# Exam – SoftUni Bazar

You need to create a **web application** where users **see and trade items**. Usually the users will visit the page, navigate either to **signing up or sign in**. When they are logged they will have the option to view **all items** listed. They can either **add** a new item, **view all their** itemsor **edit** or **delete** existing, only if **they** added it.

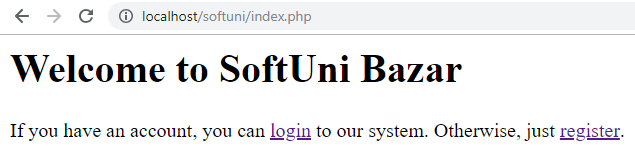
# 1. Expected Result

General rules:

1. Each **title** is **Heading 1**
2. All **attribute values and tags** are **lowercase**
3. All name fields are lower\_snake\_case variant of the text/value
4. All **tables** have **head** and if they are filled with data, they have a **body**
5. All **success messages** are stored in the **session** under the “success” key and are **immediately deleted** from session after visualization. They are visualized on the **next line after the title** in a **paragraph** with style color:green.
6. All **error messages** are visualized on the **next line after the title** in a **paragraph** with style color=red.

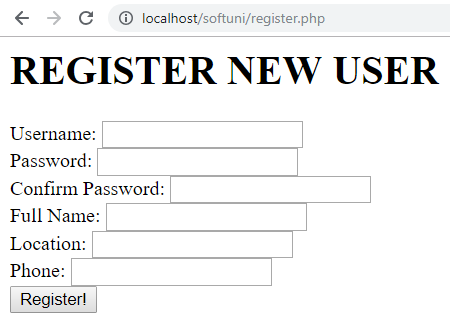
**index.php:**

* When user **is** **not** logged

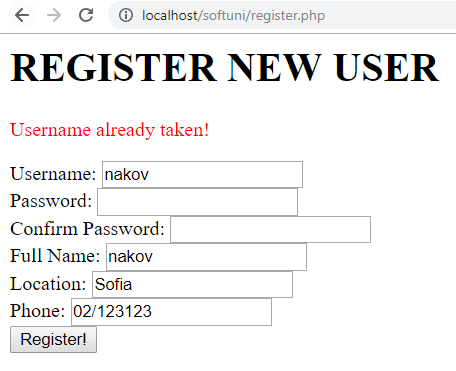


**register.php:**

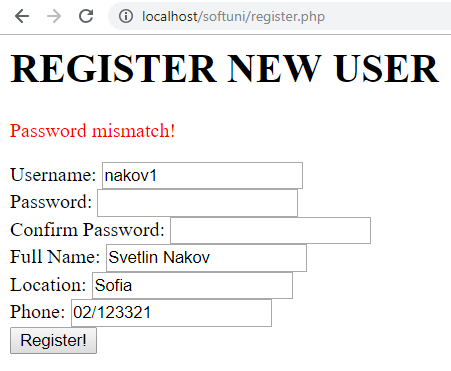
* **Before** form is submitted



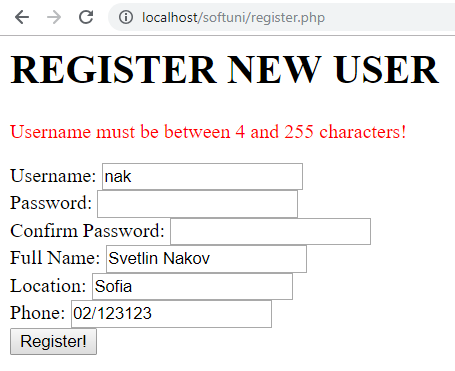
* Form **is submitted**, but there are **username** **duplicates**



* Form **is submitted**, but the **password** and the **confirm password** field differs:

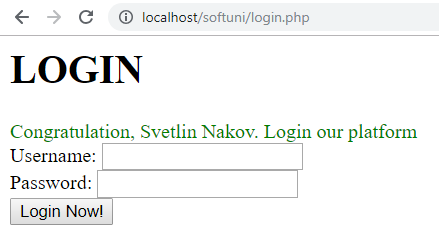


* Form **is submitted**, but **validation** did not pass:



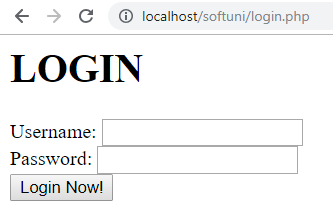
Show the **error only for the first field the validation did not pass** (they might all be out of range). The message is in the format:

* + For **text** fields: “{Field Name} must be between {MIN\_LENGTH} and {MAX\_LENGTH} characters!”.
  + For **number** fields: “{Field Name} must be between {MIN\_VALUE} and {MAX\_VALUE}!”
* **After** successful submission Congratulations, **“Full Name”**.

