# SoftUni Game Store – 5 March 2017

The game store is a digital platform where the users can buy games. Your task is to create a web application that has back-end and front-end logic using the provided resources (**TomEE**, **MVC Framework**, **Bootstrap** and views). You are free to edit those provided resources to fit your needs.

## Data Models

Create the required **entities**. Use the appropriate **data types**.

* The system contains information about **users**, and **games**
* Users can **register** in the system. After successful registration, the user has **email, password, full name, list of games** and information whether he is an **administrator or not**.
* The **first registered user** becomes also an **administrator**. You can manually mark users as admins in the database.
* A **game** has **title**, **trailer** (YouTube Video Id), **image thumbnail** (URL), **size**, **price, description** and **release date**

## Functionality

* **All users** can view the home page.
* **All users** can view details page of each game
* **All users** can add/remove games from their shopping cart
* **Guests** (anonymous users) can register an account with their own email and password
* **Guests** can login by email and password.
* **Logged-in users** can logout.
* **Logged in users** can buy games that are added to the shopping cart and those games are added to the profile of the user and cannot be bought for second time
* **Administrators** can add, edit or delete games
* When guest user tries to access a page that is allowed only for logged in user he should be **redirected to the login page**
* When user tries to access the administrator pages he must be redirected to the home page.

## Design the Database

Design **entity classes** and create a **database** to hold the **users**, **games**

