETURNS BYTEA**;

**7.     Client – Side Screen Shots**

**General Views**

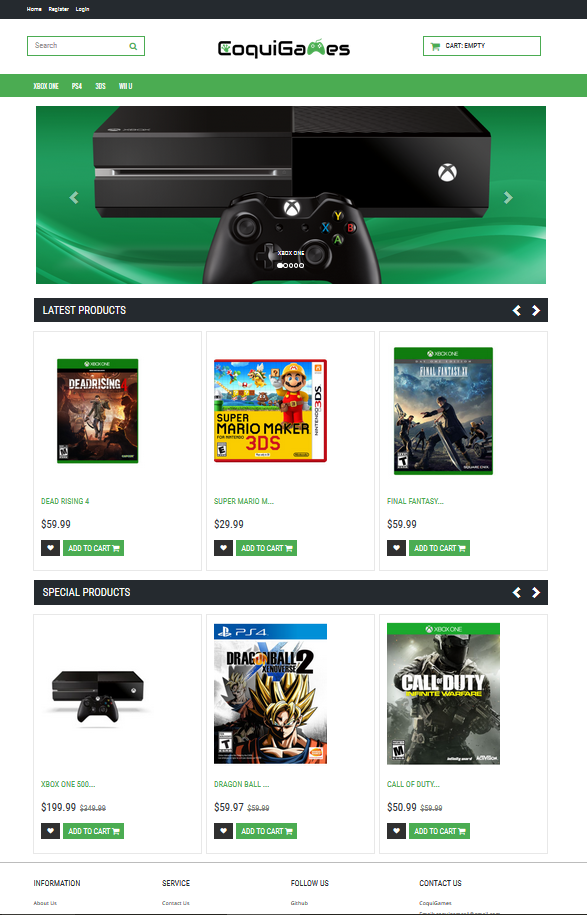


Figure 1: Home View

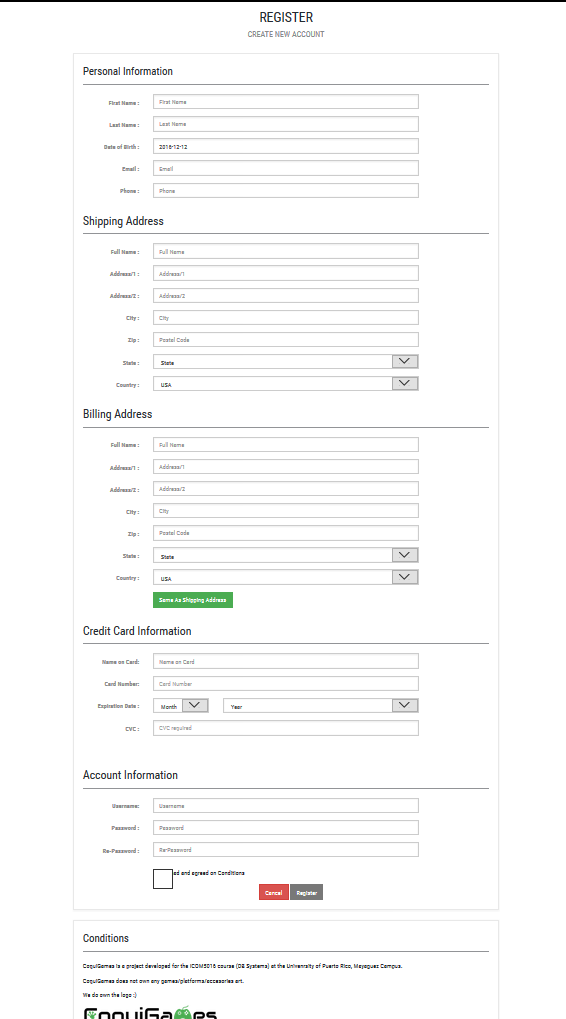


Figure 2: Register View

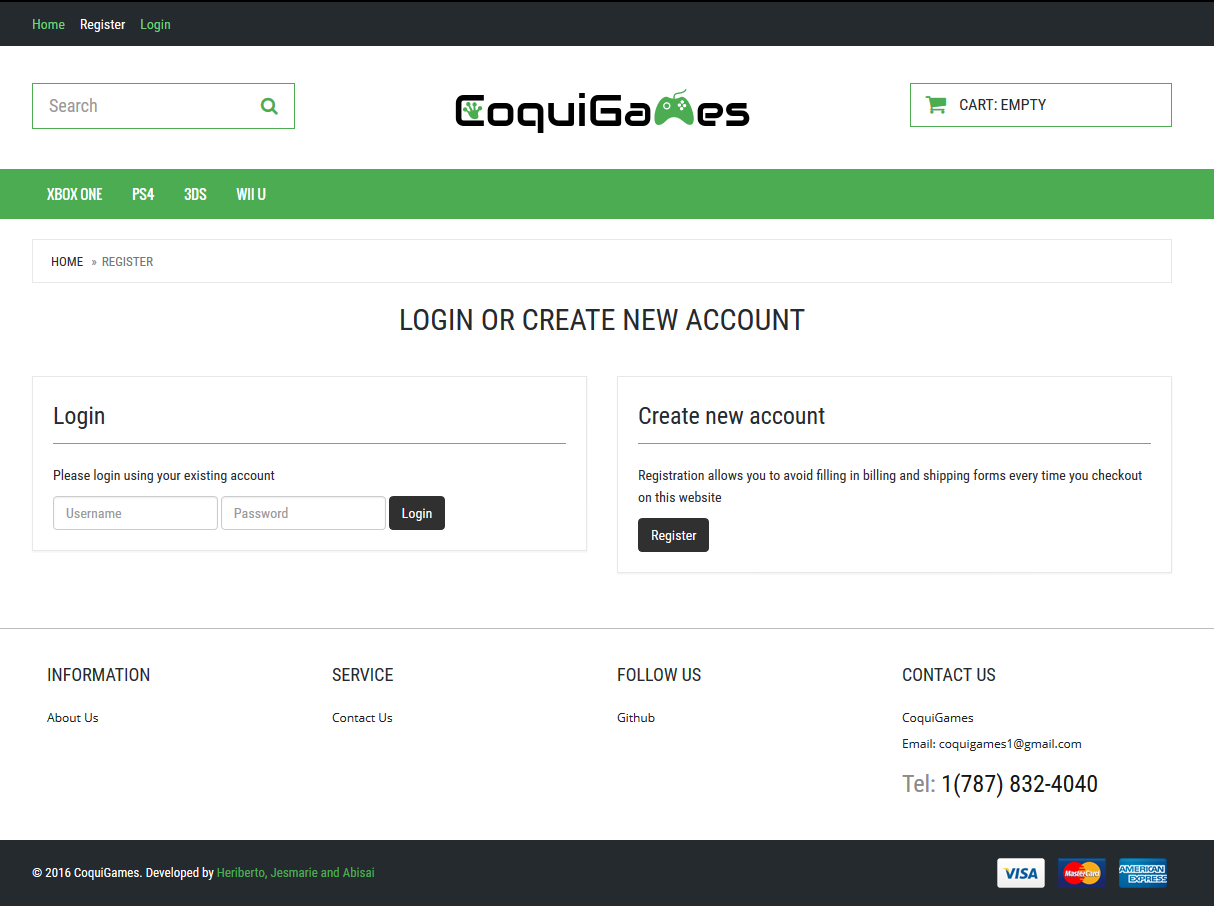


Figure 3: Login View

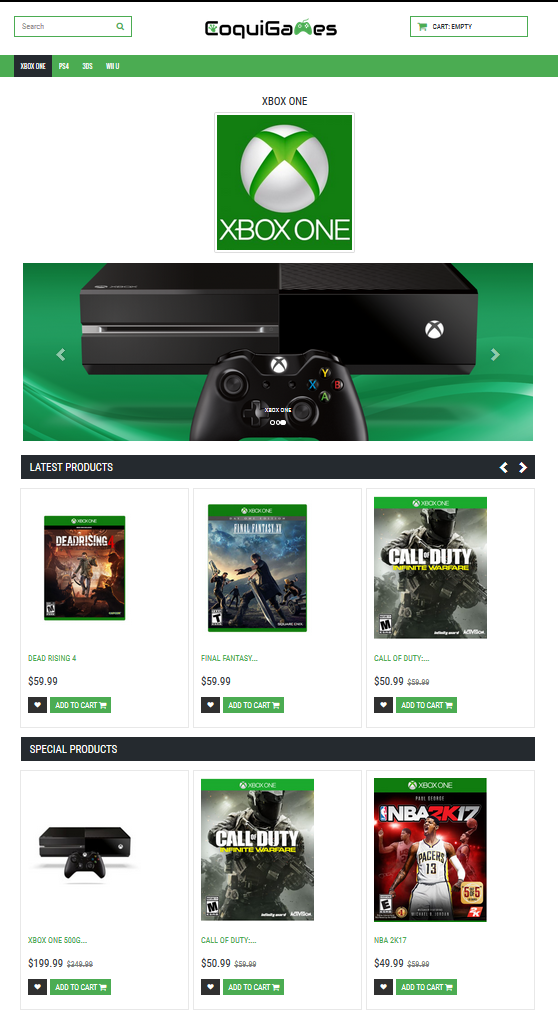


Figure 4: Xbox Platform View