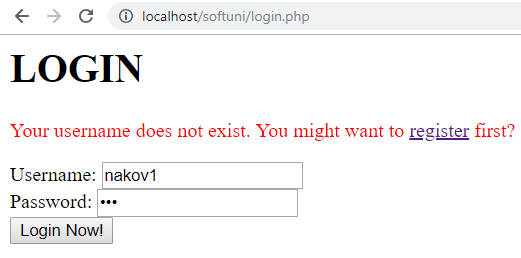


**login.php:**

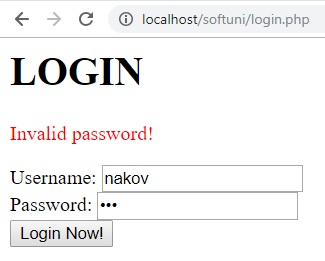
* **Before** form is submitted



* Form **is submitted**, but **username** does not exist:

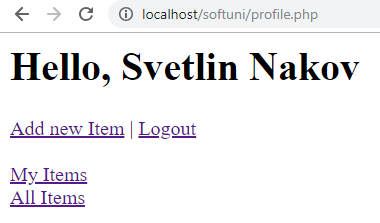


* Form **is submitted**, but the **password** is not the same:

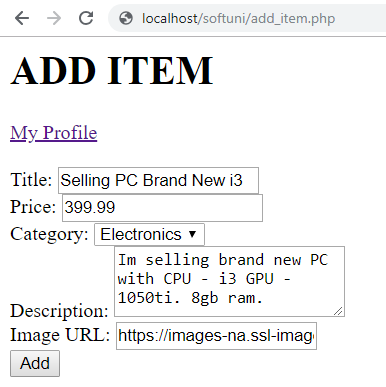


* **After** successful login:

You should get a greeting message with your **full name.** You must have 4 links for adding new item, for **logout** for viewing your items and for viewing all items



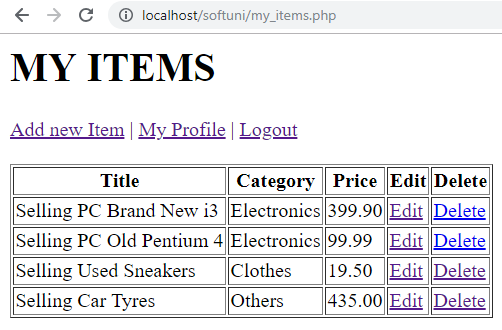
**add\_item.php** should look like this:



After adding an item you should be reditected to **my\_items.php**

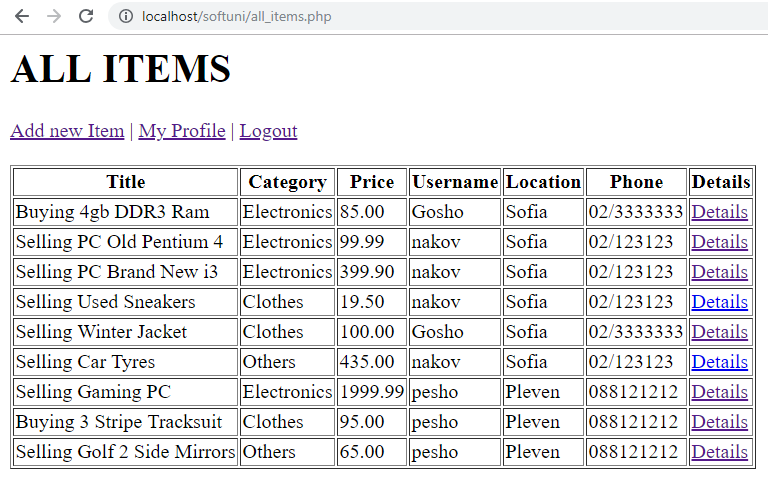
my\_items.php:

Sorted by Category ID Ascending then by Price Descending



**all\_items.php:**

Sorted by **Location Descending** then by **Category ID Ascending** then by **Price Ascending**