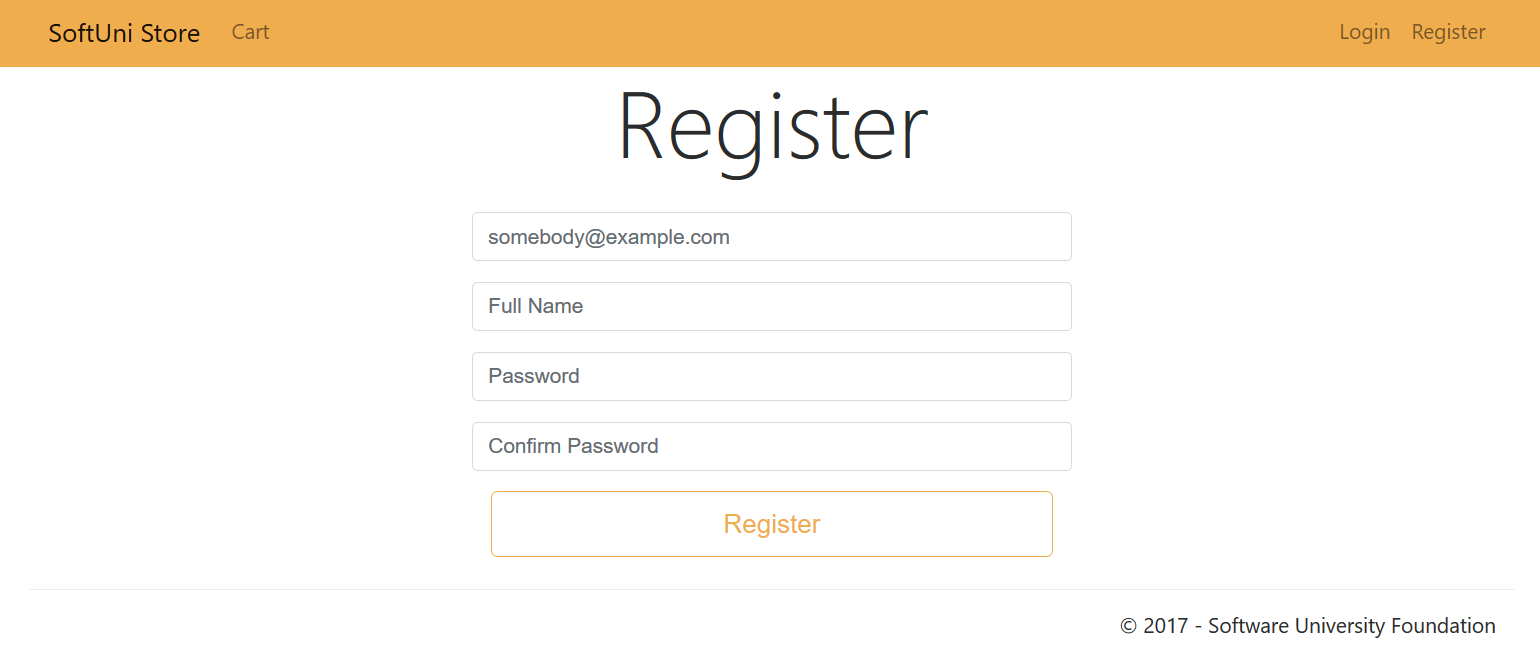
Score: **10 points**

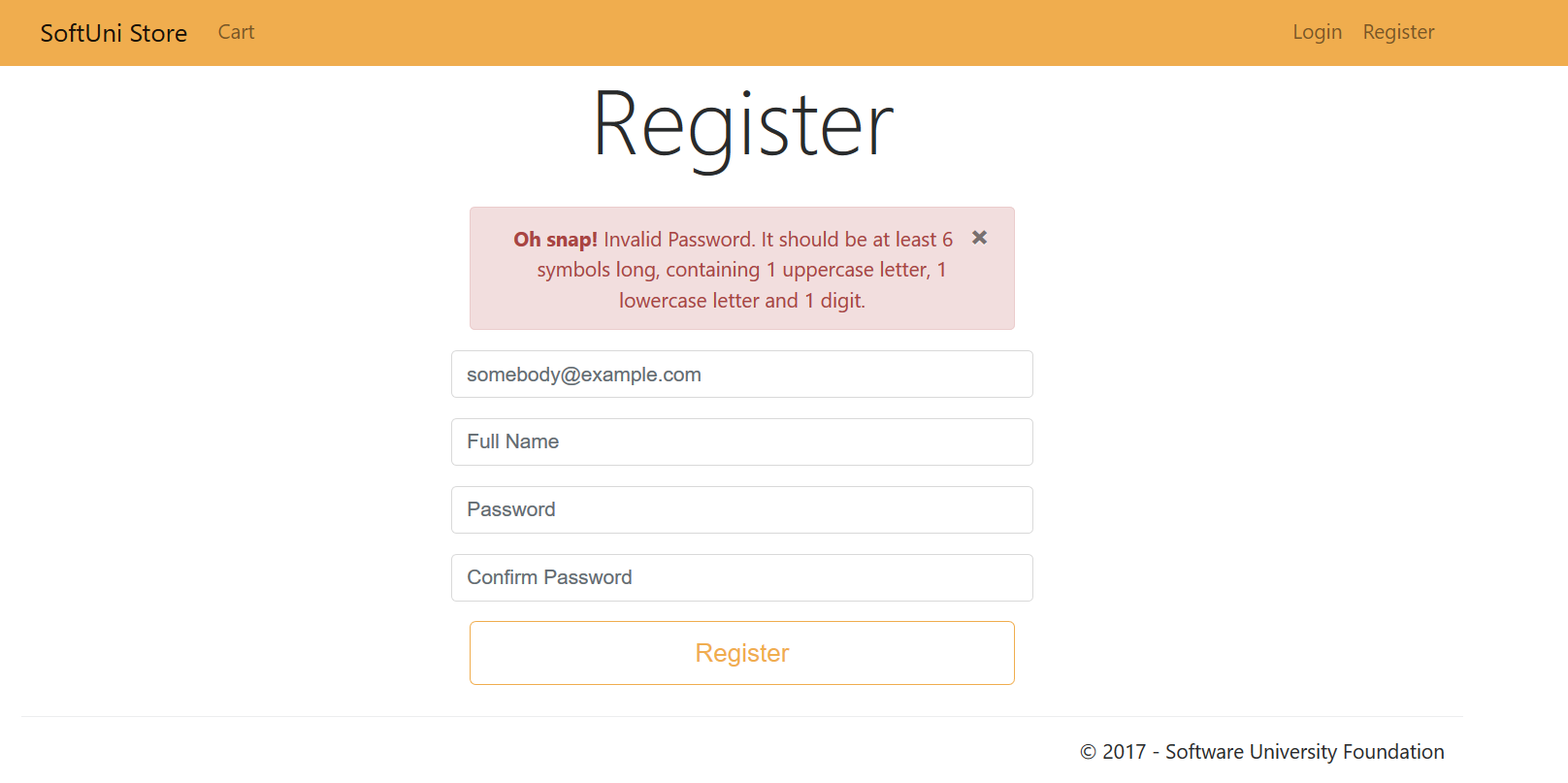
## Implement User Registration, Login and Logout