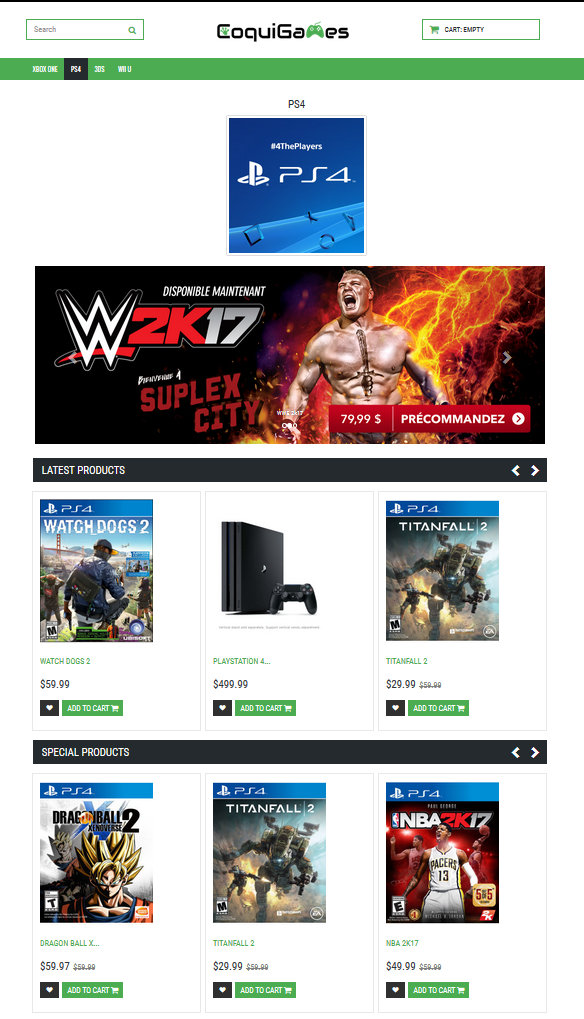


Figure 5: PS4 Platform View

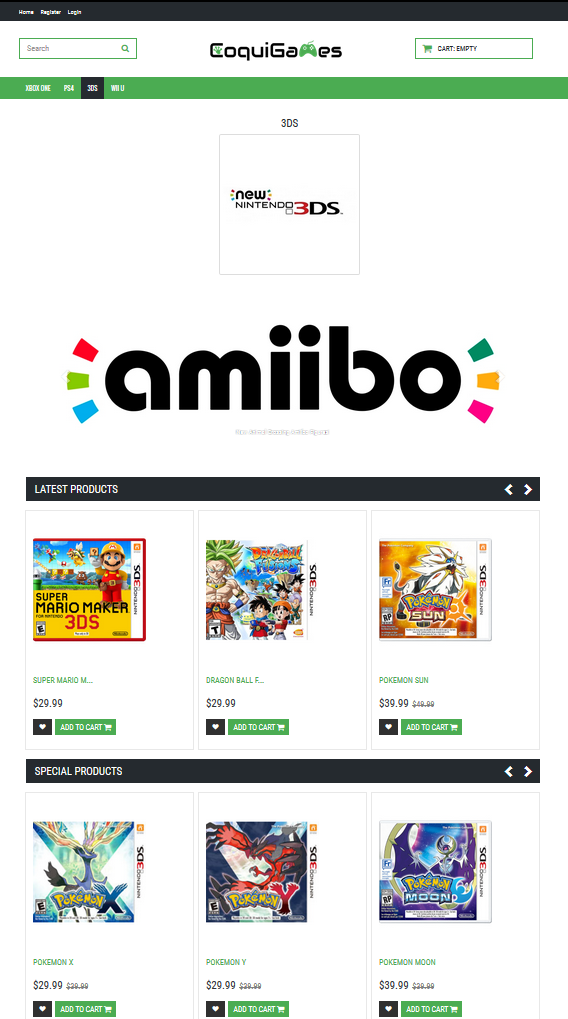


Figure 6: 3DS Platform View

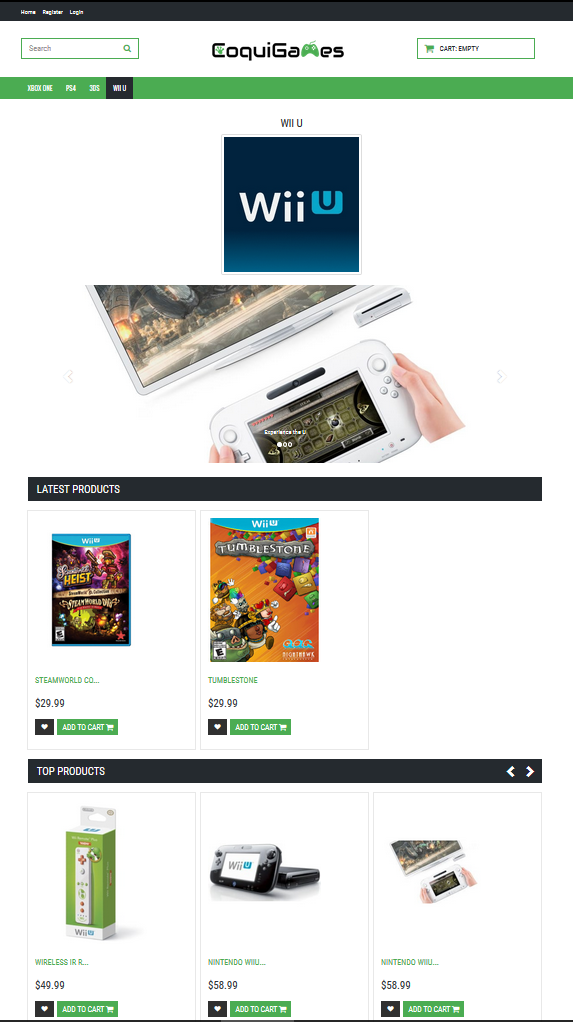


Figure 7: WiiU Platform View

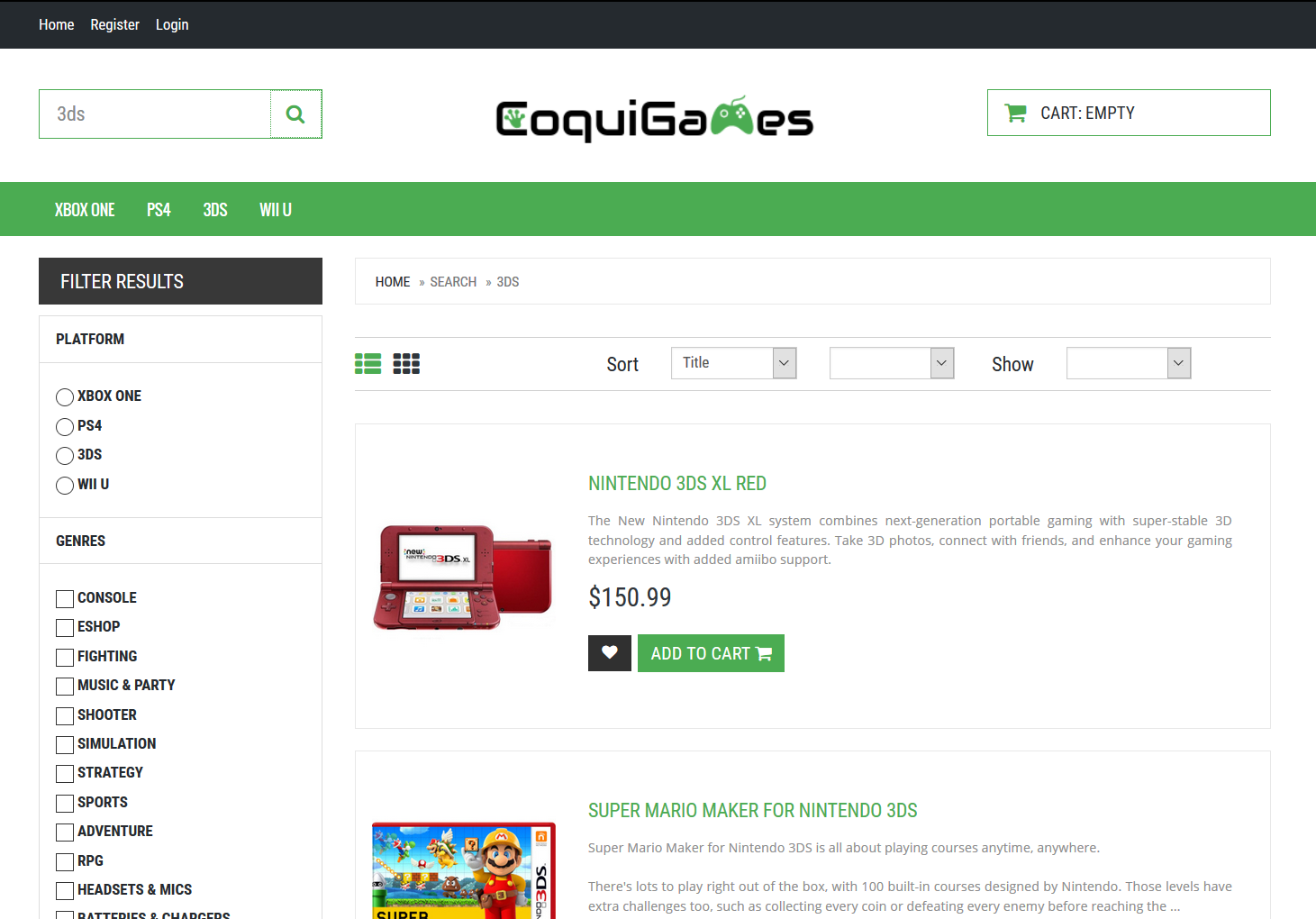


Figure 8: Search View

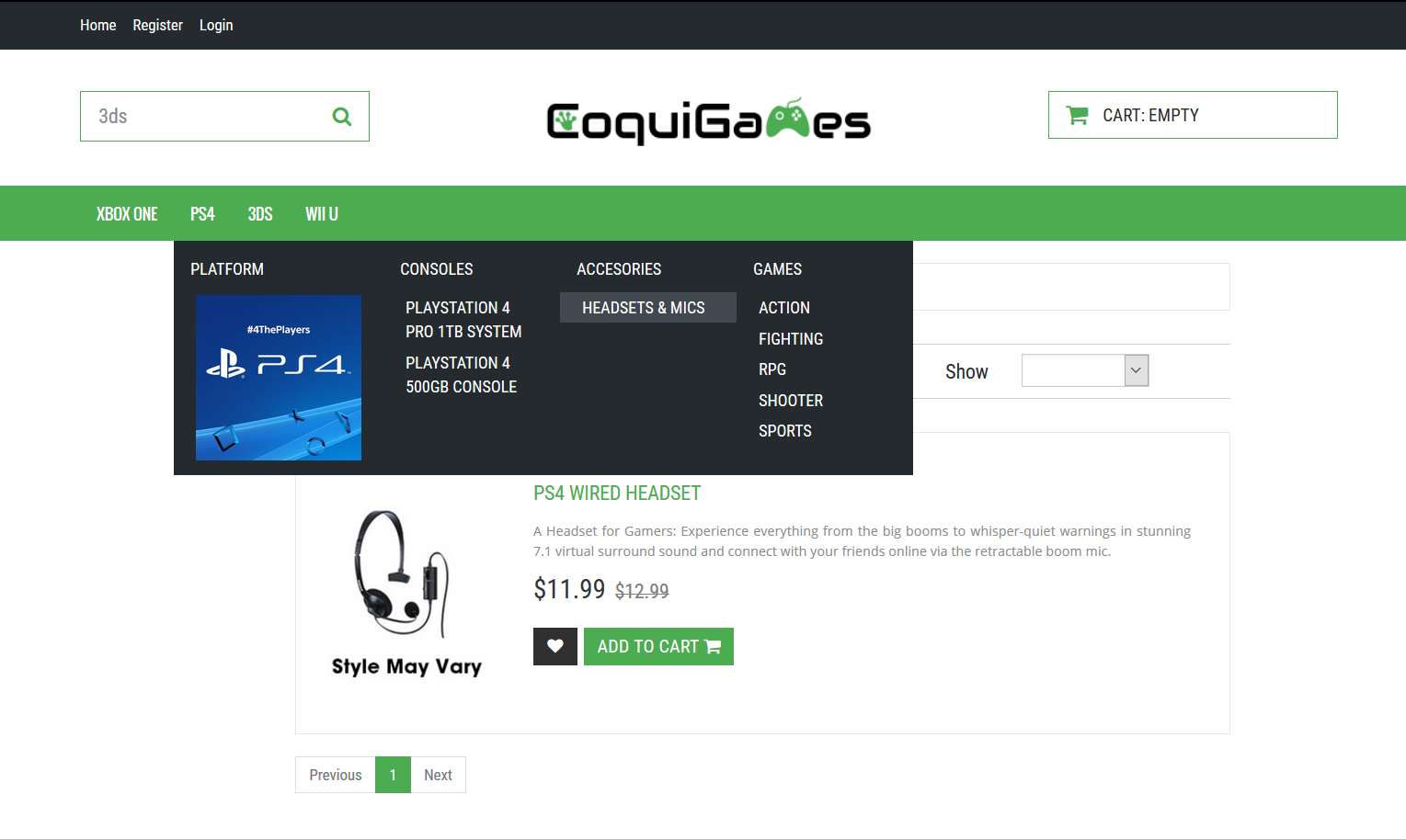


