# Teamwork Project Assignment for the [Web Development Basics @ SoftUni](https://softuni.bg/trainings/1211/Web-Development-Basics-September-2015)

Design and implement a **Shopping cart/CMS/RPG Game** **using PHP and HTML / CSS / JavaScript**. Your project must meet all the requirements listed below.

## Requirements

* **Use PHP** – the major part of your work should be PHP written
  + You should create custom PHP Framework
  + You should create the project over your framework
  + You must additionallyuse **HTML5, CSS3** to create the content and to stylize your web application
  + You may optionally use **JavaScript, jQuery, Bootstrap**
  + Use **PHP 5 or higher (7 is a bonus)**
* **User source control system**
  + **Use GitHub** or other source control systemas project collaboration platform
* Publish your project **live in Internet** – your project should be public in Internet
  + You may **share your project** to get external feedback
  + Most shared and commented projects will get additional **bonus score**
* **Valid and high-quality PHP, HTML and CSS**
  + Follow the best practices fr PHP development: <http://www.phptherightway.com>.
  + Validate (when possible) your HTML (<http://validator.w3.org>) and CSS code (<http://css-validator.org>)
  + Follow the best practices for **high-quality PHP, HTML and CSS**: good formatting, good code structure, consistent naming etc.
* **Usability, UX and browser support**
  + Your web application should be easy-to-use, with intuitive UI, with good usability
  + Ensure your web application works correctly in the latest HTML5-compatible browsers: Chrome, Firefox, IE, Opera, Safari (latest versions, desktop and mobile versions)
  + You do not need to support old browsers like IE9
* **Framework requirements**
  + **ABSOLUTELY REQUIRED** (without those ones the project is invalidated)
    - Default routing system
    - Custom routing system
      * Easy way to define custom routes with parameters and parameter types
    - Areas
      * Custom routing for areas; It should override the global routing configuration
    - Strongly typed views
    - POST request actions using Binding Models
    - ViewHelpers for Forms, Ajax submitted Forms, CSRF Token
      * Form ViewHelper
        + Radio buttons
        + Checkboxes
        + Dropdowns
        + Text fields
        + Password fields
        + Textareas
    - Out of the box escaping the html chars unless stated otherwise
  + **Other**
    - Data annotations
      * @Authorize – the method is not called if the user is not logged in
      * @Admin – the method is not called if the user is not logged and is not administrator
    - Overriding default route config with annotations
      * @POST – the method can be called only on POST
      * @PUT – the method can be called only on PUT
      * @DELETE – the method can be called only on DELETE
      * @GET – can be omitted. If declared or not, the method is called only on GET
      * @Route(“route/here”) – declares the route action is called upon
    - Dependency injection configuration (config file or annotation)
    - User roles and config/annotation for them (method is not called if user is not in the role)
    - Upload files ViewHelper
    - Help page that scans all routes (configured, annotated and basic ones) and shows them (e.g. PUT users/{id}). Opening the route shows the serialized Binding Model that accepts if any or primitive parameters if any. (e.g. {name: name, email: email}, or ?id={id}&name={name})

## Forbidden Techniques and Tools

* Using **CMS / blog systems** (like WordPress, Drupal and Joomla) is forbidden.
* Using S**hopping cart systems** (like OpenCart) is forbidden.
* Using **Laravel**, **Zend Framework**, **CakePHP** or other PHP MVC Framework is forbidden.

## Projects

Please choose one of the projects below.

### Shopping cart

**Required** functionalities:

* User registration / login and user profiles.
* User roles (user, administrator, editor)
* Initial cash for users
* Product categories
* Listing products in categories
* Add to cart functionality
* Promotions for certain time interval
  + Promotions on certain products (% discount)
  + Promotions on all products (% discount)
  + Promotions on certain categories (% discount)
  + Promotions for certain user criteria (registered more than X days, have more than X cash, etc…)
  + If two or more promotions collide on a date period for certain product – the biggest one applies only
* Visibility only of available products
* Quantity visibility
* Checkout the cart
* View cart
* Users can sell bought products
* Editors can add/delete product categories
* Editors can add/delete products
* Editors can move products between categories
* Editors can change quantities
* Editors can reorder products
* Administrators have full access on products, categories, users and their possessions

**Bonus** functionalities

* Managing the cart
* Users can sell products and put them promotions
* Users can make comments on products (review)
* Administrators: ban users
* Administrators: ban IP’s