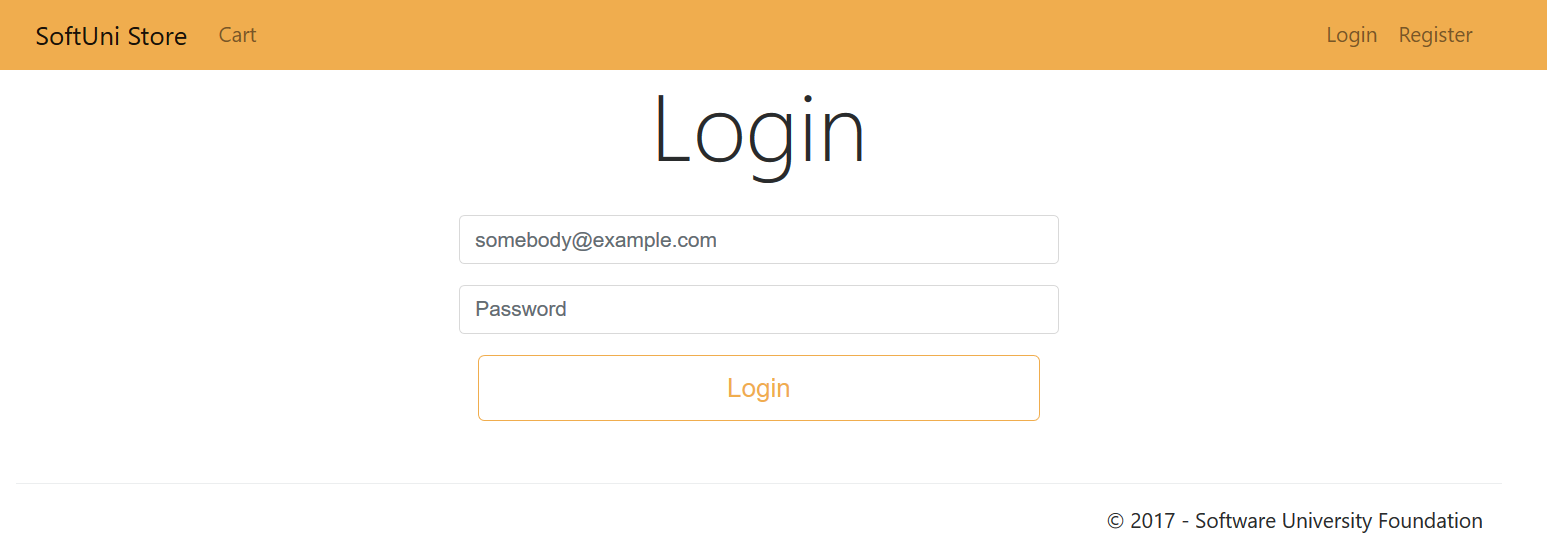
The guest users can register and log in the forum. He can provide to:

* **Register**
  + **Email** – must contain **@** **sign** and a **period**. It must be unique.
  + **Password** – length must be **at least 6 symbols** and must contain at least **1 uppercase**, **1 lowercase** letter and **1 digit**
  + **Confirm Password** – must **match** the provided password
  + **Full Name**
* **Login**
  + User can log in with **email** and **password**
* **Logout**
  + When logged in the user should have option to **log out**
  + The **Logout** mustn't clean the cart!

After register, the user should be redirected to the login page. If there are any errors with the registration, they should be displayed in the register form. All validations should be performed on the back-end.







Each type of user must have **unique navigation bar**.

Guest:



User:

https://puu.sh/uqTCg/8896fd0802.png

Admin: https://puu.sh/uqTFc/dfab16beaa.png

Score: **17 points**

## Implement Managing Games

As an admin, you have the option to **add games to the catalog**. A game should be added only to the catalog if matches those criteria:

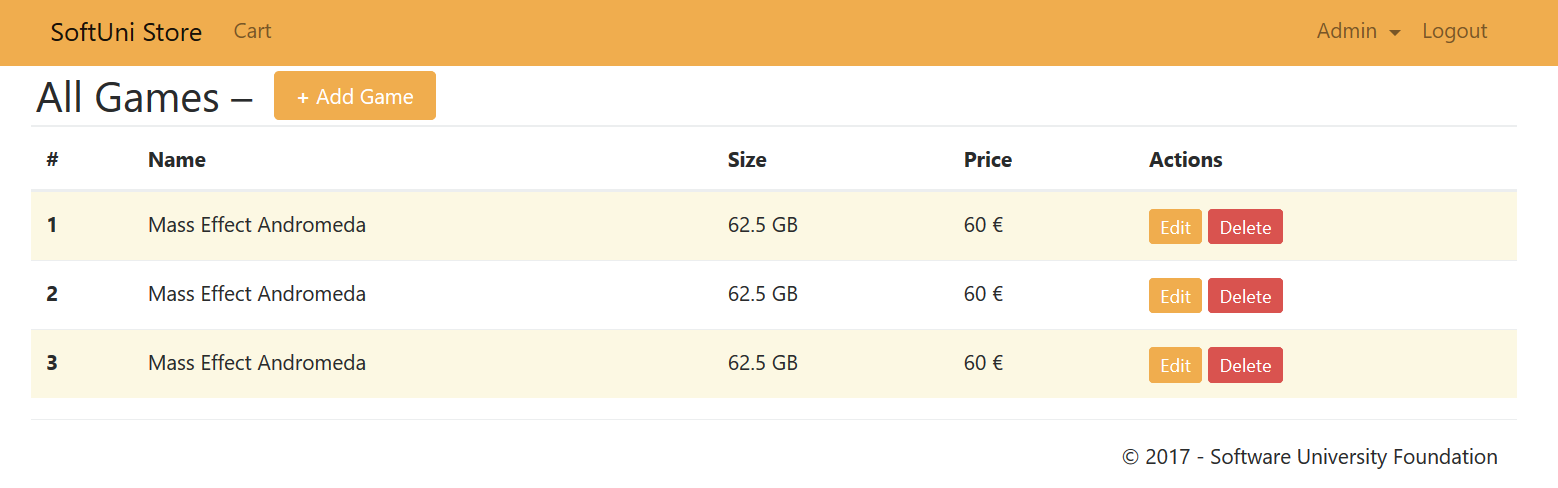
* **Title** – has to begin with **uppercase letter** and has length between **3 and 100 symbols** (inclusive)
* **Price** – must be a **positive number** with precision up to **2 digits** after floating point
* **Size** – must be a **positive number** with precision up to **1 digit** after floating point
* **Trailer**– only videos from YouTube are allowed and only their **ID** should be saved to the database which is a string of exactly **11 characters**.