Figure 9: Tag Link Result View

**User (Customer) Views**



Figure 10: Wishlist View

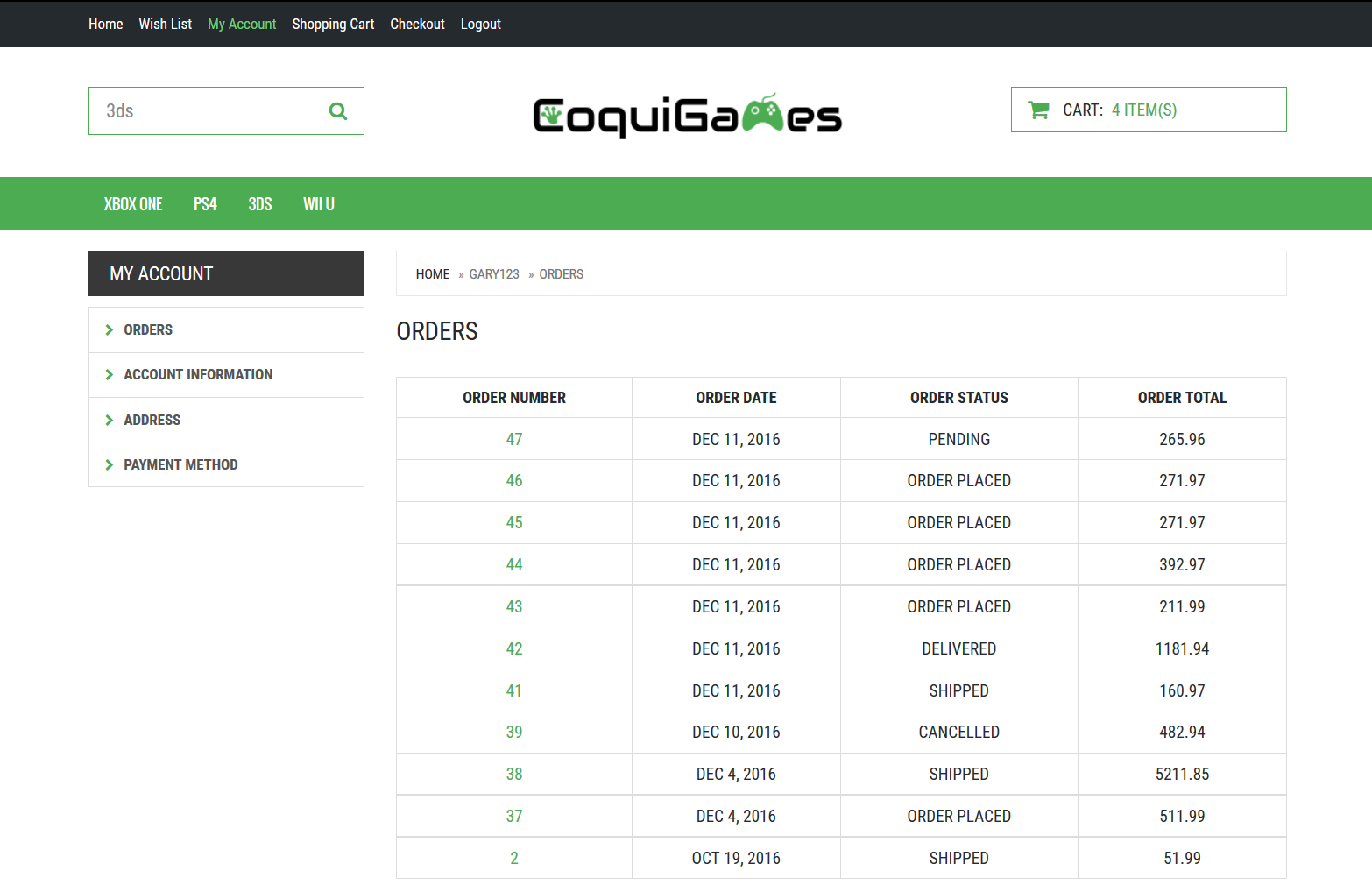


Figure 11: MyAccount-Orders View

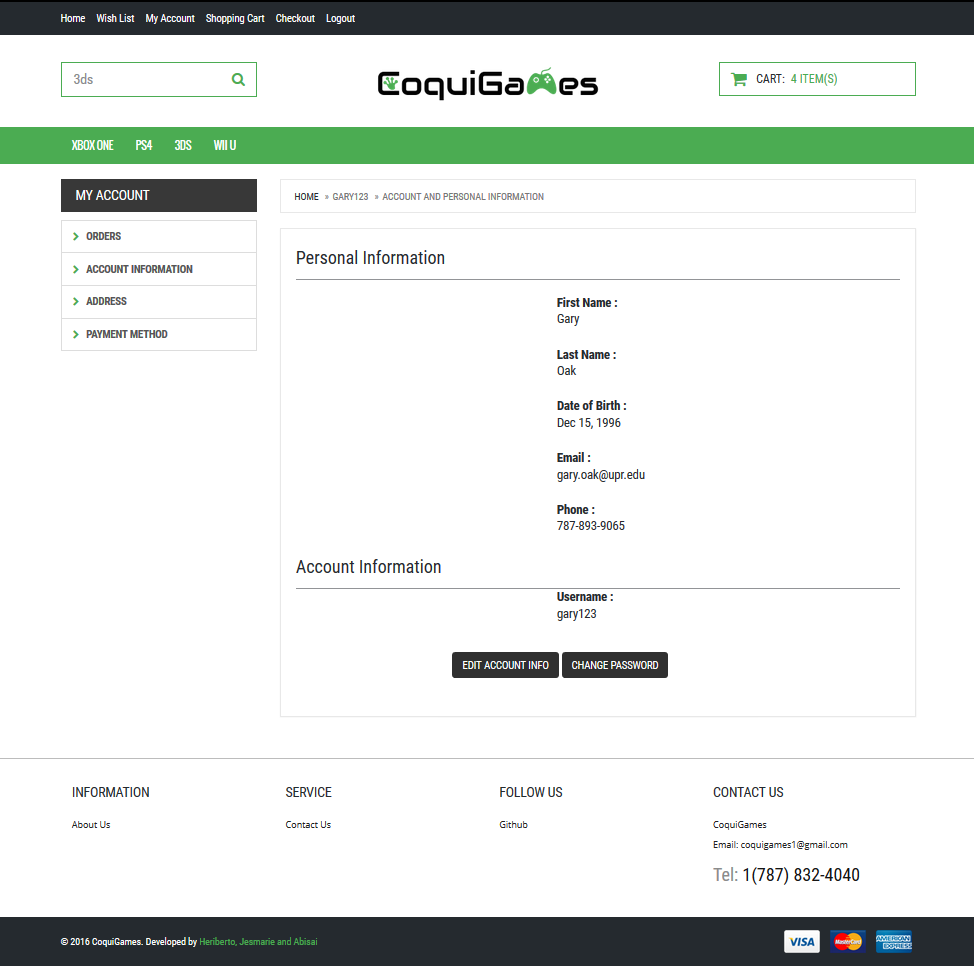


Figure 12: MyAccount-Account Information View

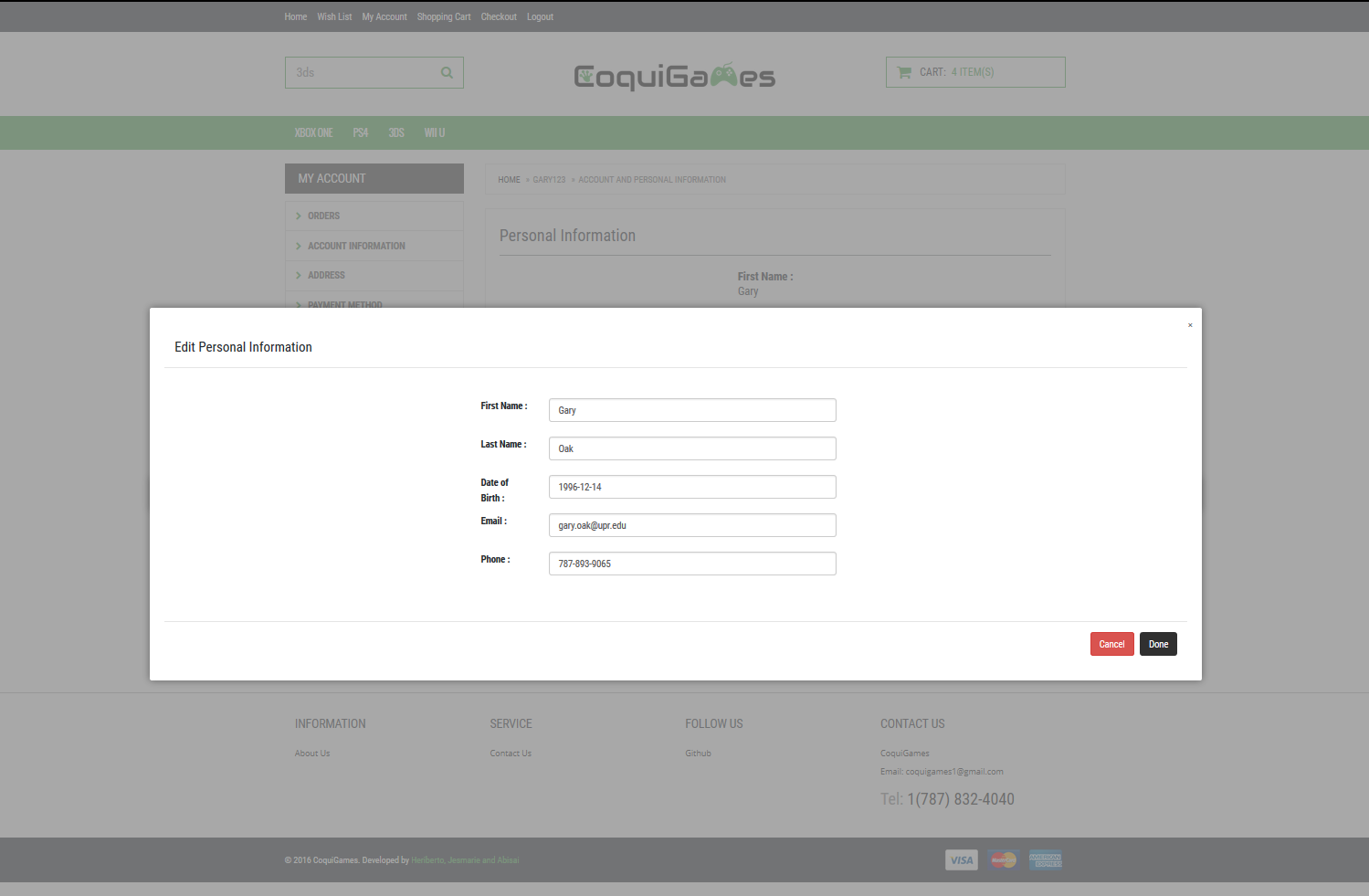


Figure 13: MyAccount-Account Information Edit Personal Info View