### Content Management System

**Required** functionalities:

* User registration / login and user profiles.
* User roles (user, editor)
* The project should be separated by two parts – frontend and backend
  + The frontend part is the part which the normal users could see
  + The backend part is the part which the editors and administrators could see
* Backend:
  + There must be a way the Editor to change the site background – either color or picture
  + The Editor can create different pages/page names (e.g. home page, contacts page, etc..)
  + The Editor can position different elements on the site/page
    - Grids
    - Forms (text fields, checkboxes, radio buttons, text areas, dropdowns, buttons)
    - Anchors
    - Paragraphs
  + The Editor can attach events to these elements (e.g. what to happen when form posts)
  + There have to be different predefined events (mail sending, db storage, comments (the posted form content could be later visible somewhere in the site)
  + The Editor can extract information from db and put it in the frontend (e.g. the form posting content)
* Frontend:
  + The User can see anything present in the frontend and use its functionalities

**Bonus** functionalities:

* Administration:
  + Administrators is and editor with more privileges.
  + Administrators can edit users as well as roles – choose which user/role what to see on frontend e.g. contact form not visible by guests
  + Administrators can ban users
  + Administrators can ban IP’s

### RPG Game

**Required** functionalities

* User registration / login and user profiles.
* When registered users are placed on a map (2D [x:y], 3D [x:y:z] or something else by your choice)
* Depends on your game story, the users start with more than one platforms (kingdoms, planets, houses, whatever the story is about)
* On each platform the user starts with predefined amount of resources (the game should have at least two resources for example **gold** and **food** / **metal** and **mineral** / etc…)
* Each user’s platform has something the user can evolve for resources. At least two of them should give income per hour of resources (e.g. buildings, one of them is **Gold** **mine**, another is **Farm,** the mine gives income per hour of resource Gold, the Farm gives income per hour of resource Food)
* The income per hour should be added to the platform’s resources on each 2 minutes (1/30 of the income) e.g. if user clicks each second on the web page and has 3000 gold per hour, on each 2 minutes one will receive 100 gold. If the user does not click for example 20 minutes and refreshes the page should receive 20/2 \* 100 = 1000 gold. If the user does not click for example 3 hours and refreshes the page will receive 3 \* 3000 = 9000 gold.
* When building is started to build, it should take some time (each level takes more time). The time should be visualized with a countdown timer. When the building is ready it should take effect.
* On each platform the user can build different army units (e.g. cavalry, fleet, seaships, etc…). Each unit should have dependency on certain building levels and combination of them (e.g. unit X needs 3rd level of dockyard and 8th level of Gas mine)
* Each unit should cost different amount of resources
* User is able to input how much units of each type wants to build
* When units are in building process they take time (e.g. unit X takes 3 mins per unit and unit Y takes 8 mins per unit, user builds 10xUnitX and 8xUnitY, after 30 mins all the UnitX will be ready and UnitY will have 34 minutes time remaining; or you may implement it on each 3 mins one X is ready and on each 8 mins one Y is ready, it was just a hint)
* User is able to attack another user with its units. The user choses how many of each unit type to send to another user. Each army journey from one player’s platform to another player’s platform takes time depending on the coordinate distance (e.g. from [2:12] to [8:6]). Both users can see informing message in their homepage. The aggressor sees who is attacking and the victim seems who attacks him. Both see the time remaining until the impact.
* When the army journey reaches the hostile platform a battle happens. Each unit should have some kind of statistics in order to make a battle e.g. UnitX is weaker than UnitY so 200xUnitX against 100xUnitY results in loss of UnitX and UnitY remains with 40.
* There is a battle report visible for both sides with the result of the battle. Who won, who loss or if the battle is draw. How many units both sides left with after the battle.
* If there are left units of the attacker, a backwards journey is made so the army is coming home
* When the backwards journey ends, the army gets down on the users platform

**Optional** functionalities

* Messaging system between users
* Game messaging system – keeping battle reports
* Espionage – a unit for taking information regarding hostile platform. Very fast unit that for seconds reach the hostile platform and gives information about the hostile army units. There should be a chance of success depending certain building level (e.g. the one that takes information is with level 10, the other is with level 8, so the chance is 10 \* ((10 – 8) \* 2) = 40%)
* Battle system. Battle takes in X rounds (e.g. 10) each round there is a priority which unit, which hostile unit to attack and how to respond to attack.
* Cron jobs, so even the user is offline, income rises on 2 minutes, etc…

## Public Project Defense

Each student will have to deliver a **public defense** of its work in front of the SoftUni team. The students will have **only ~20 minutes** for the following:

* **Demonstrate** the web application (very shortly).
* Show the **source code** and explain how it works.

Be **well prepared** for presenting maximum of your work for minimum time. Open the project assets beforehand to save time.

## Assessment Criteria

* **Functionality** (all the required functionalities according to the type of project you choose) – **0…50**
* **Framework functionality – 0..70**
* **Overview** (HTML / CSS / Usability / UX) – **0…10**
* **Code quality** (correct naming, code formatting, separation of concerns, etc.) – **0…30**
* **Security** (XSS, SQL Injection, CSRF…) **– 0…25**
* **Bonus** (bonus point are given for implementing optional functionalities according to the type of project you choose) – **0..35**