For example, if the URL to the trailer is https://www.youtube.com/watch?v=edYCtaNueQY, the required part that must be saved into the database is edYCtaNueQY. That would be always the last 11 characters from the provided URL.

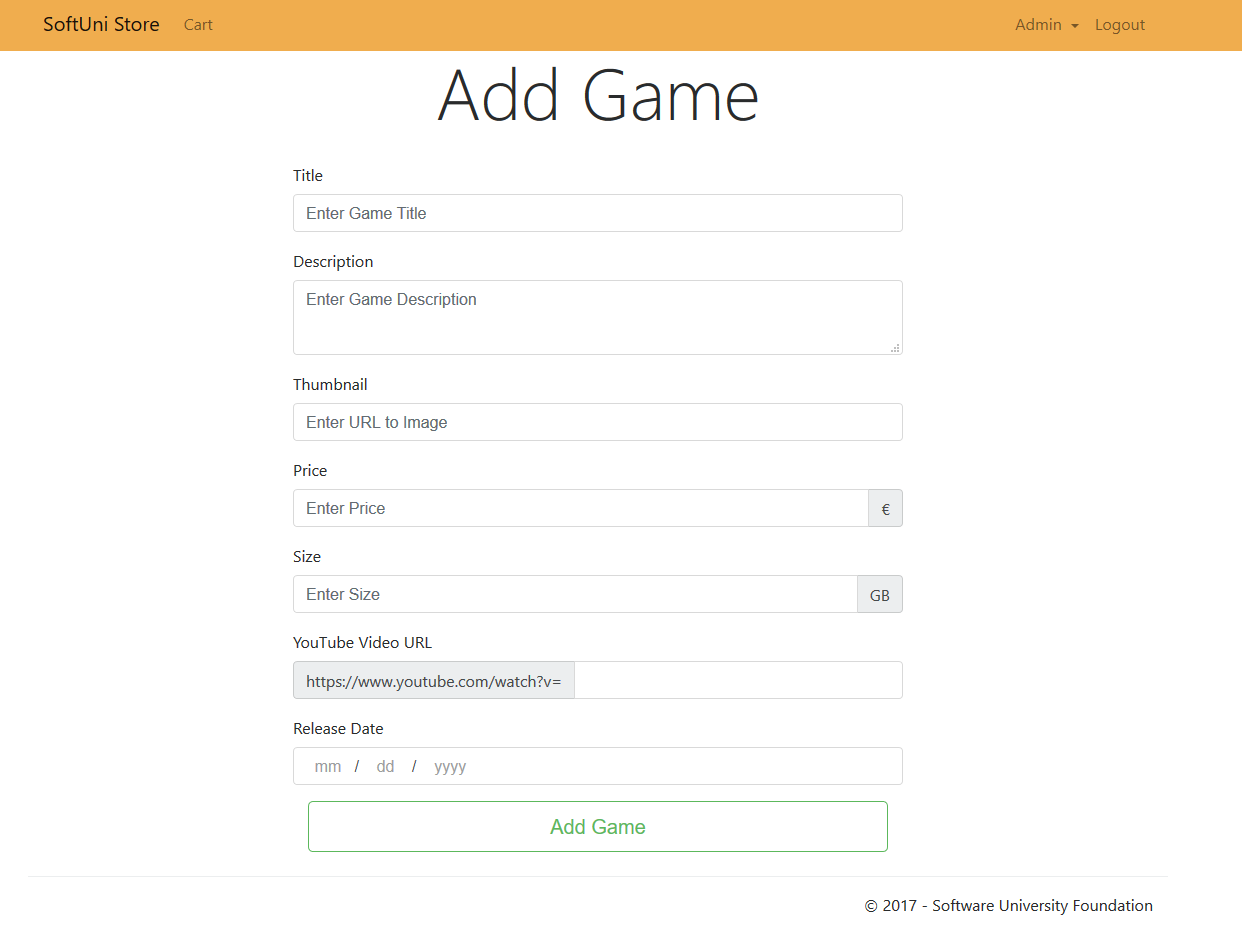
* **Thumbnail** **URL** – it should be a plain text starting with **http://**, **https://** or **null**
* **Description –** must be at least 20 symbols

All validations must be performed on the back-end.