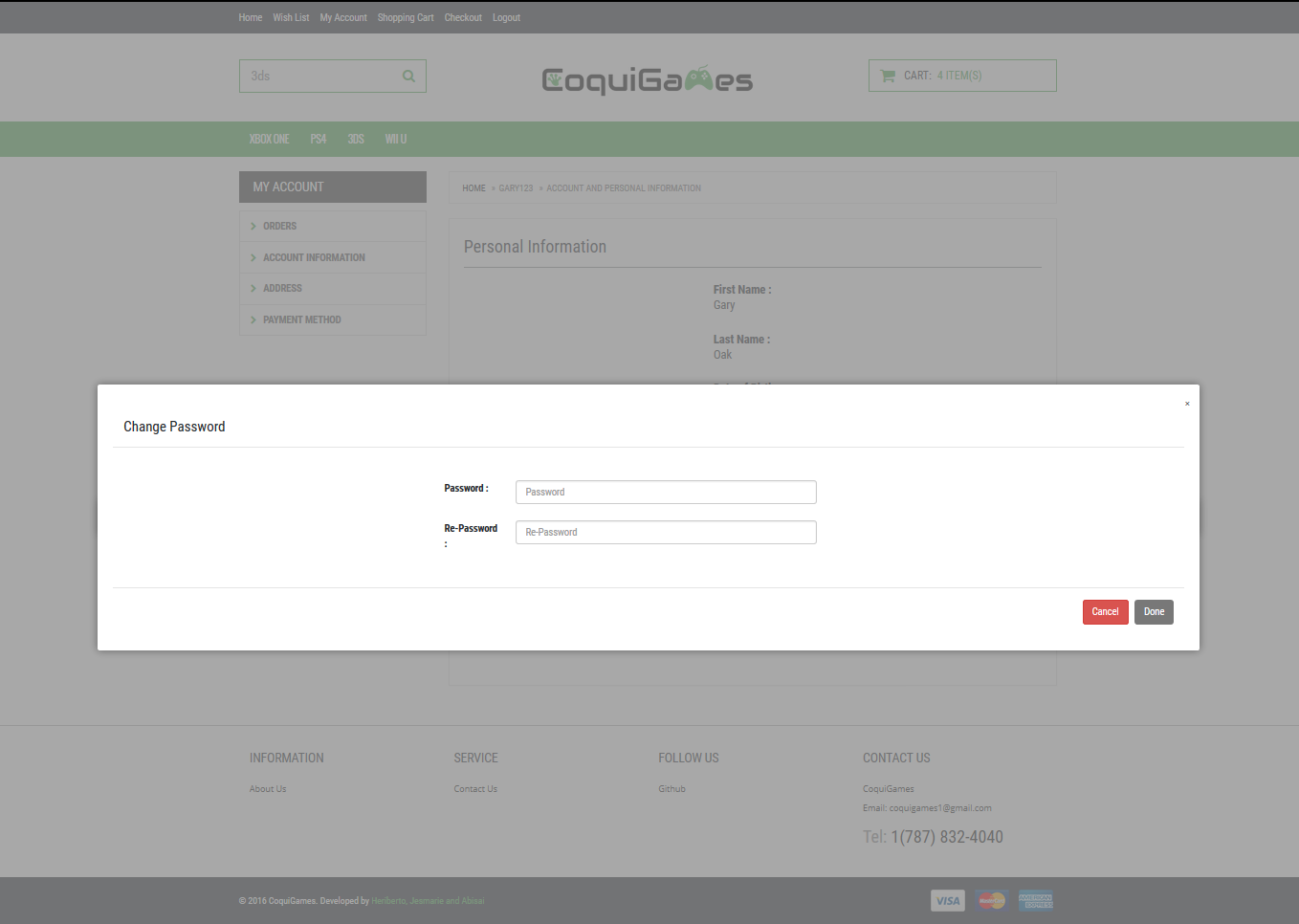


Figure 14: MyAccount-Account Information Change Password Info View

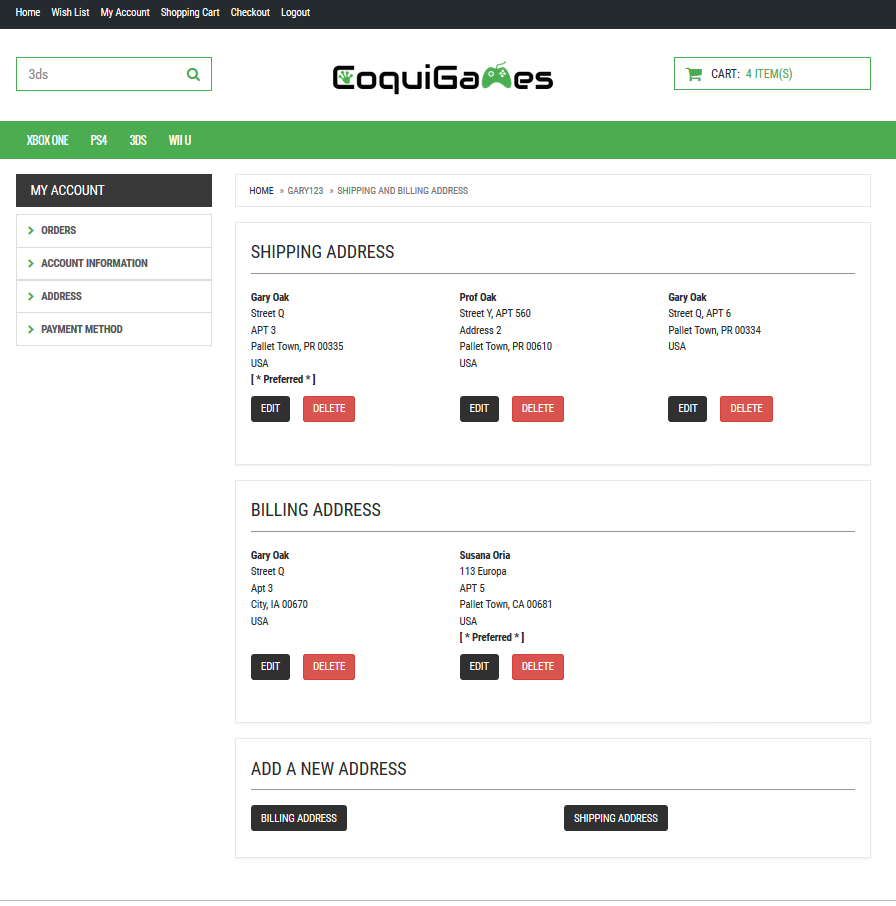


Figure 15: MyAccount-Address Information View

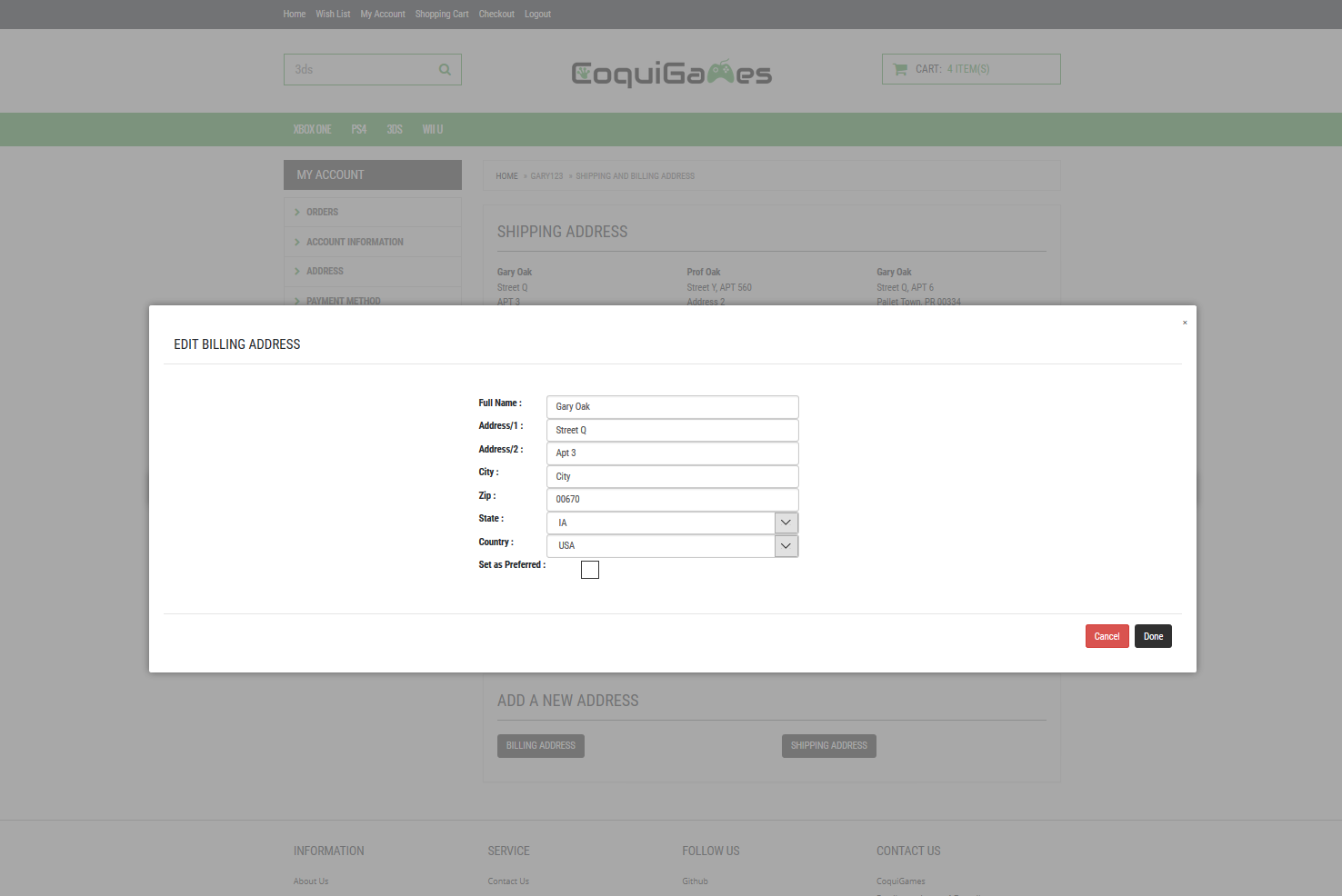


Figure 16: MyAccount-Address Information Edit/Add Address View

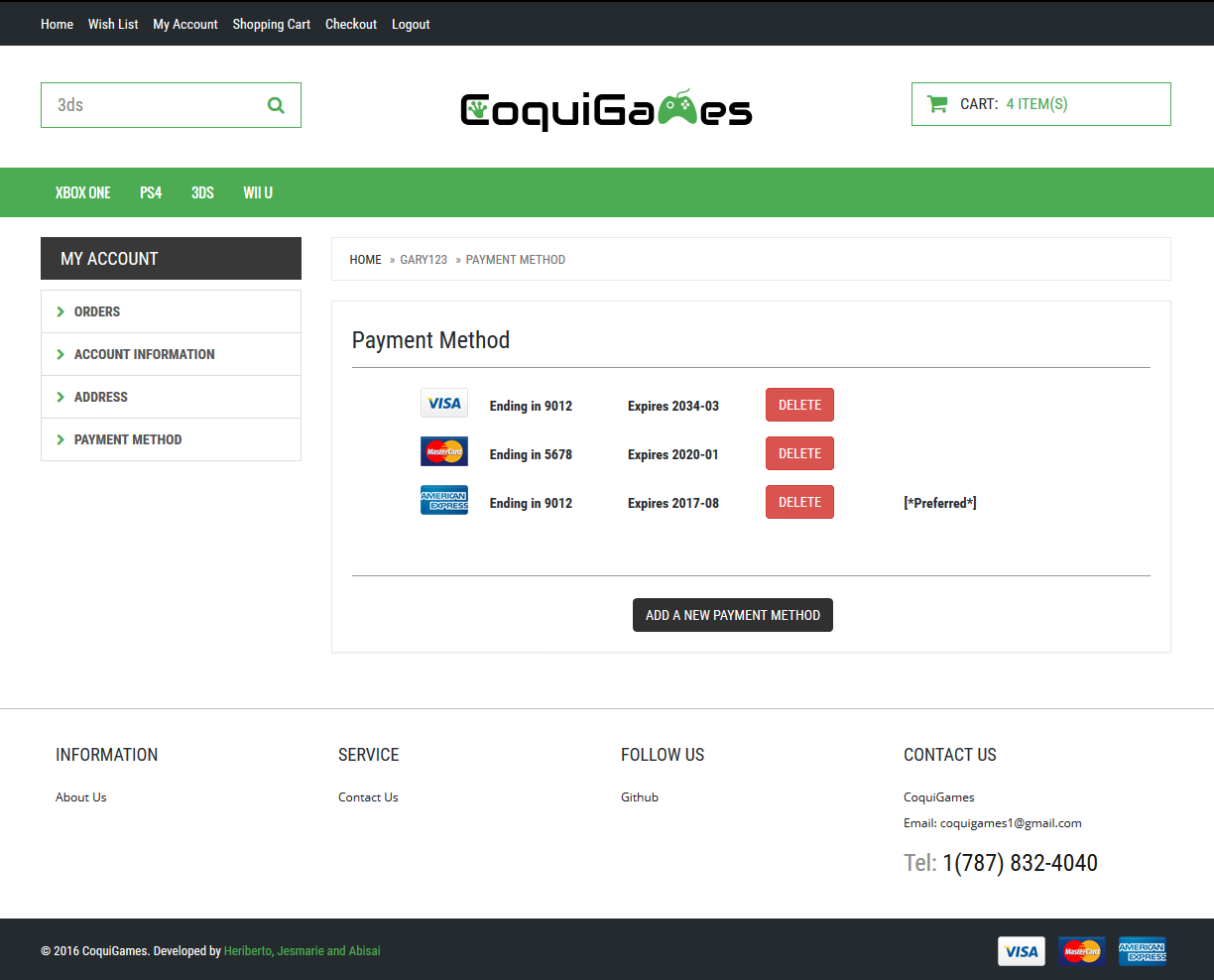