**edit.php?id={item id}** (Edit existing item):

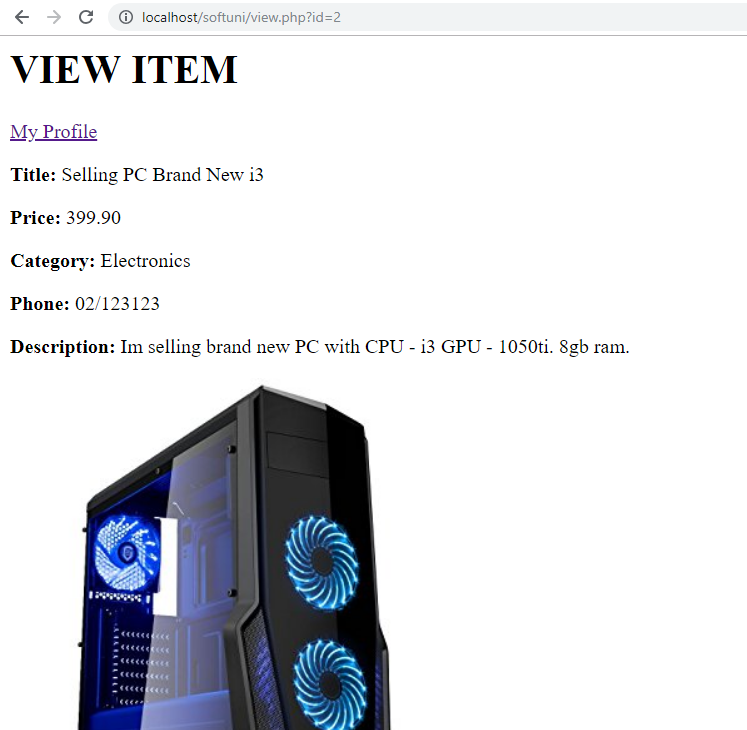


* After **successful submission**, the operation is **edited**, **redirect** to the list of my items **my\_items.php**

**delete.php?id={item\_id}** (Delete existing item):

* **Deletes** the item and **redirects** to of my items **my\_items.php**.

**view.php?id={item\_id}** Whenyou click “details” from **all\_items.php** you should be able to see the **title**, **price**, **category**, **user’s phone**, **description** and **image** about the item you selected. This option is available only for **logged in** users.



# Database

Create a database “softuni\_bazar” and add the following tables:

# Data Definition

# DDL – Users

|  |  |  |
| --- | --- | --- |
| **users** | | |
| **Column Name** | **Type** | **Constraints** |
| username | String | 4…255. Unique. |
| password | String | 4…255 |
| full\_name | String | 5…255 |
| location | String | 4…255 |
| phone | String | 6…255 |

# DDL – Categories

|  |  |  |
| --- | --- | --- |
| **Categories** | | |
| **Column Name** | **Type** | **Constraints** |
| name | String | 3…50 |

# DDL – Items

|  |  |  |
| --- | --- | --- |
| **Items** | | |
| **Column Name** | **Type** | **Constraints** |
| title | string | 3…255 |
| price | decimal | 1…50 |
| description | string | 10…10000 |
| image | string | 3…255 |
| category\_id | int | 1...4294967295 |
| user\_id | int | 1...4294967295 |

# DML – Add Categories

**Add** the following Categories: Save the DDL in export.sql file in the top-level directory of your project.

|  |
| --- |
| **name** |
| Electronics |
| Clothes |
| Other |

# HTML

# Register Form

**Create register form**. Follow the constraints below:

1. Username with minimum length of **4** symbols
2. Password with minimum length of **4** symbols
3. Confirm password with minimum length of **4** symbols
4. Full Name with minimum length of **5** symbols
5. Location with minimum length of **4** symbols
6. **Phone** with minimum length of **6** symbols
7. Button, submitting the form with value “**Register**”

All fields are **required**. Use a convention for name attributes: **lower\_snake\_case**.

# Login Page

1. Username
2. Password
3. Button, submitting the form with text “**Login**”

All fields are **required**. Use a convention for name attributes: **lower\_snake\_case**.

# New Item Page

1. Title with minimum length of **3** symbols and max length – **255**
2. Price with minimum length of **1** symbols and max length – **50**
3. Category which is **empty** **dropdown** (will be later populated with PHP)
4. Description containing **10 to 10,000** symbols
5. Image string containing url
6. Button, submitting the form with text “**Add**”

All fields are **required**. Use a convention for name attributes: **lower\_snake\_case**.

# Structure

You are required to have index.php in the project, even if it is initially empty. Consider all **object oriented best practices** you have been taught during the course.

# Initial File Structure

1. Create a file named “export.sql” holding CREATE DATABASE and all the CREATE TABLE statements. Put it in the top-level directory;
2. Create a directory named “Config” in the top-level directory. Put file named “db.ini” holding the database connection information under the following keys: dsn, user, pass;
3. Create endpoints (front layer) for all possible pages in the top-level directory: index, register, login, add\_item, edit, all\_items, my\_items, view and logout;
4. Extract common operation such as autoloading classes in “common.php”.

# Database Abstraction Layer

Create an **abstraction layer over the database** operations in three stages – **prepare -> execute -> take result**. All objects should be in the \Database namespace.

# Repository Layer

Create repository layer in the namespace \App\Repository

# Service Layer

Create service layer in the namespace \App\Service

# Item Trading Application

Make **all the layers to work**, according to the structure and the business requirement in **1.** and submit the **zip archive** in judge.