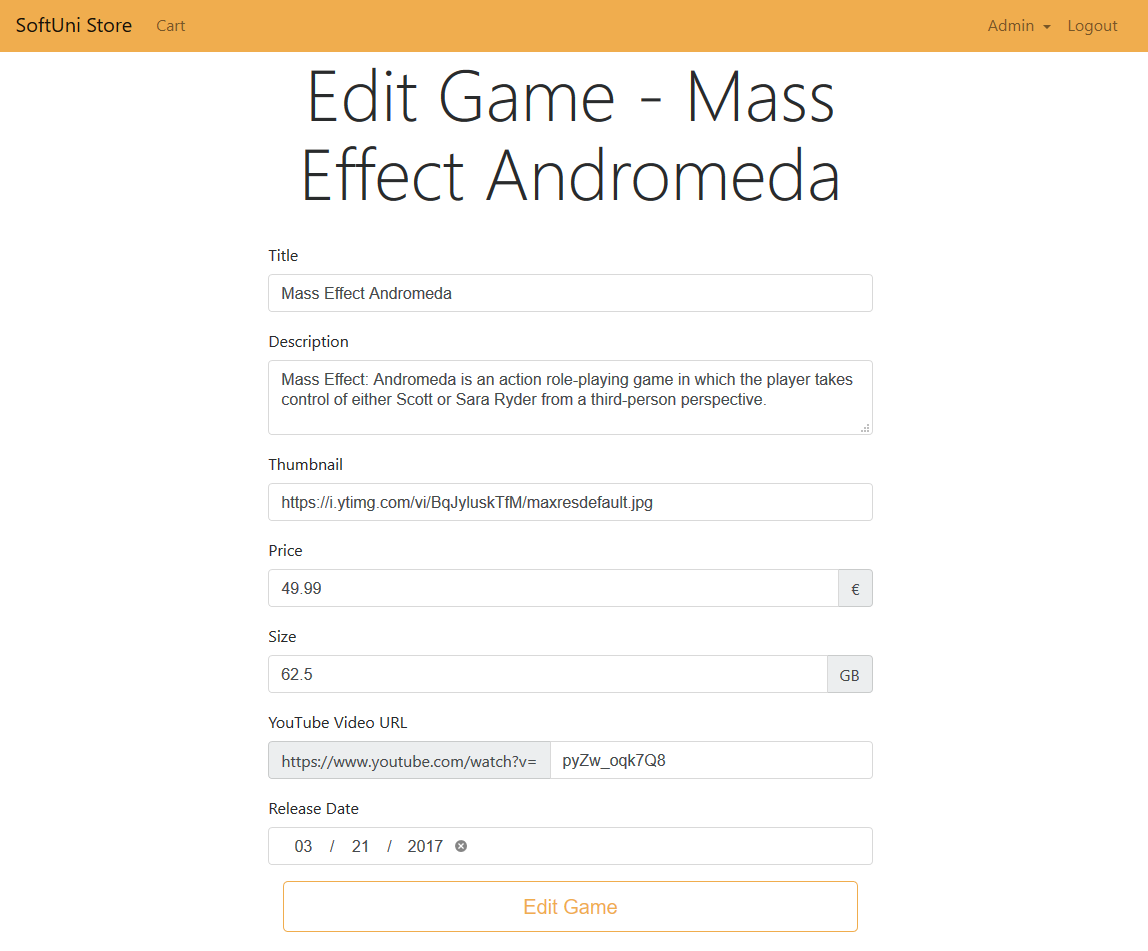
Listing all games:



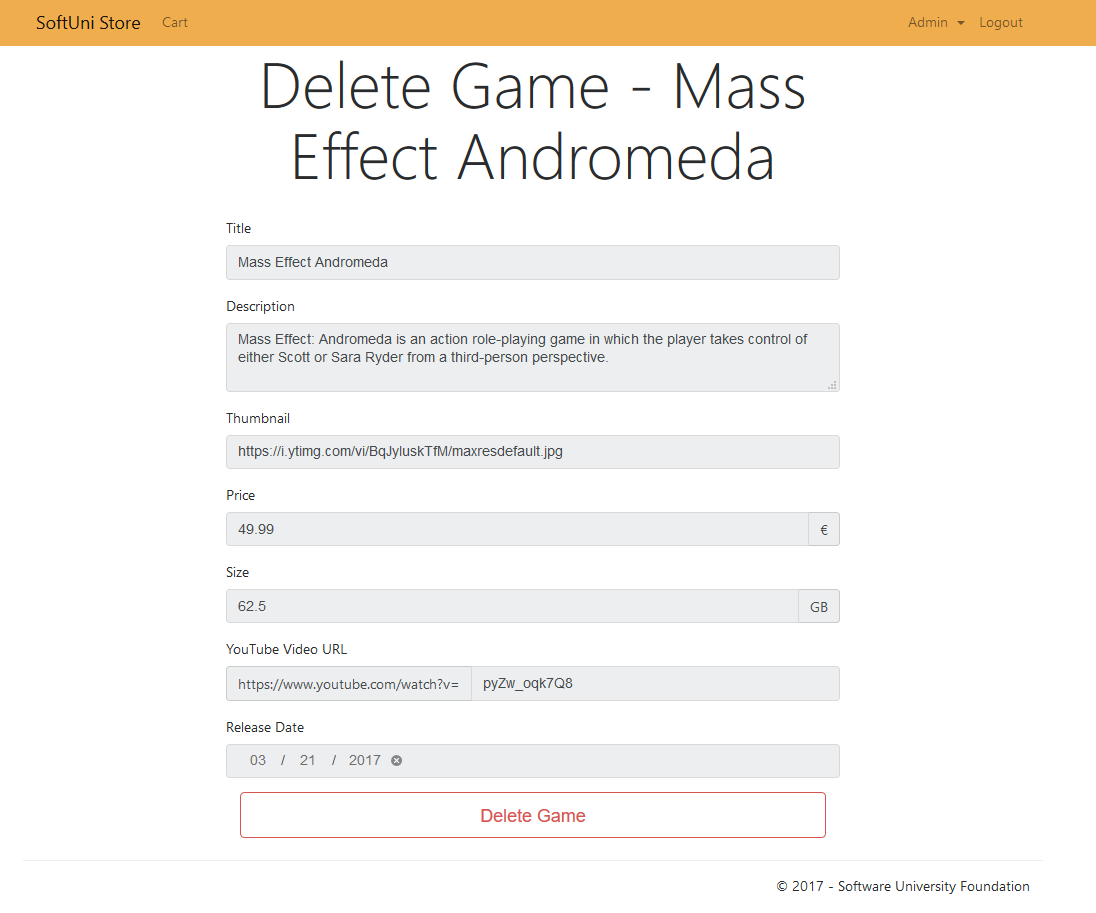
Adding a game:



Editing a game:



Deleting a game:



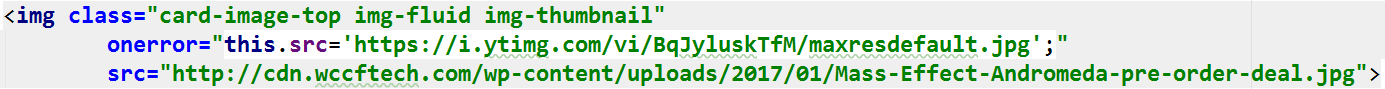
Score: **27 points**

## Implement Home Page

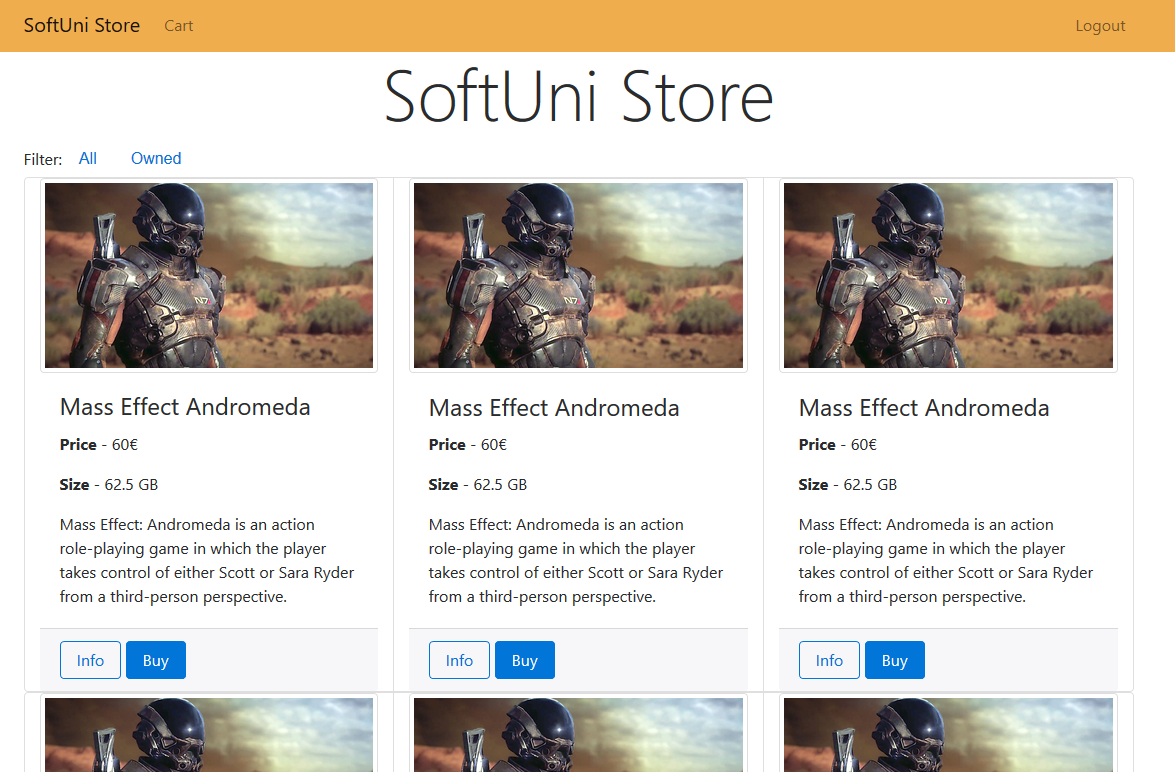
Home page should be accessible for **all users**. It should show a list of **all games** (thumbnail, title, price, size, description) with option to each game to **buy** it and see more **detailed information** for the game. Also, there should be options to **filter** the games:

* All Games
* Owned Games

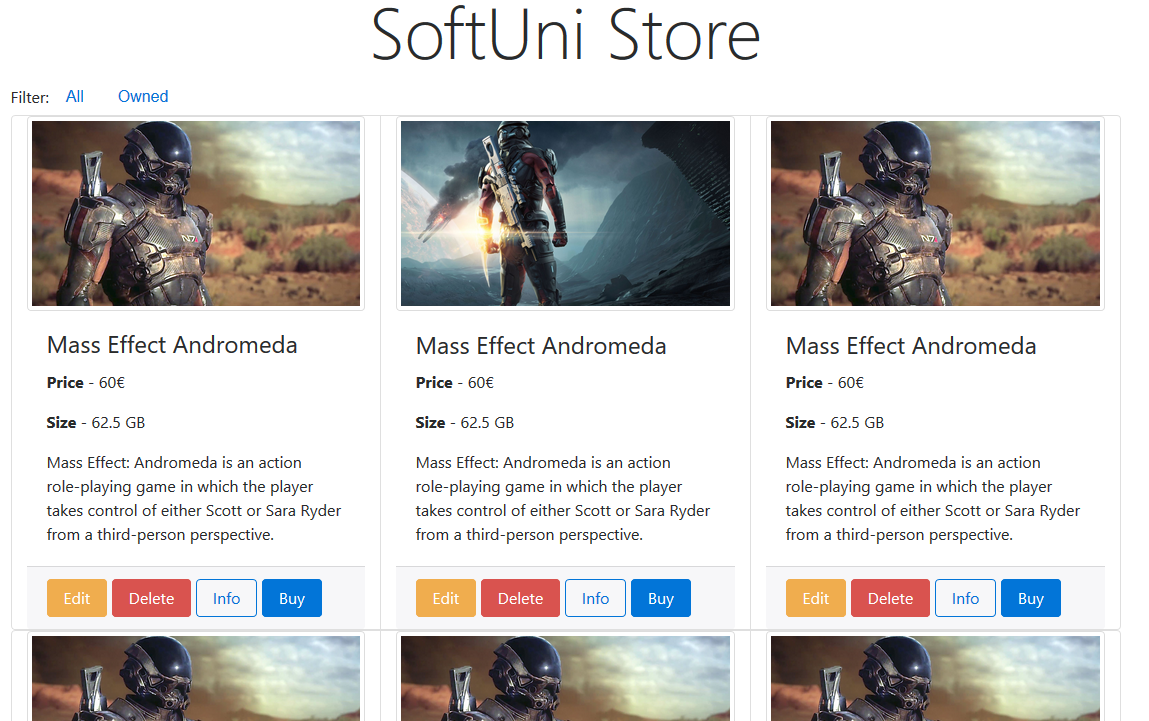
Only the first **300 characters** of the description for each game must be showed on the home page. There should be no more than **3 games per row**. If the game thumbnail is **null**, you should use the onerror attribute. You need to use the following link "https://i.ytimg.com/vi/**{YouTube Video Id}**/maxresdefault.jpg". Example:



**Guest** and **user** home page:



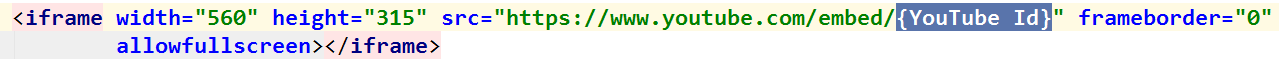
Admin home page:



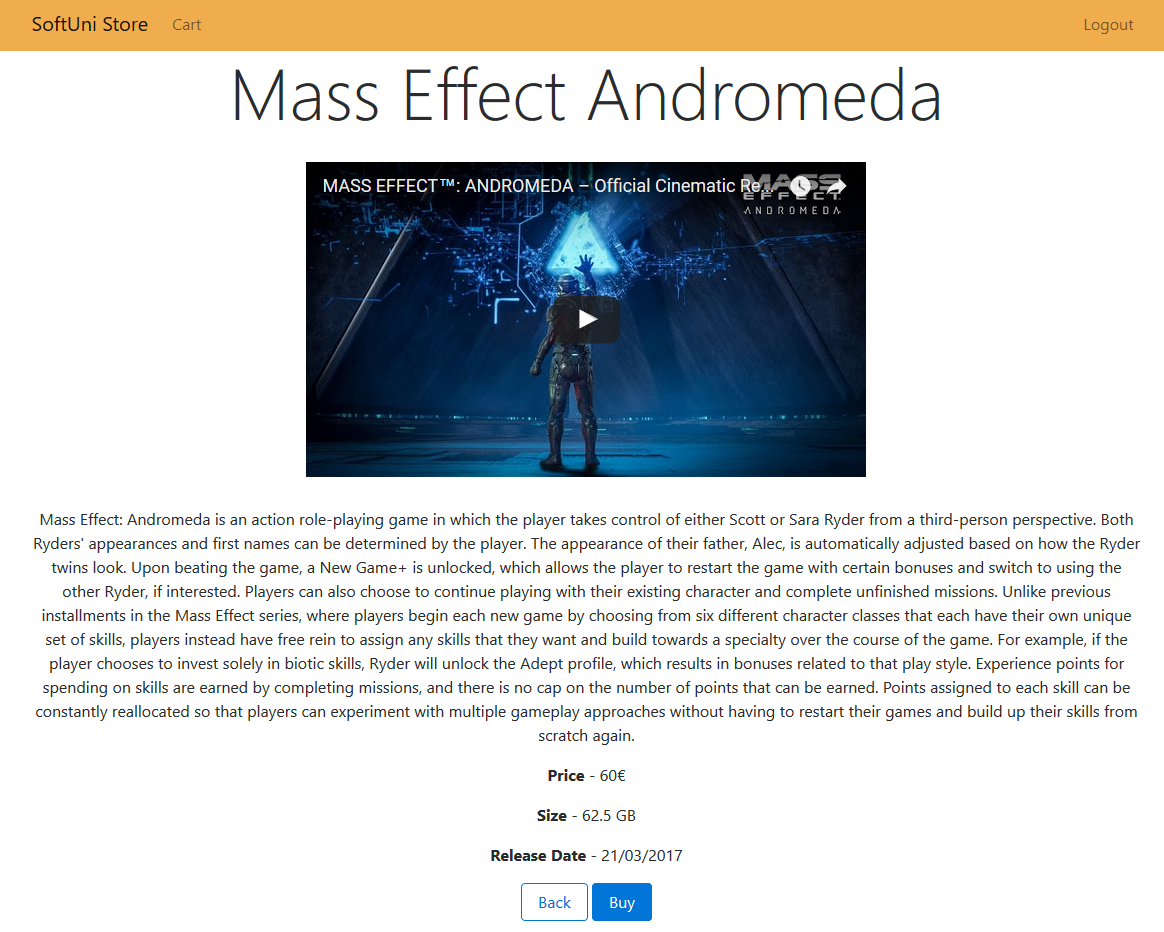
Score: **20 points**

## Implement Game Details Page

Each game should have detailed page where the users can see **all details** for that game. This includes an embedded YouTube video. You can do it, using the YouTube Video Id, that you keep for each game. You are given the following template:



You need to replace the placeholder with the actual video id.

Guests and users should see this: 

Admins should see this:



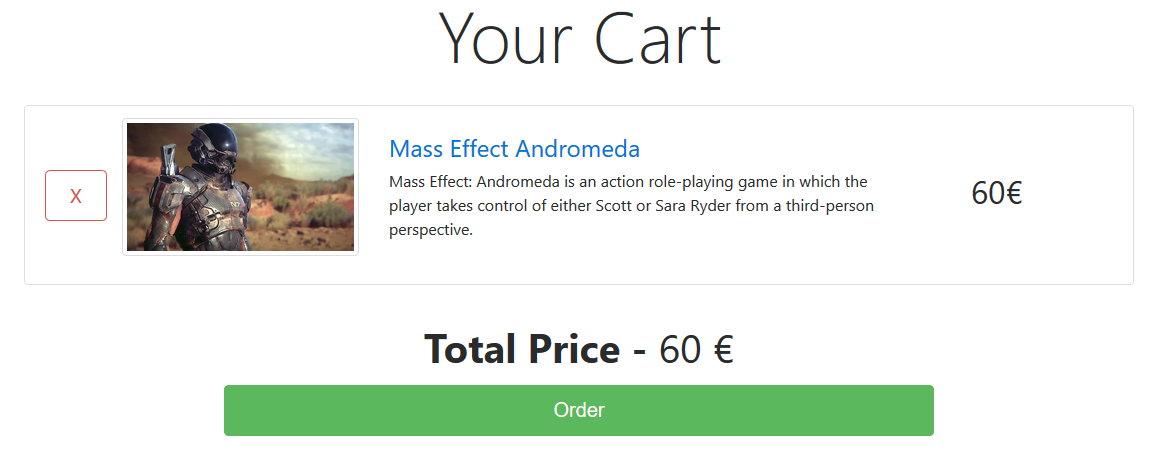


Score: **6 points**

## Implement Shopping Cart

Each user should be able to buy game. To buy a game the user has to:

* Click on the **buy button** for the desired game
* The game is now **added to a shopping cart**
* The user may **add more games** if he wants
* When he is ready to buy them all he needs to go to the cart page where there is a list of all added games with option to **remove** any if he decides he does not want to **buy it.**
* To buy the games he should click on the **Order button**. If he is not logged in, he must be redirected to the **login page** and he must log in. When is logged in the games should be added to his profile and he can see them on the home page.
* A user can **buy** a game **only** **once**!
* If he owns a game, he **shouldn't be able to add** it to the shopping cart.



Score: **20 points**

## Project Infrastructure Bonus

Bonus points code quality / good application structure / additional effort.

* Bonus points for implementing **separate data layer**.
* Bonus points for using **dependency inversion**.
* Bonus points for using **AutoMapper**.
* Bonus points for using **services** for different jobs.
* Bonus points for implementing **Repository** or **Unit of Work** pattern

Bonus Score: **20 points**