Figure 17: MyAccount-Payment Method View

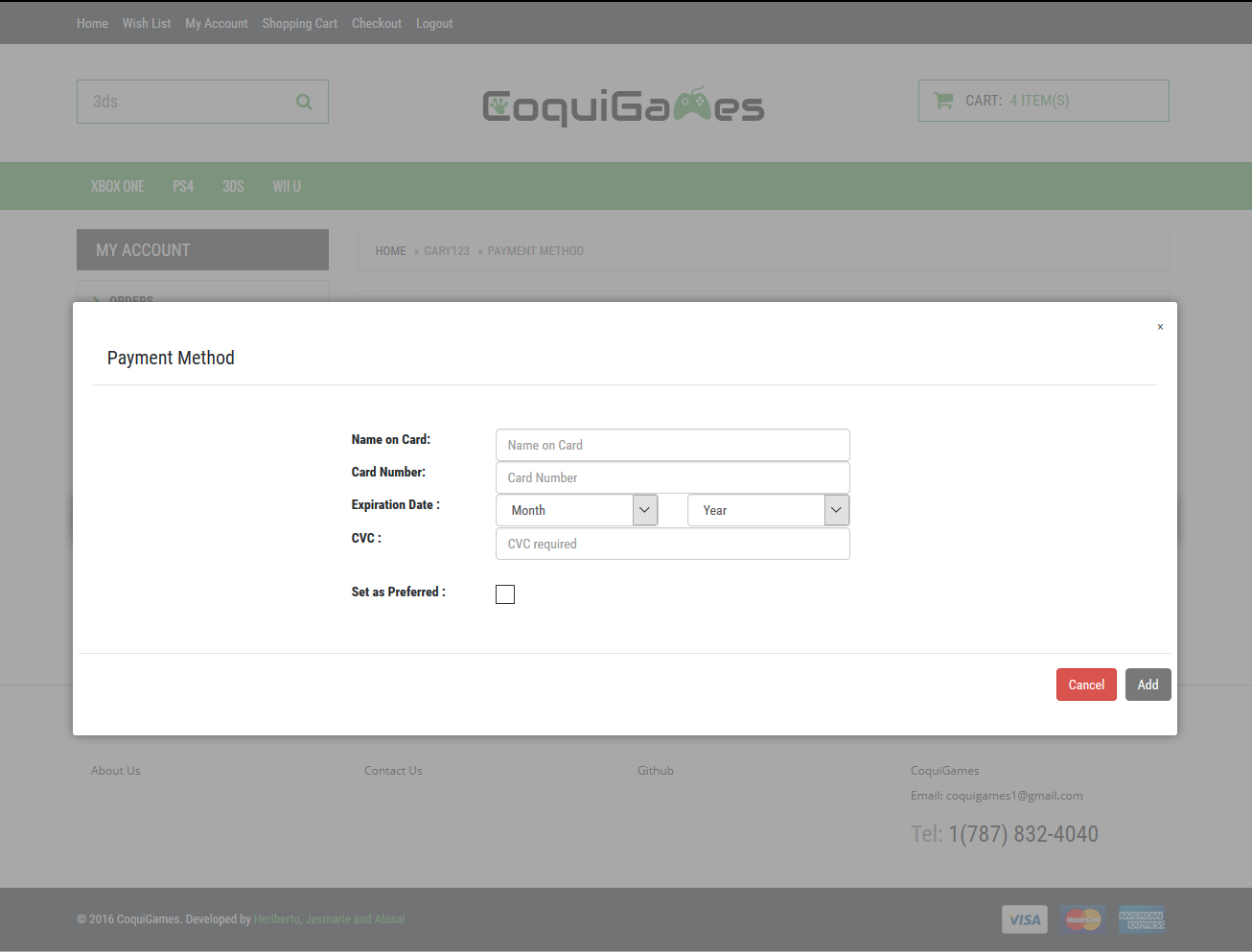


Figure 18: MyAccount-Add Payment Method View

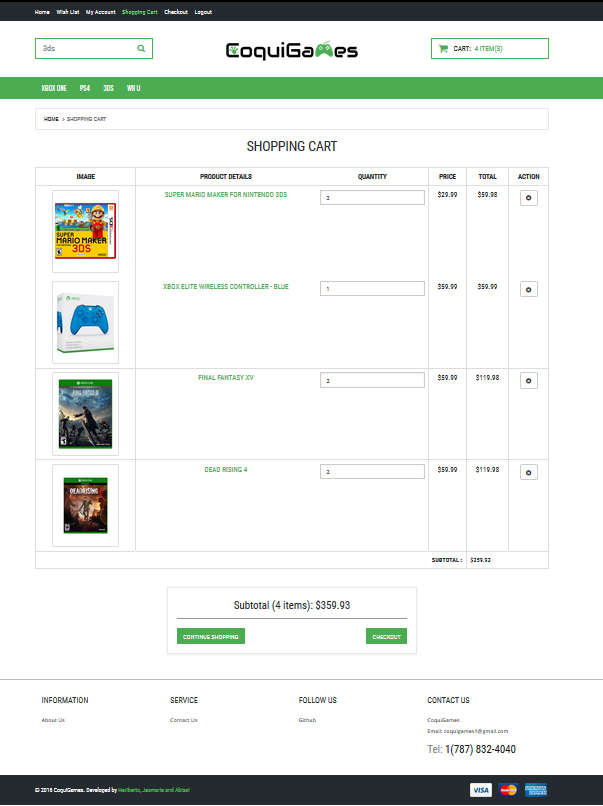


Figure 19: Shopping Cart View

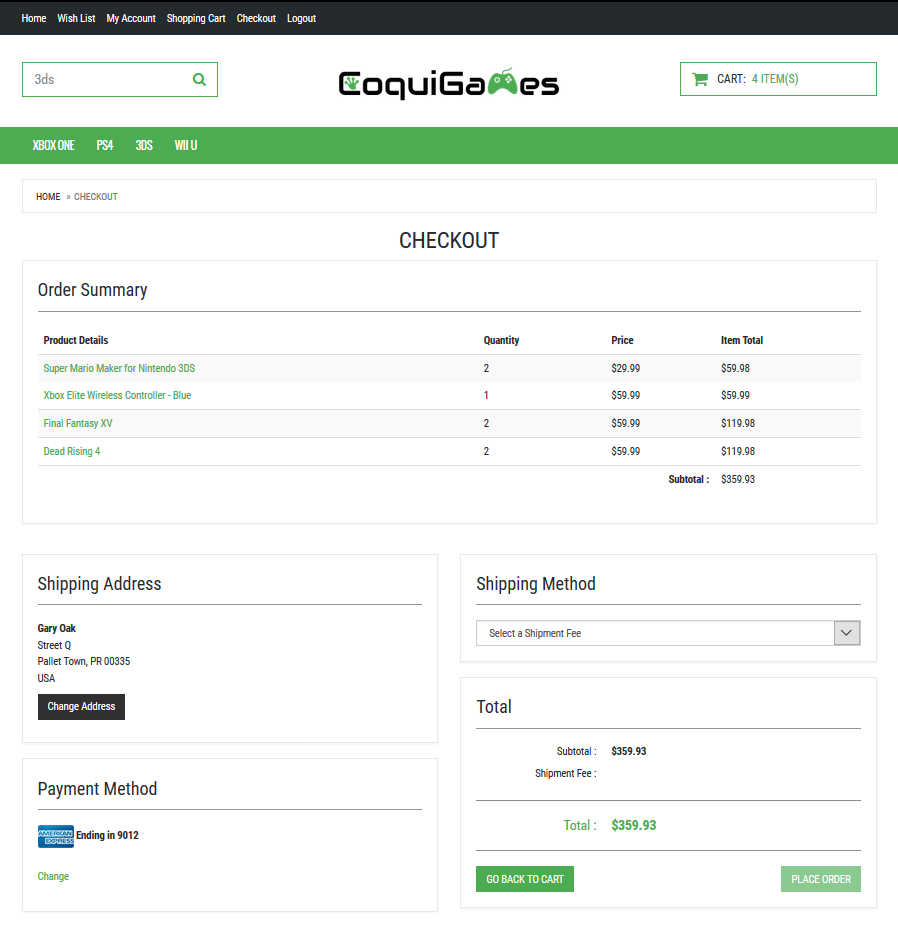


Figure 20: Checkout View

**User (Admin) Views**



Figure 21: Users Table View

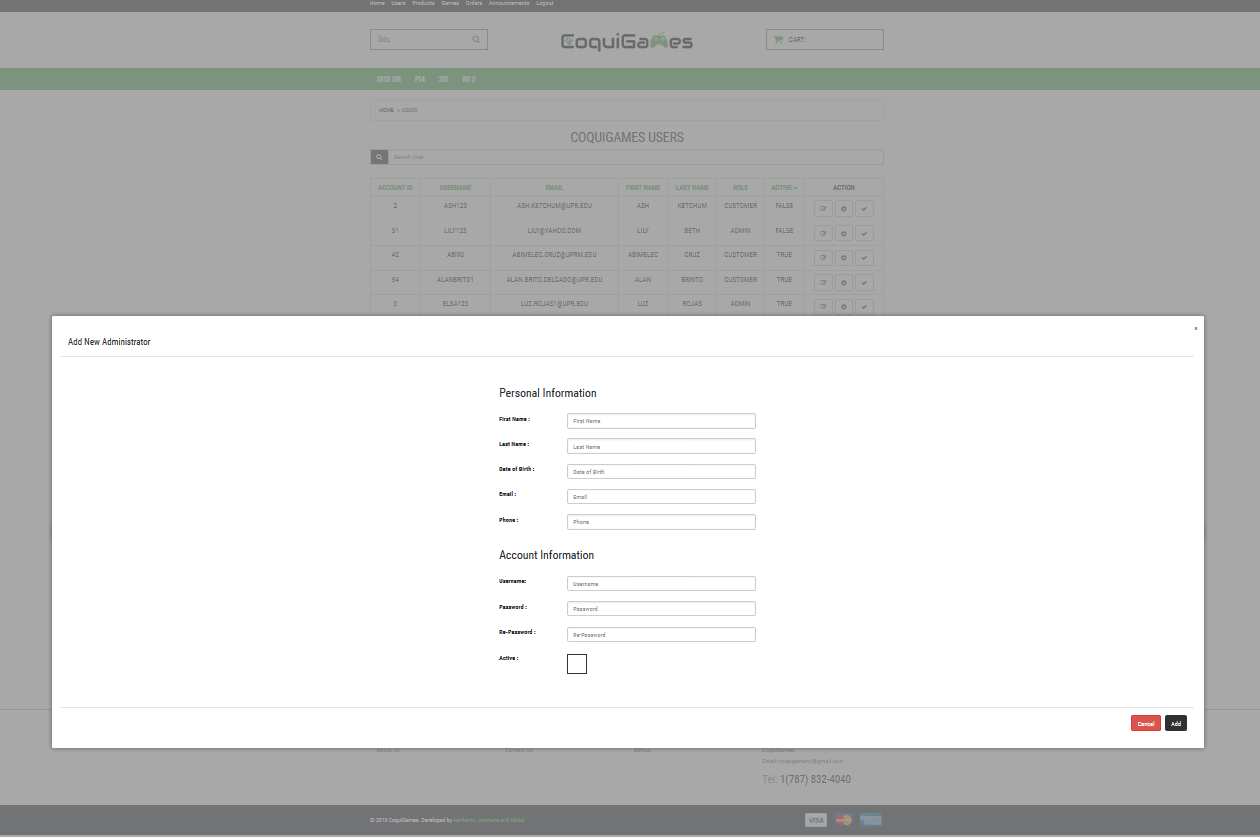


Figure 22: Users Table Add Admin View



Figure 23: Users Table Edit Admin View

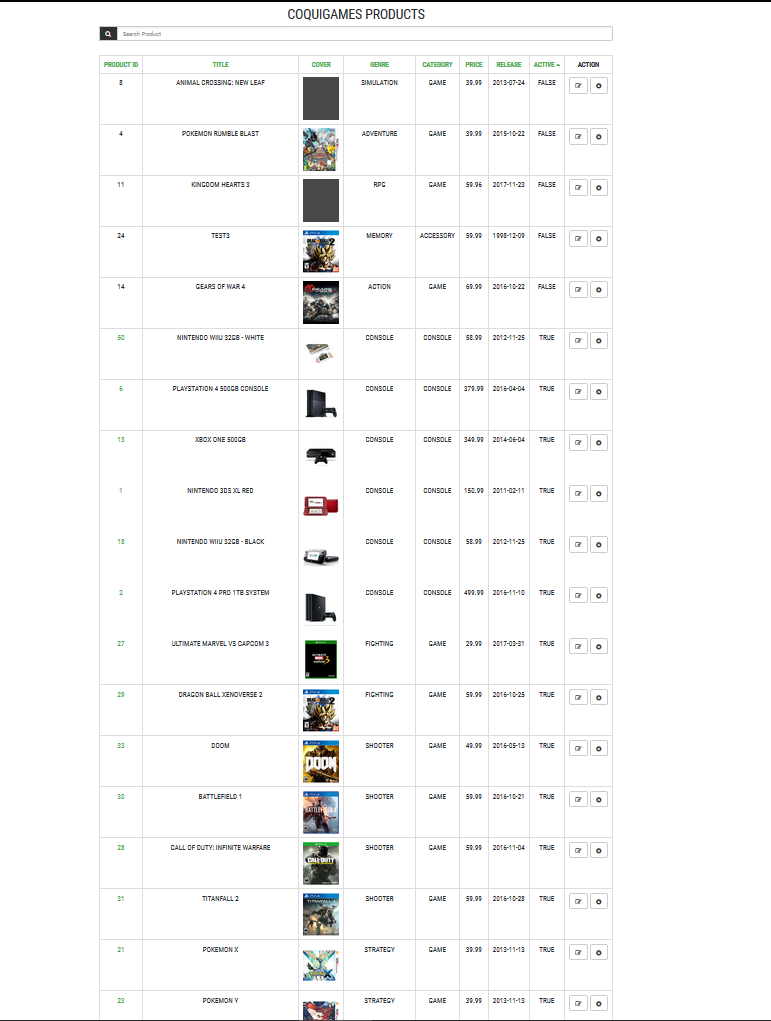


Figure 24: Products Table View

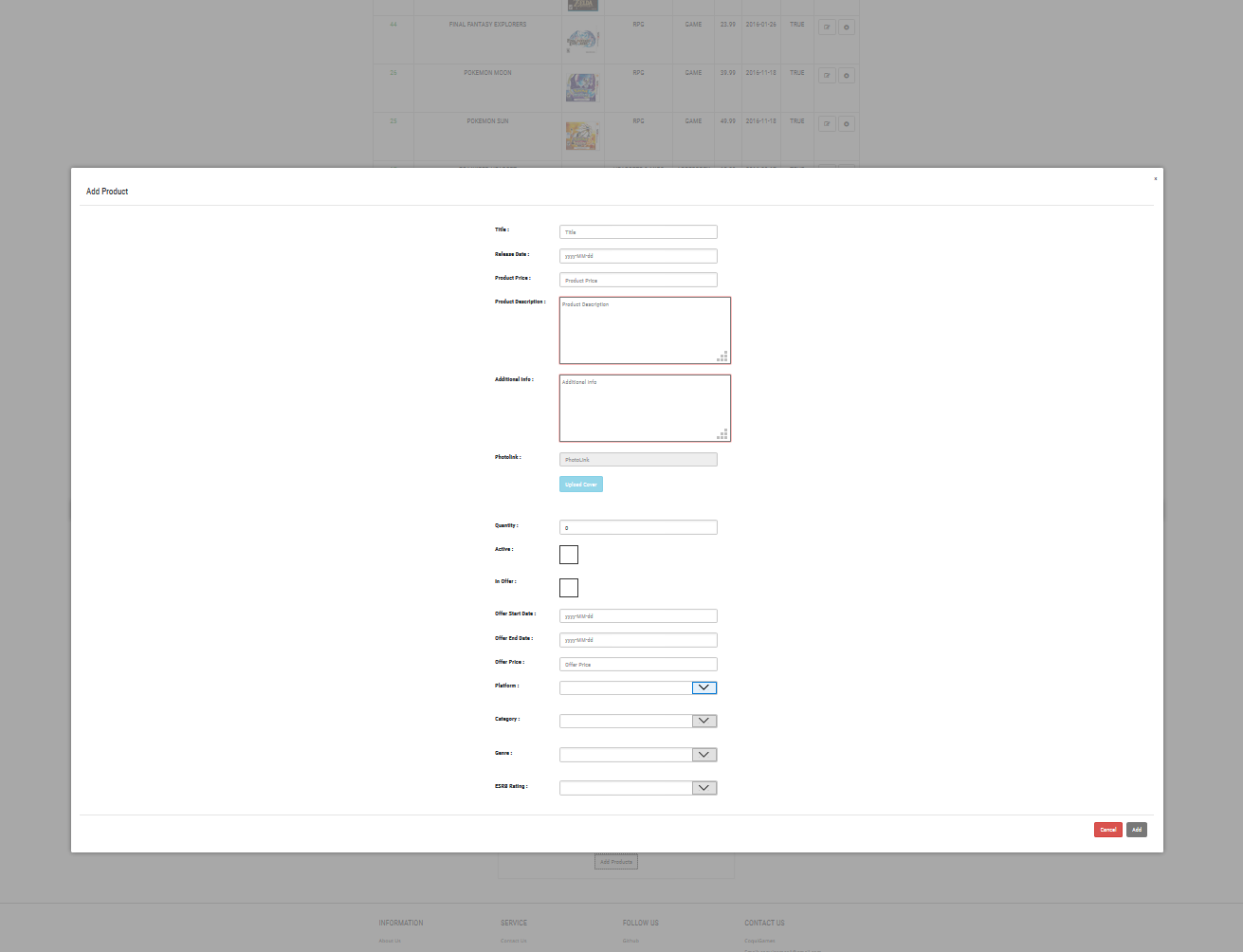


Figure 25: Products Table Add Product View

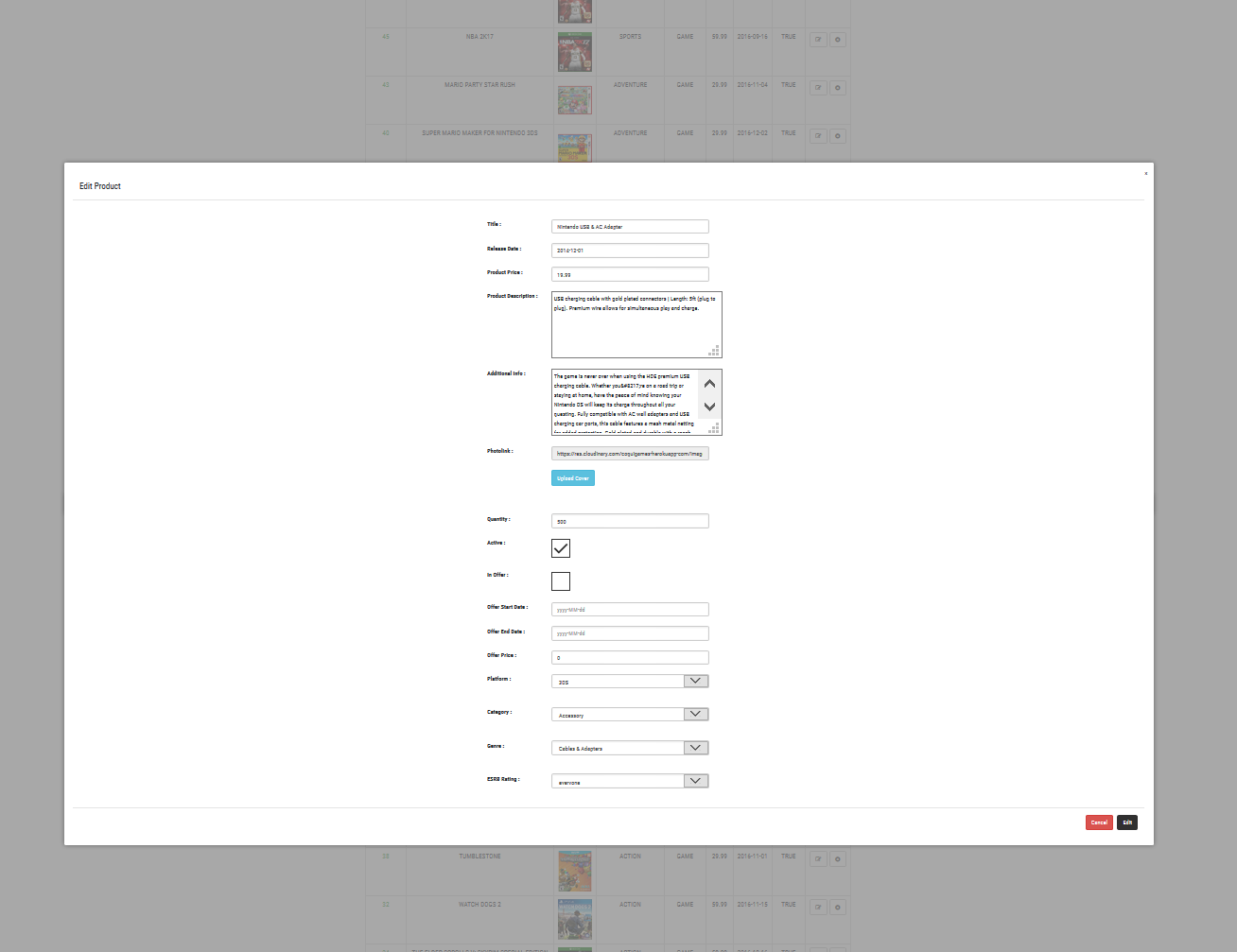


Figure 26: Products Table Edit Product View

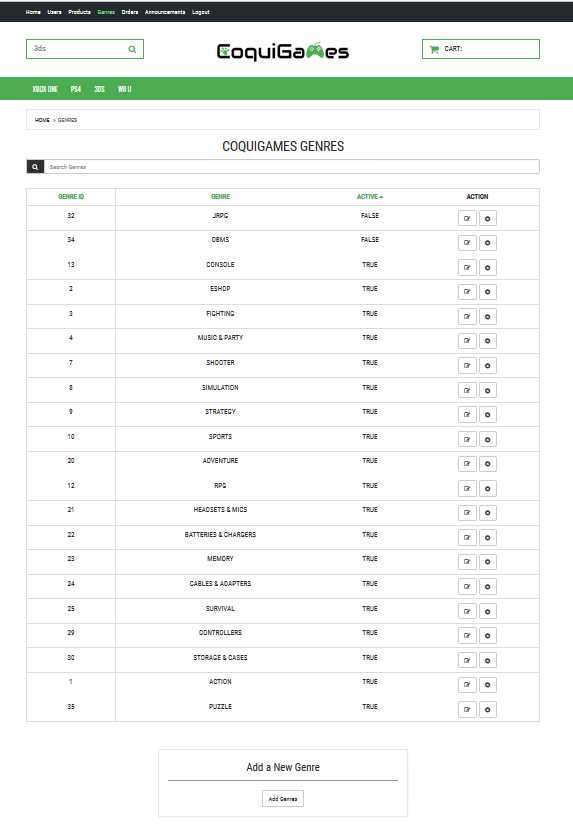


Figure 27: Genres Table View

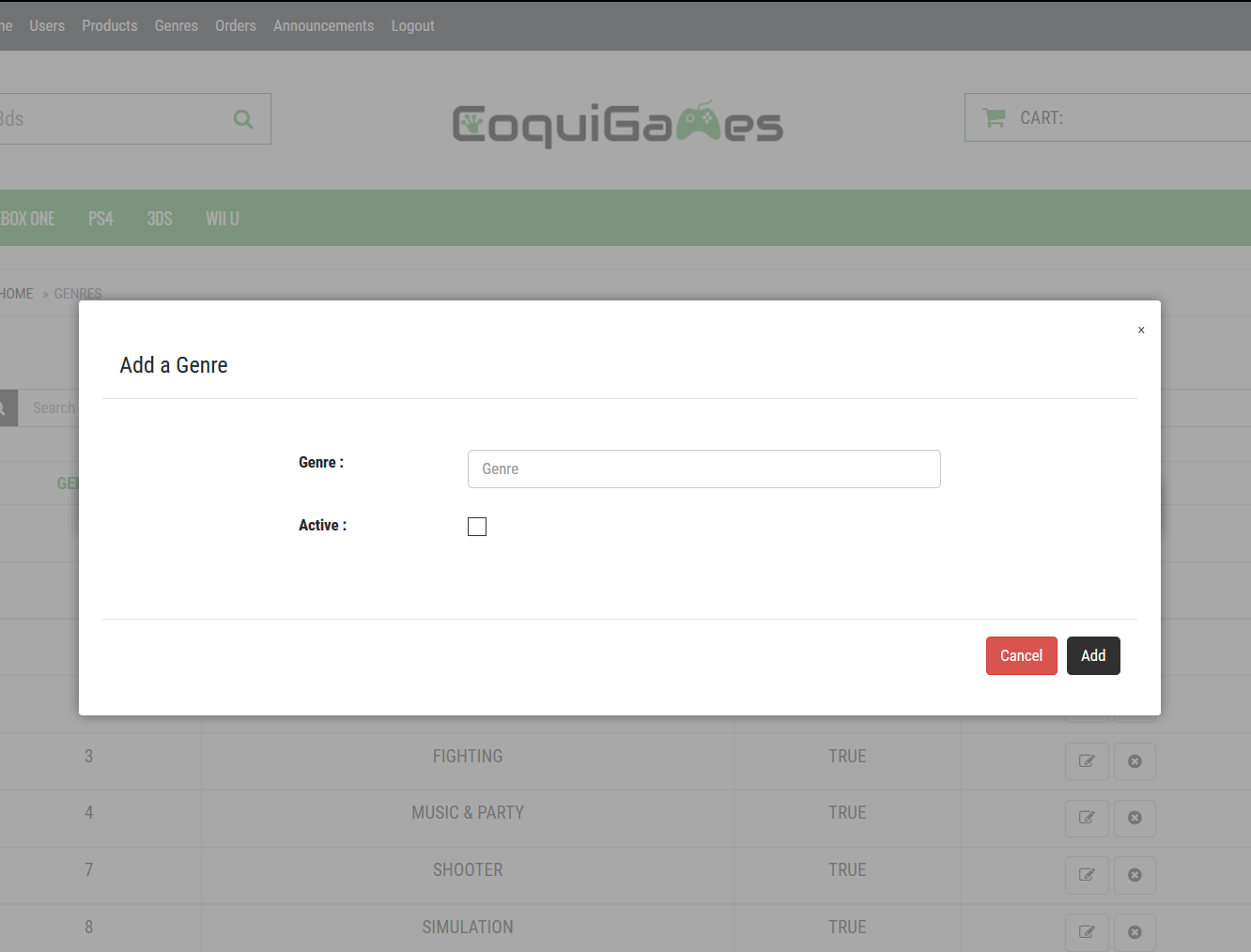


Figure 28: Genres Table Add Genre View

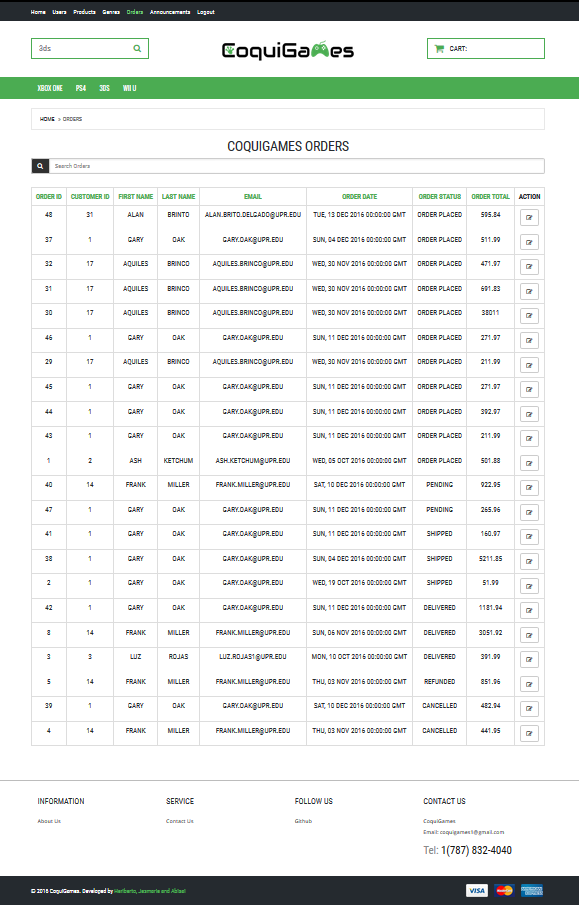


Figure 29: Orders Table View

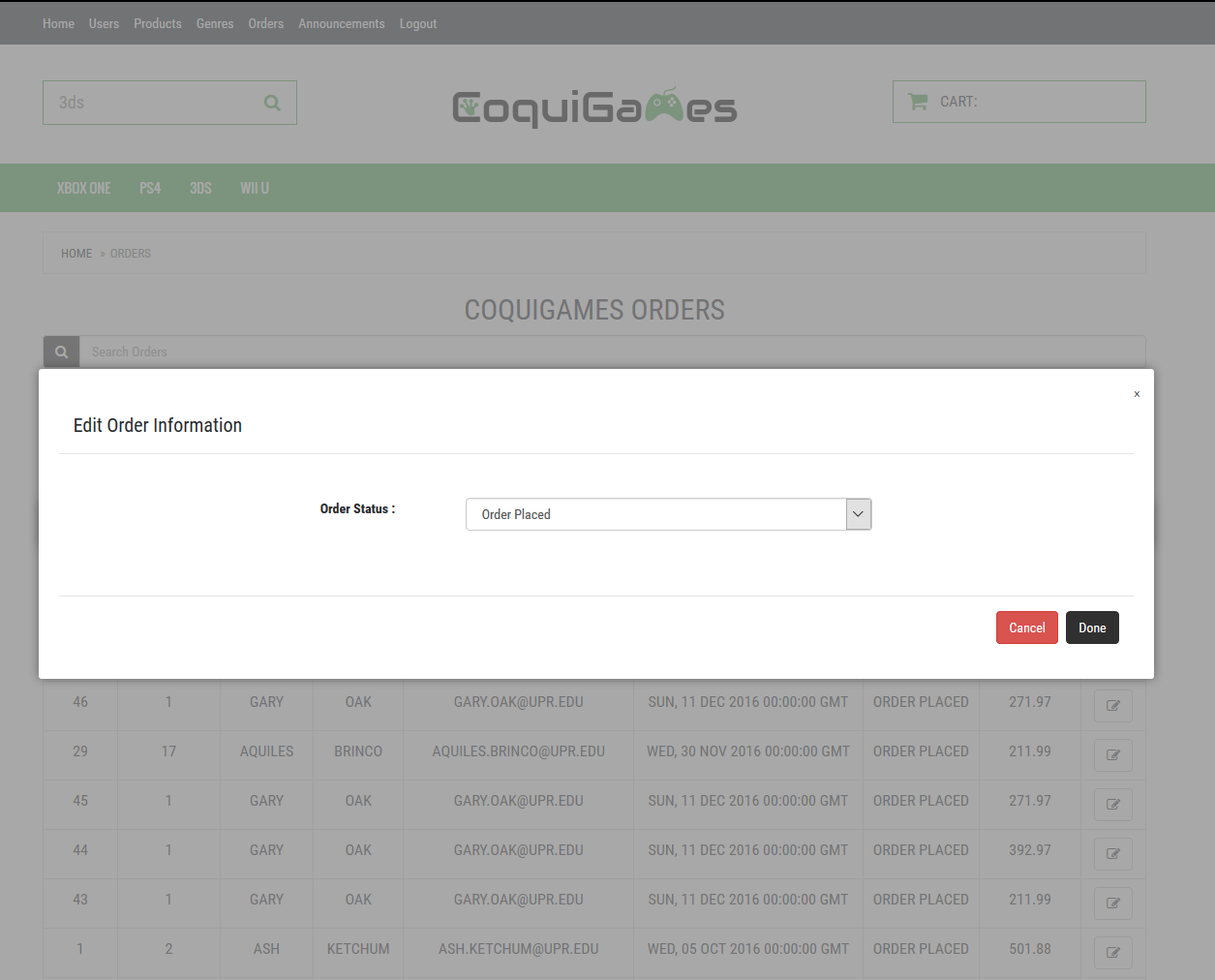


Figure 30: Orders Table Edit Order View



Figure 31: Announcements Table View

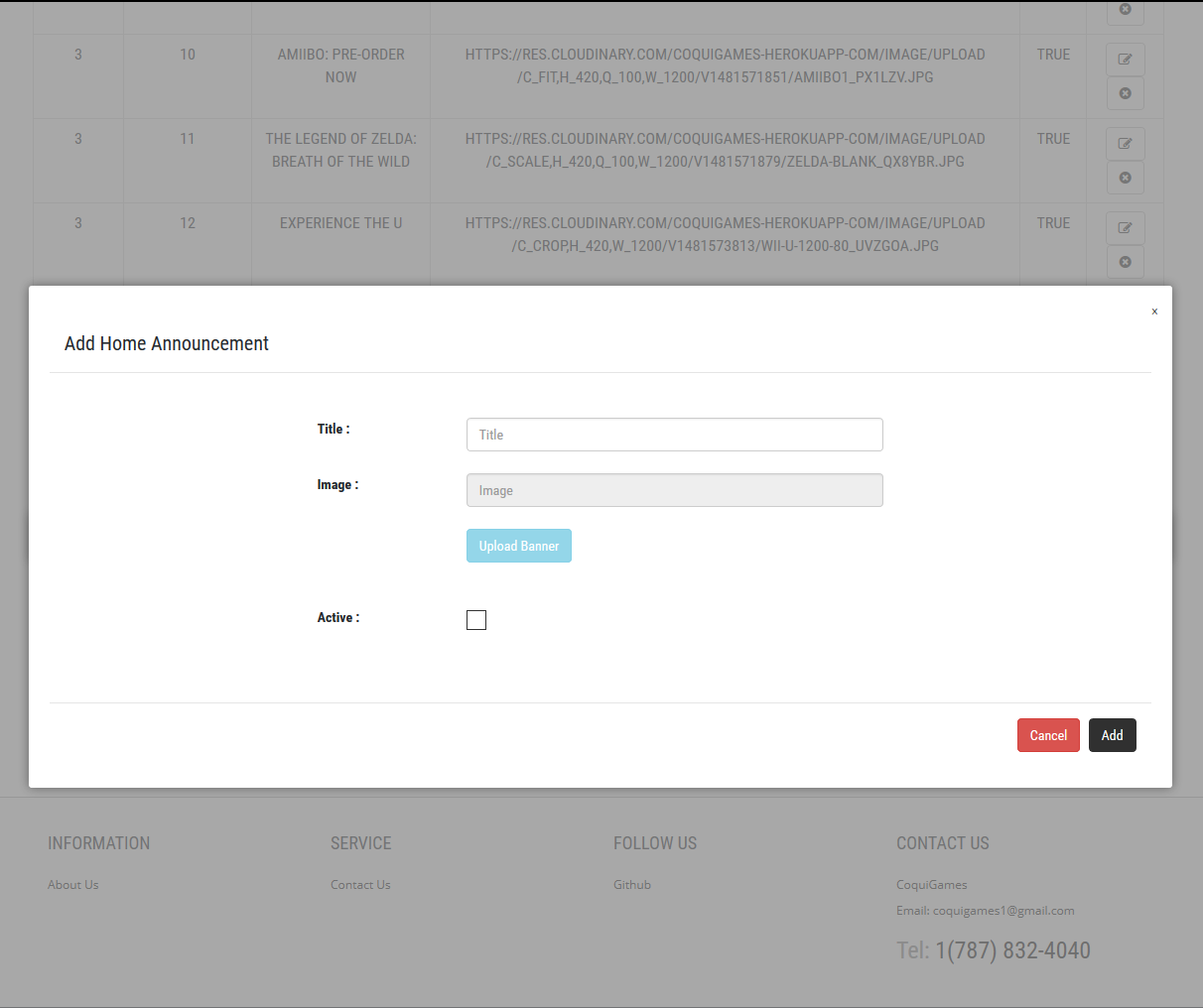


Figure 32: Announcements Table Add Home Announcement View

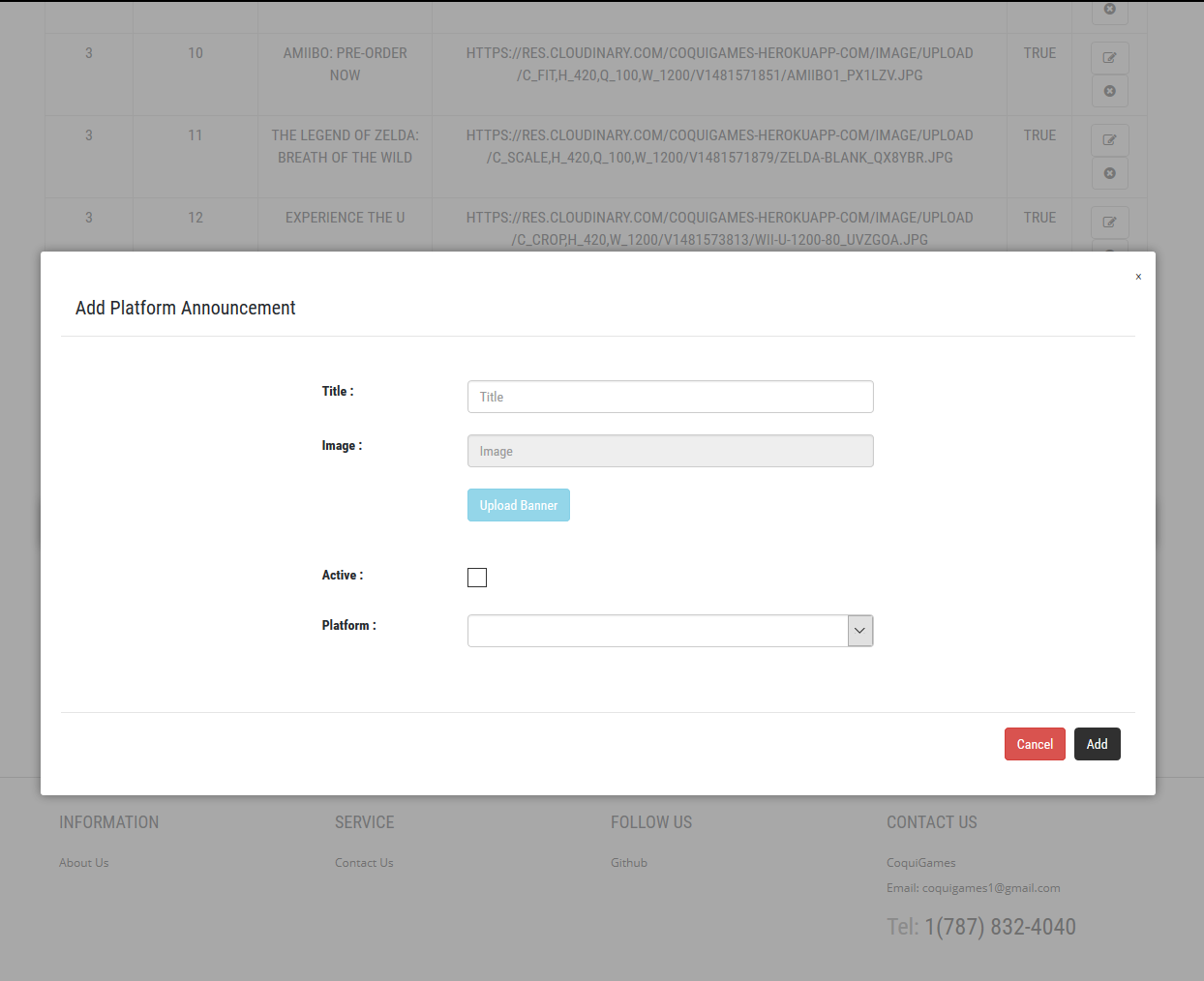


Figure 33: Announcements Table Add Platform Announcement View

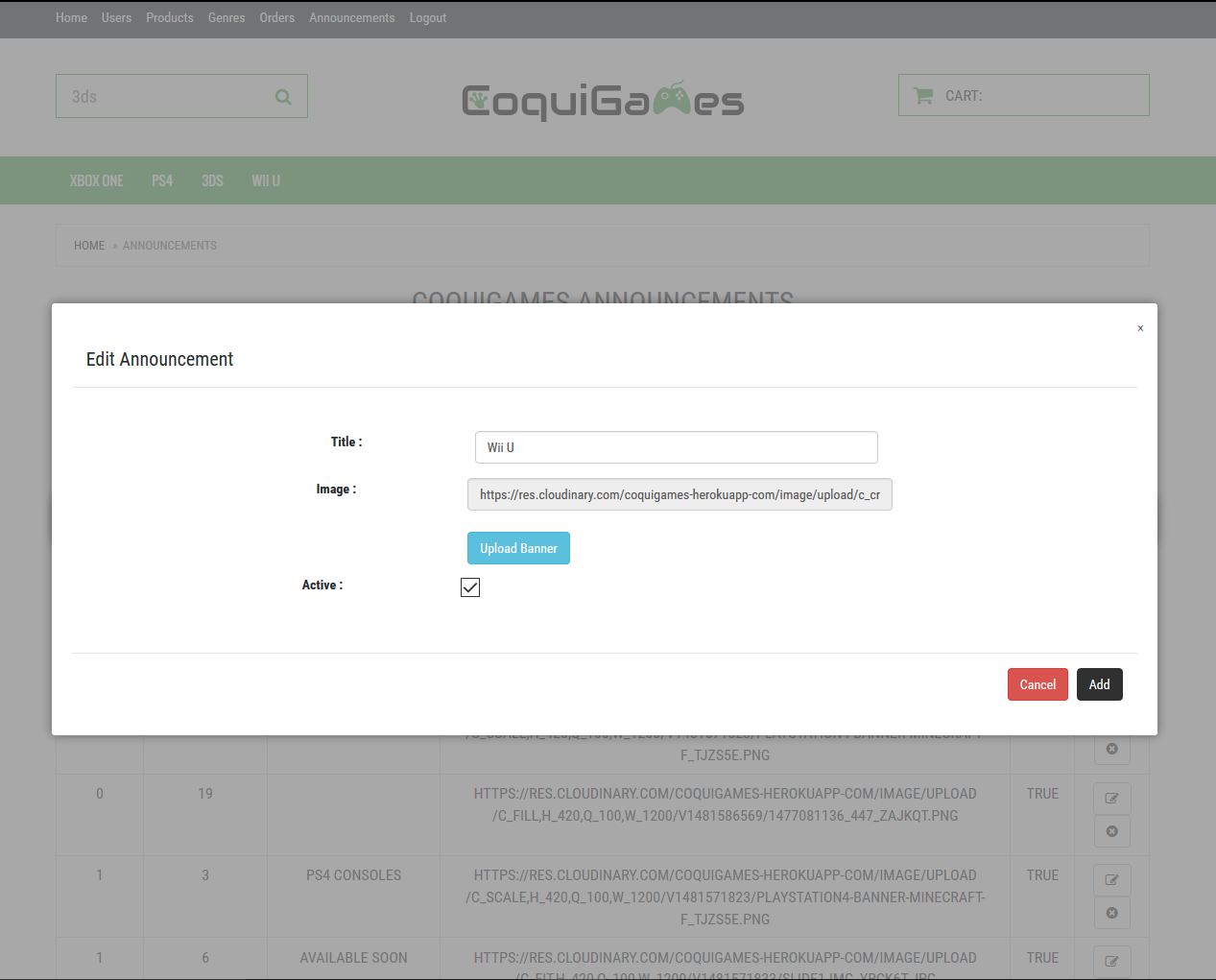


Figure 34: Announcements Table Edit Announcement View